# XML Query Based on Ontology

Tao-Shen Li, Ting Han, Guo-Ning Chen

*Abstract*—**Document Type Definition (DTD) is a fundamental tool that enables users to constrain the structure of XML documents. This tool does not support semantic query. Therefore, it has recently become an active research topic in Web intelligence to endow XML with semantics for query quality. In this paper, we develop an extension of the DTD using for the provision of formal semantics.. It is implemented using an entity declaration in the DTD to describe the ontology and recurring this description to the form of Frame Logic. After validated by our extended DTD, a semantic valid XML document will be produced, so as to be queried using existing query languages (such as XQL). Our extended DTD is based on two main principles (1) maximizing the sharing of meta-data on the Web and (2) possibly using the DTD's provisions for reducing development expenses.**

*Index Terms*—**XML, DTD, Ontology, Frame logic, Semantic web**

## I. Introduction

THE XML is widely known in the Internet community and has become the lingua franca for data dissemination, exchange and integration on the World Wide Web. Nearly every data management-related application now supports the import and export of XML, and standard XML schemas and DTD(Document Type Definition) are being developed promoted for all types of data sharing. XML implements the requirements of the universal expression and syntactic interoperability because anything for which a grammar can be defined can be encoded in XML and an XML parser can parse ang XML data. When it comes to semantic interoperability, however, XML has disadvantage as follows[1]:.

• XML just describes grammars and can't recognize a semantic unit from a particular domain, because it is designed only for markup in document structure but did not consider the common interpretation of the data contained in the document.

• XML is useful for data interchange between applications, but not for situation where new communication partners are frequently added. Because new information sources continually become available and new business partners join existing relationships on the Web, it is important to reduce the costs of adding communication partners as much as possible.

Recently, more attentions have been paid to endowing XML with the semantic property for higher quality of query. There are several approaches of defining ontology representations for semantic purposes, such as extending query languages, developing ontology languages or linking to an existent ontology. The work in [2] has shown how abstract models of agent communication and content languages are strongly related to the notion of domain ontology, and has presented a common framework for these types of model. The integration of XML query languages with IR-style similarity comparisons for ranked retrieval of XML data on the Web is presented in [3]. And the work of [4] enrich ontology using specialization processes based on some heuristics in order to offer to the expert of the domain a decision-making aid concerning its field of application. These projects ultimately adopt the method of linking to an existent ontology or extending query languages. The imaginable trouble for them may be the complicated mapping process, which influences the efficiency of the query greatly.

In [5], the ontology language OIL as an extension of RDFS is described. As a result, a full knowledge representation (KR) language can be expressed in RDFS and the extended language can be a maximal backward compatibility with RDFS. [6] defines an ontology language OWL to escape the limitation brought by RDF and RDFS, and this language is built upon RDF and RDFS. The work of [5] and [6] is to develop a kind of new ontology language and such language may be integrated and self-governed, but the workload seems vast and for users, they have to learn a new language over again. The SemanticMiner project[7] uses Frame Logic to define their ontology, and indicates that F-Logic covers most parts of OWL and allows specifying axioms freely. Additionally F-Logic uses the same syntactical constructs for both modeling and querying the ontology.

In this paper, we discuss a ontology model based on F-Logic, which this model is the template for our semantic description. We then briefly present an extension of the DTD using for the provision of formal semantics. It is implemented using an entity declaration in the DTD to describe the ontology and recurring this description to the form of Frame Logic. Our approach uses DTD to translate XML document from users, only and such translation doesn't include the change of document type and structure, it just changes the name of relevant elements or its attributes partially and the output is still a XML document. On the other hand, DTD also embodies the concept of domain but lack of semantic layer, which is a big limitation for its use in semantic web. So, our work is a consummation for DTD. By comparison with [5] and [6], our method we simplifies the development process and embodied the kernel of ontology successfully. SemanticMiner project also extended the query language for their ontology defined by F-Logic, while we just appeal our semantic purpose to an extended DTD which is similar to F-Logic in form, or the method of extending DTD just adopts the ideology and form of F-Logic.

