Building an Ontology-Based Community Portal for Scientific Applications

Ching-Long Yeh and Jia-Yang Chen
Department of Computer Science and Engineering
Tatung University
40 Chungshan N. Rd. 3rd Sec., Taipei
Taiwan
chingyeh@cse.ttu.edu.tw, g9106028@ms2.ttu.edu.tw

Abstract

The Semantic Web provides a metadata layer upon the information pool of the current web. The metadata layer is represented by using machine processable language according to the schema of ontology. We have developed an ontology-based system to manage the content of the metadata layer. In this paper, we give an overview of the system architecture and then show the application of the system for scientific web sites. We follow a simple knowledge engineering step to build ontology for the fish domain, and then create metadata for unstructured, semi-structured information usinf annotation tool and wrapper program. After creating the metadata in RDF we import them into the knowledge warehouse of the system and then show the functionality of the conceptual search and semantic navigation. We have made a performance evaluation for comparing building the knowledge schema component using either Prolog or frame-based system. The result shows that the latter achieve better performance.

1. Introduction

In the age of information glut, it is necessary to find a good way to manage information to help human consumption. The Semantic Web technology advocates a metadata layer beyond the information pool of the current web. The content of the metadata layer is represented by using machine processable languages, for example, RDF [1] and Topic Maps [2]. Thus automatic and intelligent services, other than the search engine and directory navigation services of the current web can be made based on the metadata layer. The goal of the Semantic Web is to create a universal medium for the exchange of data and to facility to put machine-understandable data on the web on interconnect personal information management, enterprise application integration and global sharing of commercial information. We have developed an ontology-based portal system architecture that supports intelligent services for user to access the content at a higher level of abstraction [14]. In this paper, we aim at demonstrating the use of the system to build a community portal for scientific applications.

The Semantic Web framework can be briefly summarized as providing a metadata layer in content-interoperable languages, such as RDF [1] and Topic Maps [2], which intelligent or automatic services can be made by machines based on the layer. As a meta-layer architecture upon the information pool of the current web, a system of the layer consists of: a knowledge warehouse that is used to store the metadata along with the knowledge schema, a front end component that provides various automatic or intelligent services for user, and a backend component that accepts the metadata of the information pool and saves in the repository.

To build a community on the system, the first step is to construct ontologies for the domain in question. Reuse of exising ontologies is encouraged as described in former research [3]. In this paper, we consult the ontologies used in scientific community portals, such as KA2 [4], ITTalks [5], and adapt to make the ontology for the purpose of this research. The ontologues are used as the schema of the knowledge warehouse. The metadata for unstructured information is created using an authoring tool. Semi-structured infromation, such as relational database, is converted into RDF by using wrapper programs. Metadata in RDF can be created using ontology editor, such as Protégé-2000 [6]. The metadata from various sources is coverted into the knowledge warehouse using the converters provided by the system. Then we can test the function of the community portal.

In Section 2, we describe the architecture of the ontology-based portal system. In Section 3, we describe ontology construction and metadata collection. In Section 4, we describe the functionality of the system by inserting a knowledge warehouse of scientific domain. Finally we make conclusion.

2. System Architecture of an Ontology-based Portal

We construct an ontology-based web portal as showed in Figure 1. There are three main components in this architecture: back end, knowledge warehouse and front end service. The goal of the back end is to act as the interface of metadata sourced from the outside world to the system. As mentioned previously, metadata is associated with resources ranging from unstructured to structured documents. We design an authoring tool that fetches unstructured document, like plain text or HTML document. The tool provides user with a graphical interface that user can mark up portion of text and add appropriate metadata for that part of text. The authoring result can be exported either in standard content representation language, like RDF, or the language used by the inference engine. Alternatively, the backend accepts standard metadata in RDF. We use Protégé 2000 [6] to produce RDF documents for the purpose of testing.

In the middle of the architecture is the knowledge warehouse component that functions as the knowledge source consulted by the inference engine. It provides repository to store ontology schema and instances of metadata. Since in this paper we employ the inference engine of a frame-based system, *flex* [7], an essential consideration is that the content of the repositories must be accessible by the inference engine. An application written in *flex,* is composed of a knowledge base and a number of service routines. The former part consists of frames and instances of frames that correspond to ontology schema and metadata instances, respectively. The latter is a number of inference rules that interact with user to achieve user's goal. The repository of the knowledge warehouse component can therefore be seen as the extension of the knowledge base of a *flex* application.

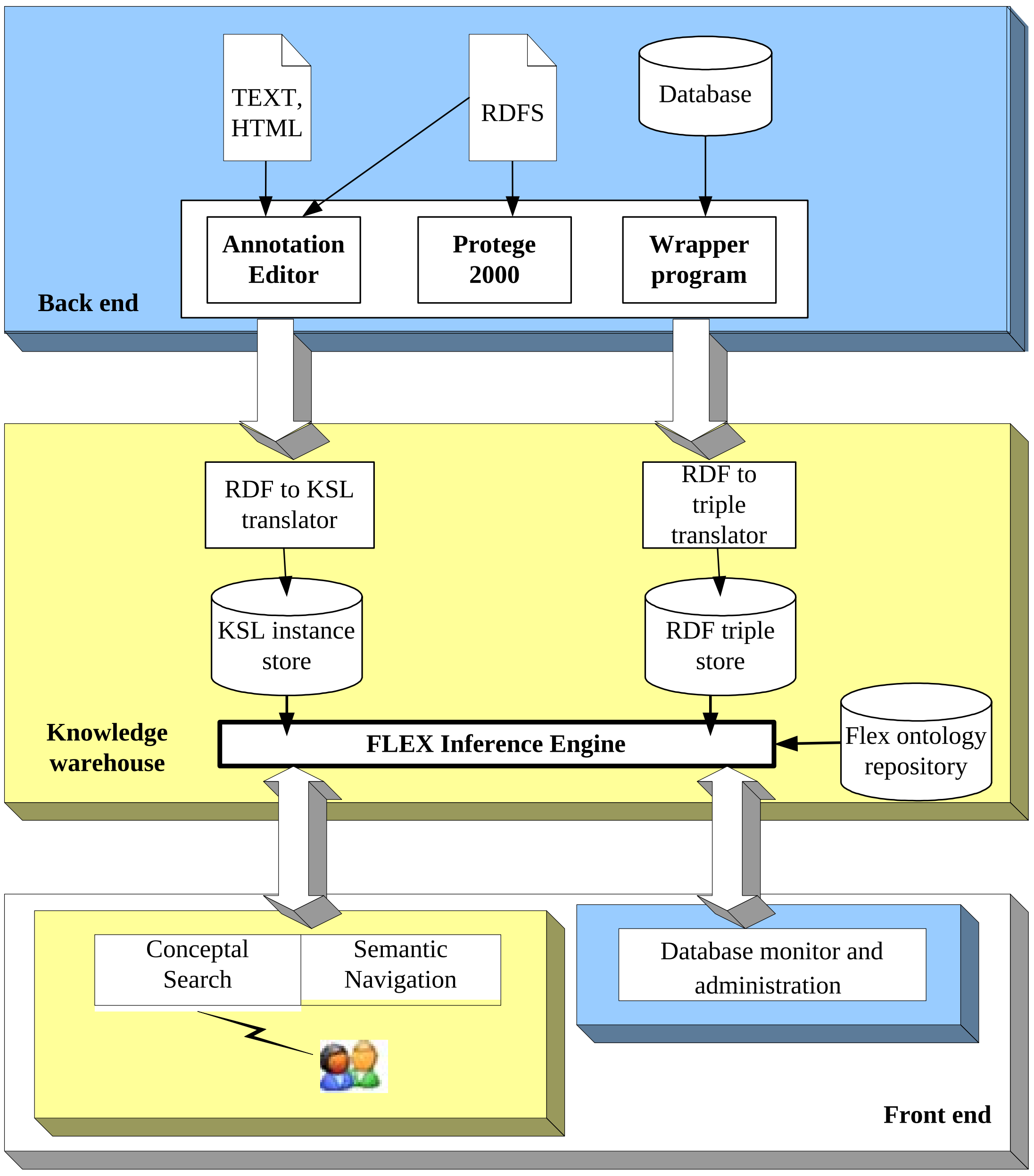


Figure 1: System architecture of an ontology-based portal

According to their functions, we divide the repository into two parts: one for ontology schema and the other for instances of metadata. The former is to be loaded into the frame system; thus the ontology schema is converted into the frame representation and then stored in the appropriate directory. For the latter part, we develop a RDF triple store based on a relational database. Alternatively, we provide file-based store of the metadata instances in frame instances. Translators need to be developed to convert RDF documents into triples and frame instances. The converted results are then stored in the repository accordingly.