The rest of this paper is structured as follows: Section 2 presents the basic concepts of the XML language and ontology. In section 3, the issues existed in the XML query will be discussed. Then ontology will be proposed as a solution. An implementation of our

This work was supported by the Natural Science Foundation of Guangxi,China under Grand Number 0229008.

Tao-Shen Li works at the Department of Computer Science and Technology, Guangxi University, Nanning, 530004 China (phone: 86-0771-3236627; fax: 86-0771-3236627; e-mail: tshli@ gxu.edu.cn).

Ting Han is a MS candidate in computer application technology from the Guangxi University, Nanning, 530004 China (e-mail: bobobye@126.com).

Guo-Ning Chen works at the Department of Computer Science and Technology, Guangxi University, Nanning, 530004 China (e-mail: chengu@cs.orst.edu).

approach will be shown in section 4, which uses entity declaration to extend the DTD and follows the form of Frame-Logic to describe ontology. A part of codes of our experiment and a query example will be also provided later. Finally, our conclusions and future work are presented.

## II. Basic concepts

### A. XML Language

XML has been a new standard for information exchange on the World Wide Web. It allows users to define labels according to their interests and there are no syntax criterions when people write them, so it has a high flexibility. However, extensible characteristic of XML is its merit, but a biggest limitation of itself as well. The reason for this amphibious problem is that there is no way for server to understand labels defined by user himself. Traditional DTD is a basic tool that enables users to constrain the structure of XML documents. It can enforce constraints on which tags to use and how they should be nested within a document, but no help to the semantic aspect. Now, people are all making great efforts to find a validation mechanism for XML documents. One validation mechanism is XML Schema, which can offer more data types and also support naming space. The other is RDF (Resource Description Framework) which adopts URI to locate resources on the web accurately. These mechanisms are all trying to find an end-result for every resource, but they cannot establish semantic relations between elements or resources inside documents. Fortunately, philosophical ontology provides us a chance to solve this problem..

### B. Ontology

In philosophy, ontology is the study of the kind of things that exist[8]. In the computer field, ontology is a sort of specific representation and description for conceptualization object using certain language, so it depends on the adopted language. According to the formalization degree of representation and description, ontology can be divided into absolute informalization, half formalization and rigid formalization. Ontology with higher formalization degree will be more favorable for computer to deal with automatically. From the definition of conceptualization, we can conclude that the terms, definition of terms and semantic web between terms in certain domain are information that should be included in the domain ontology.

Ontologies has played a key role in many fields[1,9], such as knowledge processing based web, share and Reusable Software. Ontologies are used in e-commerce to enable machine-based communication between buyers and sellers; vertically integration of markets; and description reuse between different marketplaces. Search engines also use ontologies to fine pages with words that are syntactically different but semantically similar.

If we use ontologies to define the shared conception hierarchy of certain domain, it will provide simple and comprehensible subjects which are used for communicating between the person and the application system. According to the limitation existed in the process of XML validation using traditional DTD, this paper presents an effective validation methodology based on ontology, which defines the terminology of a domain, provides a sound semantics, and formalizes relationships between the terms, i.e. it provides rich background knowledge, and implements the description via frame logic. Our method makes it possible that XML document could be semantic valid, therefore, improves the quality of query by a long way.

## III. Issue

Traditional DTD, using simple syntax, provides effective validation mechanism for user. User can customize the DTD himself to limit XML document structure conveniently, and defines labels he likes consequently. If his cooperative fellows agree on this common DTD, then all documents can be kept consistently in the process of building, transferring, importing or translating the documents.A simple DTD document (Example1.dtd) is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % P "(#PCDATA)">
<!ELEMENT general (introduction, cooperation?)>
  <!ELEMENT introduction (corporation*)>
    <!ELEMENT corporation (name, support*)>
      <!ELEMENT name %P;>
      <!ELEMENT support %P;>
    <!ATTLIST corporation corporation_ID ID
      #REQUIRED>
  <!ELEMENT cooperation (item*)>
    <!ELEMENT item (technology, partner*)>
      <!ELEMENT technology %P;>
      <!ELEMENT partner %P;>
<!ATTLIST item item_ID ID #REQUIRED>
```

The Example1.dtd above can make validity validation for following example of XML document (example1.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE test SYSTEM "test.dtd">
<general>
  <introduction>
    <corporation corporation_ID ="_001">
      <name>BenTeng Computer Corporation</name>
      <support>software development</support>
    </corporation>
    <corporation corporation_ID ="_002">
      <name>AnKang Pharmaceutical group</name>
      <support>bio_pharmacy</support>
    </corporation>
  </introduction>
  <cooperation>
    <item item_ID ="_1001">
      <technology>software development</technology>
        <partner>BenTeng Computer Corporation</partner>
        <partner>ShengLi Computer Corporation</partner>
    </item>
  </cooperation>
</general>
```