The front end service component is to provide service functions for user and administrators to access or modify the metadata contents. In this paper, we focus on the development of discovery services, including the conceptual search and semantic navigation, to access the metadata content stored in the repository. The conceptual search and semantic navigation is based on the *flex* inference engine. User can use this interface to access information on the web site. It is included a tree navigation and data search. We also provide a query language for advanced query.

3. Collecting Metadata and Storing in Knowledge Warehouse

In this section we describe the process of enriching the content of the knowledge warehouse. We first establish the schema of the knowledge warehouse. Then we store the metadata collected from sources of different structures in the knowledge warehouse.

3.1 Ontology construction

Many people acknowledge the advantage of using ontology on their applications. However, it is not an easy task to build ontology for specific domains of applications. It requires knowledge that involves various stakeholders including domain experts, system analysts and technologists. In large and complex application domains, the construction of ontology can be even lengthy and costly. Thus adopting well-established steps towards the creation of ontology becomes a crucial issue when building a knowledge warehouse of an ontology-based portal. The knowledge engineering methodology [8], for example, provides simple while precise steps towards the creation of ontology building from existing sources or starting from scratch. In this paper we create ontology by consulting existing sources. Before resorting to existing sources of ontology, we have to determine the domain and scope to guide the acquisition of the ontology.

When we determine the scope of the knowledge domain, we can find the some of ontology in the ontology library of DAML, for example, or the other sources of ontology. We choose suitable ontology from the ontology library and trim the unnecessary class and properties. For example, in our ontology we would like to describe the concepts of people, events and things. We find them in the ontology library and we get the ontology of KA2. The ontology of KA2 provides some classification of the scientific domain and it describes some relation about people, events and things. The KA2's ontology describes 'event', 'organization', 'person', 'product', 'project', 'publication' and so on. They want to use ontology to aid them to do the knowledge acquisition.

We try to manage the information about scientific domain and provide the relevant knowledge for the expert or researcher of the favorite domain. About the general or standard information such as the group meeting announcement, conference declaration and some important activity, we can announce these data according to the definition of metadata. We can use the class of ‘event’, ’organization’, ‘person’ in KA2’s ontology to describe the general information. About the information of research like the information about fish and animal, we also want to find a suitable ontology to describe these data. We get the fish information from the fish database of Academia Sinica, and we are according to the classification system of the fish web site and its data to build the hierarchy of data. After we determine what kind of the thing, the general information or the information of research, which we want to process, we can get a suitable ontology and begin to choose and modify the ontology for our project.

There are three main approaches to aid large-scale ontology construction. The first one facilitates manual ontology engineering by providing natural language processing tools, such as editors and ontology import tools [6, 11]. The second approach relies on the machine learning and automated language processing to extract concepts and the relation of the concept. The last approach combines the first one approach with the second approach, but few systems use both approach. There are many tools about the first approach, such as Kaon [9], Protégé-2000 [6], Chimeara [10], and OntoEdit [11]. In this thesis we employ this approach to build our ontology. This approach can save a lot of time to rebuild the similar ontology and we can reuse the similar ontology by importing and modifying the different concept. There are researches using the second approach with natural language processing [12]. They make use of WordNet [13] as the knowledge source to extract the semantic concepts and try to automate fetch the important concept in the complex document.

Creation of computer accessible knowledge begins between the two extremes of very formal and very informal knowledge. In general, we will process the informal knowledge for a kind of formal knowledge. We in common use the method of adding annotation to describe the data but annotating information sources by hand is a time consuming task. Therefore we want to use a semi-automatic or automatic tool to help people annotate information sources. We talk about un-structured data, semi-structured data and structured data how to transform into the knowledge warehouse.

3.2 Creation of metadata instances

About Un-structured Data

How to get the useful information from the un-structured document is an important thing for the issue of the information collection. We develop an annotation tool for semi-automatically processing the data now that machine cannot automatically process the un-structured document. The kind of annotation tool builds for facilitating the annotation of the unstructured information. We know, there are many information appearances in the HTML document, but the computer cannot access this information. These data just display for people reading not for machine reusing. So we use the annotation editor to choose the important information and annotate them with node label. The node label lights up the significant message in the pure text area, such as the reporter of that seminar, the time of the speech and the address of the group meeting. We can use this method to semi-automatically transform the informal knowledge into the formal knowledge and store them in the knowledge warehouse. Figure 3-1 shows that we have built an annotation editor to help people to create the knowledge from the un-structured data.

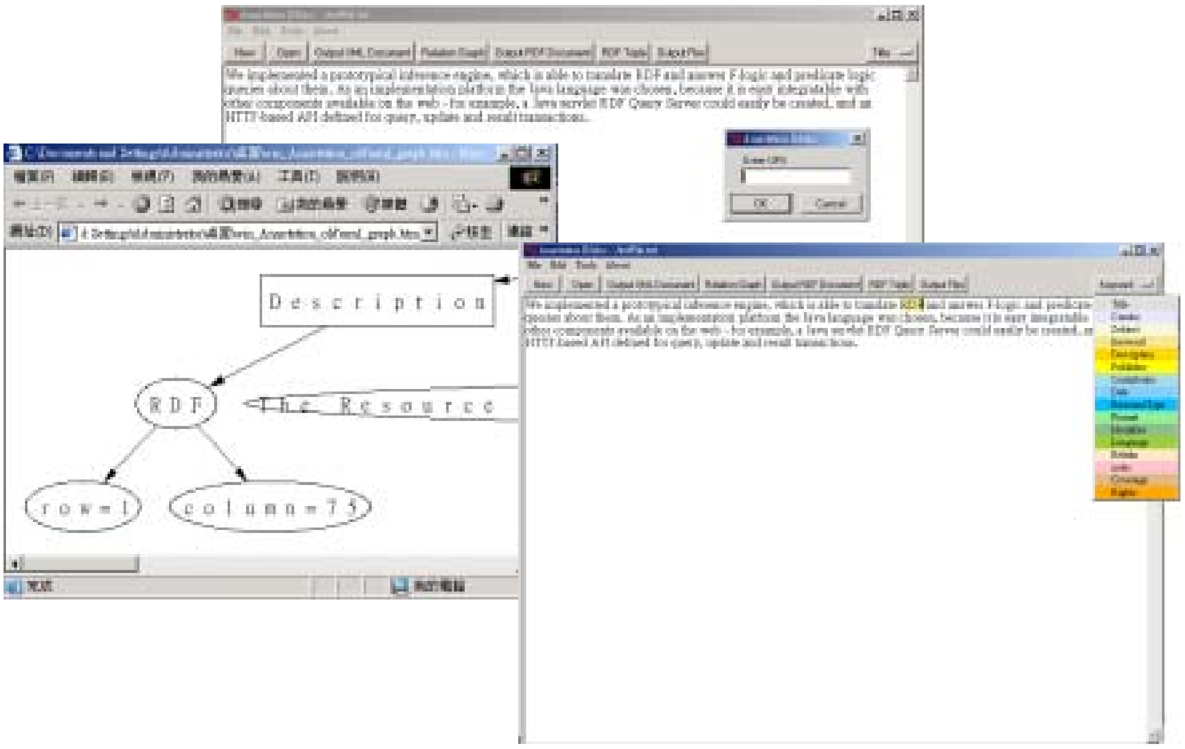

We implemented a prototypical inference engine, which is able to translate RDF and answer F-logic and predicate logic queries about them. As an implementation platform the Java language was chosen, because it is easy integratable with other components available on the web - for example, a Java servlet RDF Query Server could easily be created, and an HTTP based API defined for query, update and result transactions.