From example above we can see that the basic function of traditional DTD validation is to prescribe logic structures of documents. It defines elements of page, attributes of element and relations between elements and attributes, such as the order or appearing times for the sub-elements of certain element. It also defines whether an element has value and what the data type of its value is, etc. So, the action of the DTD is to interpret all details that a valid XML document needs to be knew, just like a syntax analyzer.

Well then, is server able to meet the requirement of user when user asks for processing? First, let's come to a query request as follow:

```
//corporation/name[../support="software development"]
```

This query sentence is written according to the syntax of XQL. Its requirement is to find out the corporations which have technology of software development and return the list of corporation name to user. However, if we analyze the semantic of document further, we will easily find that *ShengLi computer* has cooperative relation with *BenTeng computer*. That is, *ShengLi computer* has the technology of software development too. Its name should be also returned to user. But, such result can only be obtained using following query sentence:

```
//corporation/name[../support="software development"]
$union$
//item/partner[../technology ="software development"]
```

Obviously, server cannot build query sentence above if we only rely on the traditional DTD. How can we solve such problem? Practical test shows that we can make such query easy to accomplish if we use an ideology of semantic validation based on ontology introduced in the following section.

## IV. Implementation

### A. Frame Logic

People use semantic network and frame logic to describe hierarchy relations and association relations between concepts in artificial intelligence (AI). Hierarchy relations use the way of the diagram example to describe different and relative views between concepts, while association relations are structured representations about individual class.

Considering the goal of trying to form a corresponding relation with the DTD hierarchy, we use frame logic to describe domain ontology. "F-Logic is a deductive, object oriented database language which combines the declarative semantics and expressiveness of deductive database languages with the rich data modeling capabilities supported by the object oriented data model."[8] In the following parts, we will present an ontology document example2.dtd. It is an ontology description based on frame logic. It describes a domain including people and corporation. The content of this document is composed of the hierarchy of the domain, relations between concepts and some axioms or rules. There are three sections in it. First section of the ontology describes the notional level of the domain, where "::" shows a inclusive relation that the concept on the right is the upper one to the concept on the left. Second section describes the unification of concepts. It introduces the concepts into the attributes definition, defines attributes of the concepts in the "[ ]", and explains the data type definition of the attributes with "=>>". The last section describes some accepted rules or relations, where "<->" shows these relations.

```
(example2.dtd)
 domain[ ].
    people :: domain.
        employee :: people.
            technician :: employee.
                programmer :: technician.
                    bachelor :: programmer.
        student :: people.
            bachelor :: student.
    corporation :: domain.
        State corporation :: corporation.
        foreign capital corporation :: corporation.
            IT company :: foreign capital corporation
        joint-stock corporation :: corporation.

people[name=>>STRING;email=>>STRING;company
    =>>corporation; address =>>STRING].
employee[employee ID=>>bachelor].
technician[supervise=>>programmer].
programmer[cooperate  with =>> programmer].
bachelor[administer =>>technician].
student[student ID=>>NUM].
corporation[incorporator=>>people;name=>>STRING;
    property=>>STRING;summary=>>STRING].
foreign capital corporation[linkman=>>people; corporation
    ID=>>NUM; number of employee =>>NUM; calling=>>IT
    company].
IT company[property=>>corporation].

FORALL Jerry, Tom
    Jerry: programmer[cooperate with ->> Tom] <->
        Tom: programmer[cooperate with ->> Jerry].
FORALL Jerry, certain corporation
    Certain corporation:corporation[incorporator->>Jerry]<->
        Jerry: people[company ->> certain corporation].
FORALL Jerry, certain foreign capital corporation
    certain foreign capital corporation: foreign capital
corporation[linkman ->> Jerry] <->
        Jerry: people[ company ->> certain corporation].
FORALL Jerry, Tom
    Jerry: bachelor[administer ->> Tom] <->
        Tom: technician[supervise ->> Jerry].
FORALL corporation A, corporation B
    corporation B: foreign capital corporation[calling ->>
        corporation A] <->
corporation A: IT company[property ->> corporation B].
```