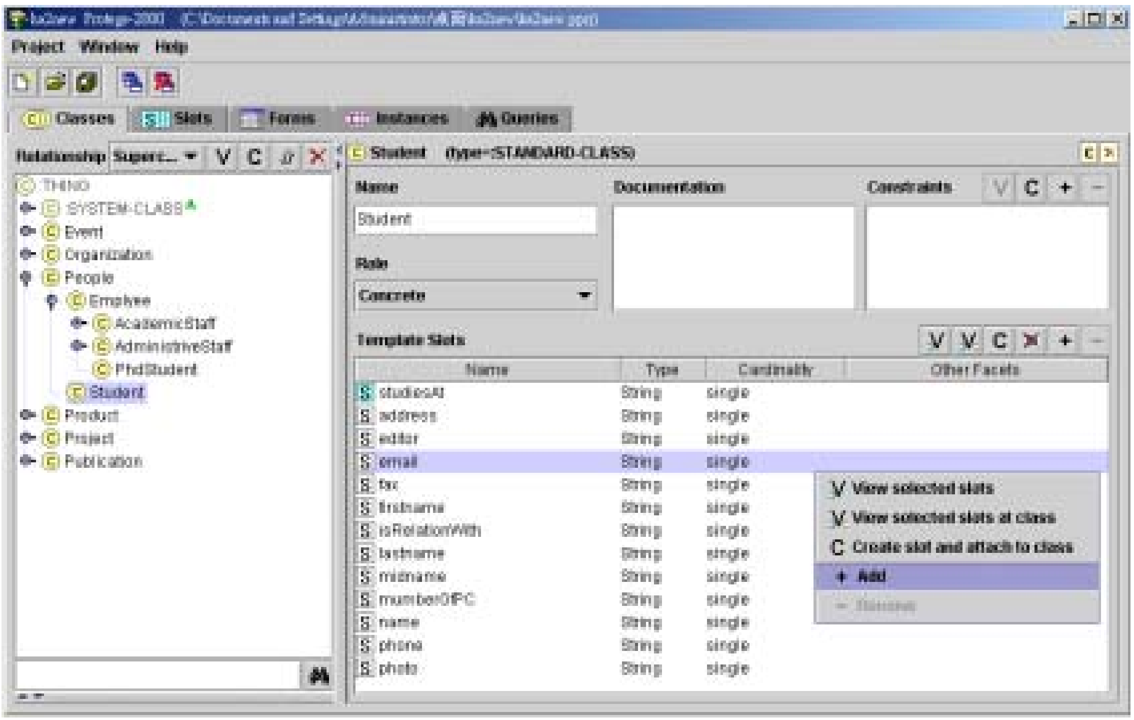
Figure 3-1: Screen shots of the annotation Editor

About Semi-structured Data

We often need to find a regular method for processing the information in a large number of data. With this idea, we design a wrapper program that automatically converts the semi-structured data which store in the relation database into the knowledge warehouse. We can integrate the heterogeneous data into the common formal format. These programs are designed for each specific case. Therefore coding the transformation program is the key point for creating the information from the semi-structured data pools.

About structured Data

Directly producing the structured data for knowledge expression is the fast way to achieve the goal of data exchange. Therefore the new data we should consider to directly make them with formal structure. We can create the structured data with some editors for helping us quickly create these data. For example, we can use protégé 2000 directly create structured information. Protégé 2000 allows us to build a new data with formal knowledge for information processing, as shown in Fig 3-2. If we want to describe the information of time, we just choose the relational class in the editor and write the data in the slot. After the editor produces the structured information, we can store these data in the knowledge warehouse.



Student (type=:STANDARD-CLASS)

Name: Student

Role: Concrete

Documentation

Constraints

Template Slots

Name	Type	Cardinality	Other Facets
studiesAt	String	single	
address	String	single	
editor	String	single	
email	String	single	
fax	String	single	
firstname	String	single	
isRelationWith	String	single	
lastname	String	single	
midname	String	single	
numberOfPC	String	single	
name	String	single	
phone	String	single	
photo	String	single	

Figure 3-2: The structured data editor

3.3 Inference Engine and Repository

We use *flex*, a frame-based system, as the inference engine. flex is a Prolog-based toolkit for us to design an intelligent system. It is an expressive and powerful expert system toolkit which supports frame-based reasoning with inheritance, rule-based programming and data driven procedures fully integrated within a logic programming environment. It contains its own English-like Knowledge Specification Language (KSL). We can describe the data with KSL and write some rules to process the relation of data.

Our repository consists of two kinds of stores: one is directory-based file system and the other is relational database. The file system is used to store the frame instances represented in the language of *flex*, KSL. The file system is tree-structured directory. We nominate a Root directory corresponding to the root of ontology schema. Down the Root directory are subdirectories of the corresponding subclasses in the ontology hierarchy. In each directory is a file that stores the instances of metadata of the corresponding class in the ontology. The file contains instances in KSL. Each time when a metadata instance file in RDF is converted into KSL, the resulting instances are appended in their files.

The file system structure mentioned above is suitable to store instances in KSL and is easy to be accesses by the inference engine of flex. During the course of executing a service program, the inference engine can load the required instances by giving the paths of the files that contains the instances of the classes in question. Upon accepting the concepts user is looking for, the conceptual search program recognizes the associative classes, finds out the paths and then loads the corresponding file according to the path. For example, when we want to find employee, we will to find person class first. We know employee is a subclass of person, so we would search the data about employee under person class. Because the data store in machine with file system architecture, the machine can map to human's concept easily.

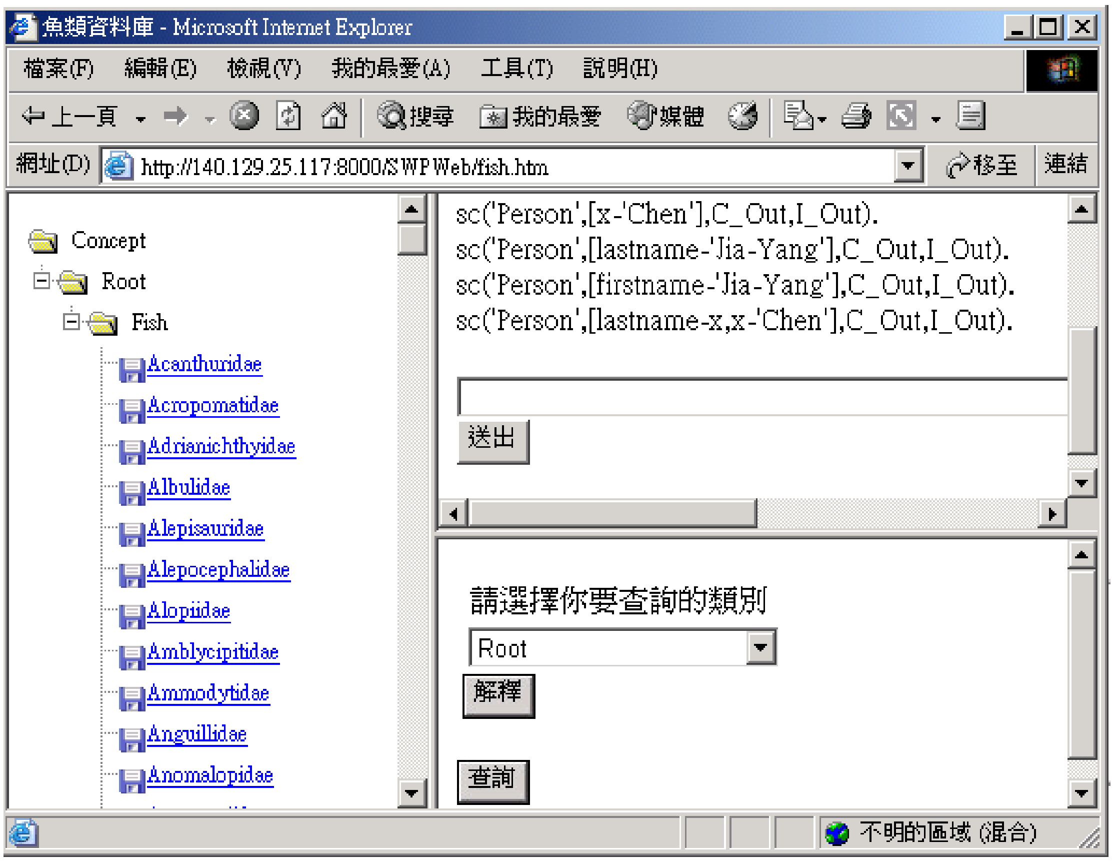
In order to deal with the structures in RDF documents and promote the efficiency of search, we provide a persistent store on relational database for RDF triples. The RDF triple store consists of five tables: `literals`, `namespaces`, `predicate`, `resources` and `triples`. They not only include the information in the tables but show the relation of concept with identified number. The conceptual relations are implicitly stored in those tables. Therefore it is difficult to know the connections between subjects and objects. The relationships can be revealed by appropriate programming. After the program returns the relations corresponding to the original conditions, user can have a chance to know the information.