### B. Semantic Validation Using the DTD

Example2.dtd exhibits the advantage of ontology when describing conceptural hierarchy. According to this advantage, two concepts which have no direct relation could be associated via their common upper concept. Thus, the semantics of document can be understood on the higher level.

The substance of the ontology is to show a kind of inherited relation between concepts. If a concept owns a upper concept, then it will inherit the attributes of its upper concept naturally. We can also conclude that if there are two concepts having some common attributes, it is possible that they have the same upper concept. Thus, concepts are constrained to a semantic intersection chain which the concept has itself. Regardless the methods to express the concept are different, all concepts should be found as long as they have same semantic. However, whether need to define a new language that carry on formalization description to the ontology? Obviously, this task is very complicated and enormous. Considering that traditional DTD has had a powerful function for structure validation, if we use the DTD to carry on formalization description to the ontology and introduce ontology to the DTD, it will be the most valid and efficient way. That is the basic idea of this paper. Figure 1 shows the framework description of the tasks to need to be complete for implementing this idea.

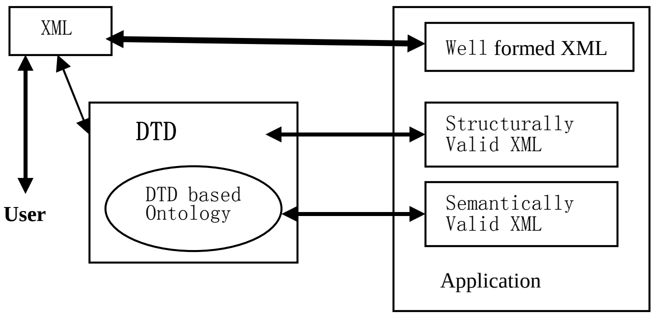

Figure 1 framework description of the task for implementing DTD semantic validation

Before introducing ontology to the DTD, first problem we need to solve is what should be used to define ontology in the DTD. It perhaps is the method that people first thought of to define every ontology as an element. However, concepts often exist in the definition of attributes in the actual circumstance, or we can say more constitutionally that some concepts contain some attributes defined by another concept. So, it is not enough that just defining every ontology as an element simply. Therefore, we consider adopting an extended method that using element to define ontology and introducing the defined concept to the attribute definition at the same time. The result for doing so is that we can find the semantic root of every concept. This ideology seems like the definition of class in OOP (Object-Oriented Programming). Conclusion of our analysis is that inherited relations between concepts are produced at the same time they are defined, in other words, such relations are inherent.

The next problem to solve is how to implement inherited relations between concepts in the DTD. Because what we need to hold is the semantic contents of the document, and these contents are defined only in the values of element or attribute, so the problem will be translated into the definition of values. In the DTD, there are ten kinds of inner data types and an entity declaration method for values definition. Among them entity declaration makes values definition to become more flexible. It can be used to define reused data blocks or to cite non-XML data for simplifying the DTD and enhancing the readability. However，it is to be noticed that entity is a placeholder representing content and has inherent substitution ability, and the substituted content could be a meaningful phrase or concept. The key point of this paper is to develop and emphasize this powerful ability contained in entity declaration of the DTD, and apply this ability to the attribute definitions that contain some concepts. Consequently we can implement associations between concepts through ontology. From this point of view, entity transfers the common information between concepts just as an excellent carrier of ontology element, and also produces more meaningful ontology elements. In fact, this process is a kind of evolution for concepts. Certainly, in order to implement the inherited relation between concepts, the concept transferred by entity and the concept to be defined must be in the same semantic chain.

The special point of our semantic validation is that what entity declares is an ontology element which has been defined. With that we can define other elements that are semantic interrelated with it and produce new ontology. For example, in <!ELEMENT branch (#PCDATA | %tree;)* >, new ontology element branch is defined by ontology element tree which has been defined and inherits its attributes at the same time. If we continue to associate all the concepts like that, semantic chains between concepts will be established. It is obvious that the description process of whole document is the conformation process of ontology actually.

## C. *Example of Semantic Validation Using the DTD*

According to the design idea above, we performed some researches and experiments. In the following parts we shall list a part of codes and illustration. In these DTD documents, the three parts of ontology description given in example2.dtd can also be expressed in the document.

```
<!-- entities for realizing the is-a hierarchy -->
<!ENTITY % people "people | employee | student | technician |
   programmer | bachelor " >
<!ENTITY % programmer "programmer | bachelor " >
<!ENTITY % corporation " corporation | state corporation |
   foreign capital corporation | joint-stock corporation | IT
   company " >
<!ENTITY % foreign capital corporation " foreign capital
   corporation | IT company " >

<!-- element declarations for ontology concepts -->
<!ELEMENT people (#PCDATA | name | email | company |
   address)*>
<!ELEMENT programmer (#PCDATA | name | email |
   company | address | employee ID | administer | cooperate
   with)*>
<!ELEMENT corporation (#PCDATA | incorporator | name |
   property | summary)*>
<!ELEMENT foreign capital corporation (#PCDATA |
   incorporator | name | property | summary | linkman |
   corporation ID | number of employee | calling)*>
<!ELEMENT IT company (#PCDATA | incorporator | name |
   property | summary | linkman | corporation ID | number of
   employee | calling | property)*>

<!-- ATTLIST declatation for ontology attributes -->
<!ATTLIST people
   name     CDATA #IMPLIED
   email    CDATA #IMPLIED
   company  CDATA #IMPLIED
   address  CDATA #IMPLIED>

<!-- element declaration for ontology attributes -->
<!ELEMENT incorporator (#PCDATA | %people;)* >
<!ELEMENT linkman (#PCDATA | %people;)* >
<!ELEMENT address (#PCDATA) >
<!ELEMENT company (#PCDATA | %corporation;)* >
<!ELEMENT cooperate with (#PCDATA|%programmer,)*>
```

The first part of this DTD document is the entity declaration which translates inherited relation between concepts in ontology into substitution relation between concepts. The second part defines the value and attribute of upper ontology elements which are corresponding to the entities declared in entity declarations and belong the outmost layer of the frame logic. The third part implements inherited relation of concepts actually and uses entity declaration of defined ontology elements to define new ontology, which consequently makes new ontology inherit values and attributes defined in the second part and the new ones become the nether concept of the defined ontology.

Though the DTD above is a subset of this domain description, we can still find that nether concept inherit its upper concept very well through declaration of parameter entity, so that it contains either the commonness of the domain which it is affiliated to or its own individuality. This method presents a valid way to help extracting common information between concepts or searching for concepts through some information.

### D. *Query Based Ontology*

The functionality of ontology in our approach can be generalized as defining a common vocabulary and improving the quality of query answers. Using the DTD based ontology discussed above, we can validate XML documents much further from semantics. Our experimental system implemented the process of translating users' XML documents into the semantic valid XML documents according to our extended the DTD. After translating, elements having semantic association in XML will be found through a common element which may be a unattached element or be showed in element attributes.

Let's come back to the example1.dtd presented in section two. We know that the meanings of partner in element item indicate certain corporation, and the technology of the cooperation is just the attribute of partners. So, we need to define substitution of corporation for partner, then naturally, the technology of partner becomes the attribute of corporation named support. Following entity definition implements this target:

```
<!ENTITY % corporation "corporation | partner">
<!ENTITY % support "support | technology">
```

Certainly, we should define element to support the attribute of corporation, just like:

```
<!ATTLIST corporation
      corporation_ID ID #REQUIRED
      name CDATA #IMPLIED
      support CDATA #IMPLIED>
```