The `literal` table stores the literal which appear in RDF document. There are many differences namespace in the namespaces table. The `resources` table describes the class and namespace. The `triples` table connects the literal, class and namespace. In order to access the data in the triple store database, we design an internal query language. The internal query language helps to query the data in the relation database and produce the conceptual class and the answers. It is an interface for inference engine to access the real data.

4. Application of the System

We encode the content of web page in a machine-understandable semantics in the form of ontology. And we use our agents, the search and navigation, on the semantic web collection the user requirement. In the fact, the information on the web is too large and changes too quickly for any agent to process the complete knowledge on the whole existence knowledge pool. Even if the agent can process the knowledge, it costs too long time. However, with an open world, an agent may spend an unbound amount of time to find an answer to a query when the answer none exists. We use the local close world in our system to overcome this problem. Therefore, when we decide inserting a knowledge ware of scientific domain, the bound of the knowledge is decided. We believe if the semantic web does not use this approach to solve this problem, they cannot stop to find the answer when the answer is not found on the web. Therefore one of the ability on our system is that we can show the answer in the bounded time.

For scientific information such as fish, animal and plant information, we can use the metadata to describe the relation of the resources. Our system architecture allows us to add semantic content into the knowledge warehouse, to relate this content to its ontology and to provide contextual information for users about the domain. Using this information, the query service can provide more accurate responses than are possible with the search engines available on the Web. We have applied these techniques to the domain of bioscience. According to the web page of fish, we build a prototype of knowledge warehouse about it and provide user to access the knowledge, as shown in Fig 4-1. It provides user to access the data using semantic navigation and conceptual search. For the development of general information such as the time and address of a meeting, the reporter of a seminar, we can annotate with the note label to describe these issues.



```
sc('Person',[x-'Chen'],C_Out,I_Out).
sc('Person',[lastname-'Jia-Yang'],C_Out,I_Out).
sc('Person',[firstname-'Jia-Yang'],C_Out,I_Out).
sc('Person',[lastname-x,x-'Chen'],C_Out,I_Out).
```