In succession, we can define technology and partner using defined entity -- %support and %corporation:

```
<!ELEMENT technology (#PCDATA | %support;)*>
<!ELEMENT partner (#PCDATA | %corporation;)*>
```

According to amended example1.dtd, example1.xml was translated into the new one following after our processing:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE test SYSTEM "test.dtd">
<general>
  <introduction>
     <corporation corporation_ID ="_001" name = "BenTeng
        Computer Corporation" support = "software
        development"/>
    <corporation corporation_ID ="_002" name = "AnKang
       Pharmaceutical group" support = "bio_pharmacy"/>
  </introduction>
  <cooperation>
    <item item_ID ="_1001">
       <corporation name = "BenTeng Computer Corporation"
            support = "software development"/>
       <corporation name = "ShengLi Computer Corporation"
            support = "software development"/>
    </item>
  </cooperation>
</general>
```

From the above-mentioned translation result, we can see that effect is remarkable The element partner has been replaced by the element corporation and has the attribute support. Now, we can implement the simple query for XML document as we wish and the query answers will never miss some information correlated with the request. Our query is just request for the name of corporations who have software development technology. For well output we limit the format:

```
<result>
{
  for $c in doc("H:/xml/testdata/test.xml")//corporation
  where $c/@support = "software development"
  return
   <corporation>
    { $c/@name }
   </corporation>
}
</result>
```

Just as we expect before, we get the corporation name not only "*BenTeng Computer Corporation*" but also "*ShengLi Computer Corporation*"

```
<result>
    <corporation name="BenTeng Computer
        Corporation" />
    <corporation name="ShengLi Computer
        Corporation" />
</result>
```

The experiment shows that the method we presented in this paper can improve veracity of information retrieval remarkably. This trait is a big help for users who need precise query. In additions, ontology using the DTD description has well maintainability. The introduction of new ontology is accomplished with definition of its elements and has no influence to the defined ontology elements, namely new ontology is just linked to the semantic chain simply and the link point is the upper concept associated with it directly. After this, we can define offspring belonging to this new ontology itself and extend the semantic chain.

Certainly, the unification of ontology and XML is not just limited to the DTD, but the simpleness of the DTD and its strong suit for describing structured data provide ontology with a simplest and easiest carrier. With the perfectness of relevant criterion of XML, we believe that the unification of ontology and XML schema will be deepened continually, so will be the semantic comprehension for XML document based content.

### *E. Intelligent XML Query System*

As we showed above, our method has several important advantages for XML files querying. First, concepts with the same meaning can be translated into a formal expression which exists in Semantic Validation the DTD where a common vocabulary of system has been defined. Second, concepts with no relationships can be also associated in terms of rules defined in Semantic Validation the DTD. On the other hand, XML files after translation have uniform labels and coherent contexts, which means that users can get same format answers that are more understandable and follow the habits of users. Based on these features, putting our method to application will be helpful. We present here an implementation frame for XML query system based on ontology. It contains a Semantic Validation the DTD as semantic model of system.

The infrastructure of System takes charge to accept XML files' register, put XML files into format validation, structure validation from the DTD included and finally semantic validation from Semantic Validation the DTD of system. Then, XML files rejected by semantic validation will be translated into formal expressions and stored into database with all the eligible XML files. We can set a validation module and a translation module to implement these tasks through interacting with the Semantic Validation the DTD, and let them give their attentions to two points, one is whether the concepts contained in the labels are formal, and the other is whether the nested structure of concepts is coincident with the successive relationships between them.

Based on the data stored in the database, we can establish our applications. We start this from User Interface which receives users' query using a series of forms (XForms is a recommended standard now). After parsing, we translate these forms into queries written in XQuery language, then they can be performed by XQuery Engine directly. From database, finally, XQuery Engine gets all the information user want and displays the answers on the User Interface.

Here, we present a common framework for XML query application. Actually, because our method can be designed into an unattached module, many existing information systems can be integrated into their infrastructure to provide semantic support expediently. The only problem is that the model of the Semantic Validation DTD needs to be accepted widely. Now the easiest and soonest way we can take is to write it according to existing ontology (or domain ontology). We think that our method will be more compactly for intellectualized upgrade of information system.

## V. CONCLUSION

The research on XML Query is currently increasing because of the wide dissemination of business data over the Web. We argue that the main problem related to the Query of XML is that the query result may neglect the content having semantic relevancy to the user request. That is, when there are several names for the same information, the trouble omes. Because of this, very heterogeneous ontology models arise for XML Query when we compare different models.

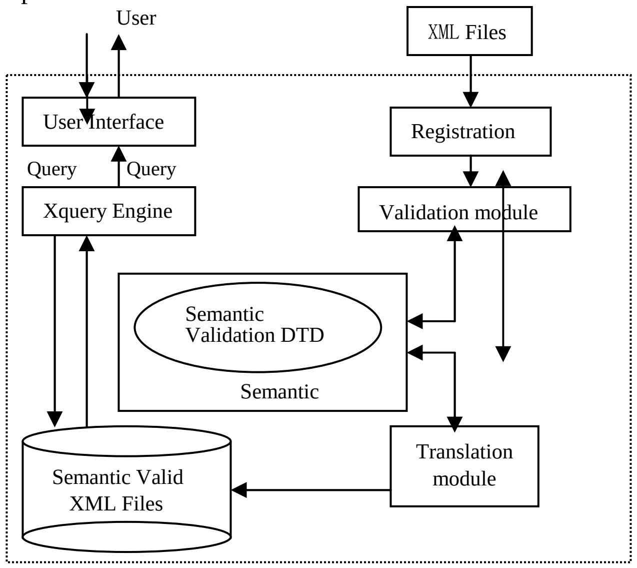

Figure 2 XML Query System Based Ontology

Our contribution to this problem is a method using ontology to enrich the DTD. Particularly, recurring to entity declarations, we deal with the problem of how to represent ontology in the DTD.

Another strong point of our approach is that we use the fabric of f-logic to establish ontology, therefore make this abstract concept comprehensible.

Currently, we are extending our method with an analysis of XML instances in order to improve the definition of query language. After finishing the extension, we will continue to optimize the quality of query.

Other current works are related to the quest for a better carrier of ontology, as well as the research for DAML+OIL and OWL in vogue.

## REFERENCES

[1] S. Becker, S. Melnik, F. Van Harmelen, et al. The Semantic Web: the Roles of XML and RDF. IEEE Internet Computing, September-October 2000, 63—75
[2] Stephen Cranefield, Martin Purvis, Mariusz Nowostawski. Is it an Ontology or an Abstract Syntax? Modelling Objects, Knowledge and Agent Messages. The Information Science Discussion Paper Series, Number 2000/08, April 2000, ISSN 1172-6024
[3] Anja Theobald, Gerhard Weikum. Adding Relevance to XML[J]. Modern Information Retrieval. WebDB (Informal Proceedings) 2000: 35-40
[4] E. Desmontils, C. Jacquin, L. Simon. Ontology enrichment and indexing process. Research Report, 2003, No 03.05
[5] Jeen Broekstra,Michel Klein,Stefan Decker, et al. Enabling knowledge representation on the Web by extending RDF Schema. Computer Networks 39 (2002) 609－634
[6] Grigoris Antoniou1, Frank van Harmelen2. Web Ontology Language: OWL. Handbook on Ontologies in Information Systems. Springer-Verlag 2003
[7] E. Moench, M. Ullrich, Hans-Peter Schnurr, et al. SemanticMiner–Ontology-Based Knowledge Retrieval. j-jucs, volume 9 (2003) issue 7, 682-696
[8] K. Hori. An ontology of strategic knowledge: key concepts and applications. Knowledge-Based Systems, volume13 (2000), 369-374
[9] B. Chandrasekaran, J. R. Josephson, V. Richard Benjamins. What Are Ontologies, and Why Do We Need Them?. IEEE Intelligent Systems, January-February 1999, 20-26
[10] Liao Minghong. Ontology and Information Retrieval. Computer Engineering, 2000,26(2): 56－58
[11] Wan Jie, Teng Zhiyang. Application of Ontology in Content-based Information Retrieval. Computer Engineering, 2003,29(4):122－123,152