Figure 4-1: The fish knowledge access

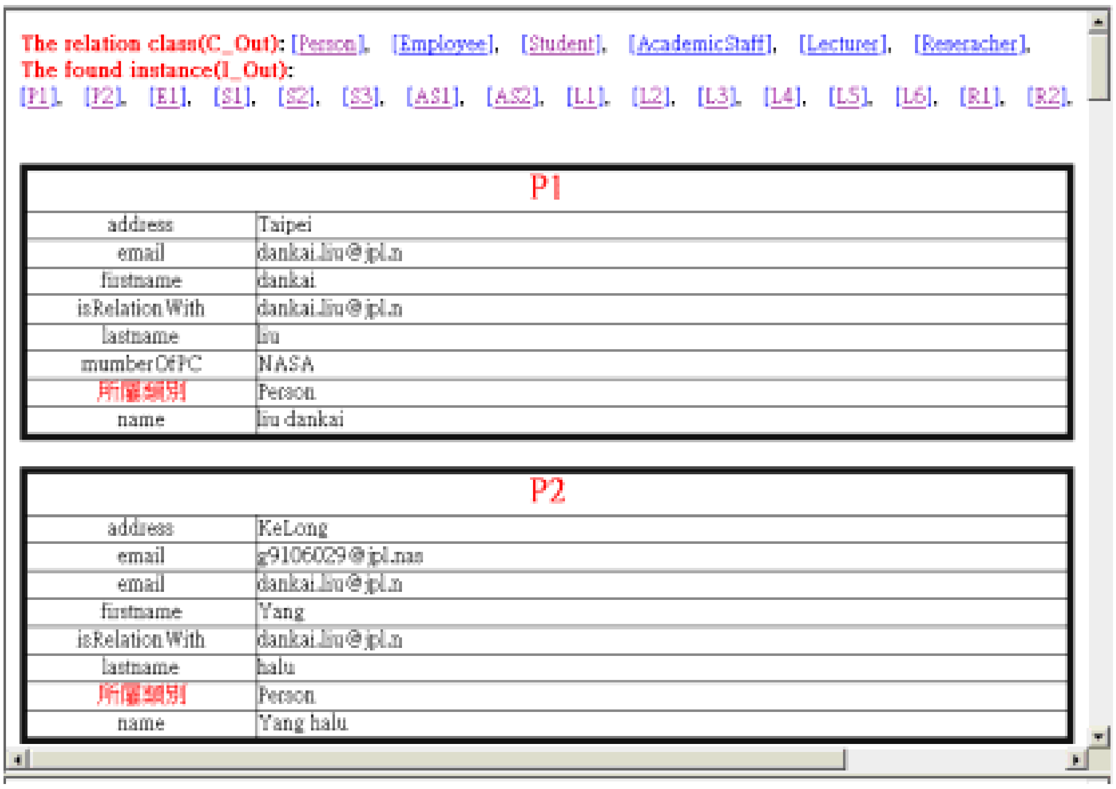
It is also an important message in the scientific domain and we can access the information with more precise query.
We obtain the data from the fish database of Academia Sinica. The web site provides abundant fish information in Taiwan. We use a wrapper program to convert these data in our data store. According to the classification system of the fish web site, we know that there are total 252 families about fish. We use the fish classification of the world to describe the fish information. According to the family of the fish we give fish a rough classification. Therefore we build 252 subclasses under the class of fish, and every class has at last one instance. We describe these data with KSL and store these data in the file system and the RDF triple store.

The fish database of Taiwan contains lots of fish information in Taiwan. They classify the fish by its family, and provide the picture and sound about a fish. We describe the data form the knowledge provider and manage these data using our system. We can navigate all of the family and find the detail information about the fish with linking the remote resource. The user also can use the internal query language directly access the data in the repository, but the complex query language will be the boring thing to the beginner. User can also use the navigation or search interface to find the information about the fish

We provide three methods to search data in the knowledge warehouse, conceptual search, semantic navigation and internal query language. The internal query language is an infrastructure for conceptual search and semantic navigation. In order to access our database, we design an internal query language. We take an example to explain how to use this query language. If we want to find data and we know that the data is in ‘person’ class and it has the predicate called ‘lastname’, we can describe the request as Ex. 5-1.

```
sc('ka2new_Person',[lastname-x],C_Out,I_Out).
```
Ex. 5-1

Ex. 5-1 is a way to query some instances which belong to the class of 'ka2new_Person'. 'ka2new', the prefix of the 'ka2new_Person', means the namespace of the 'Person' class. It is a way to discriminate the person in the domain of 'ka2new' form other person in another domain. The low case of x means that we do not care the value in the field. Therefore the meaning of the foregoing query is to find some classes which are related to 'ka2new_Person' and have a property of 'lastname'. If the search engine can find the instance of this query, the answer exists in I_Out. The result of internal query language is shown in Fig 4-2.



The relation class(C_Out): [Person], [Employee], [Student], [AcademicStaff], [Lecturer], [Reseracher],
The found instance(I_Out):
[P1], [P2], [E1], [S1], [S2], [S3], [AS1], [AS2], [L1], [L2], [L3], [L4], [L5], [L6], [R1], [R2],

P1	
address	Taipei
email	dankai.liu@jpl.n
firstname	dankai
isRelationWith	dankai.liu@jpl.n
lastname	liu
mumberOfPC	NASA
所屬類別	Person
name	liu dankai

P2	
address	KeLong
email	g9106029@jpl.nas
email	dankai.liu@jpl.n
firstname	Yang
isRelationWith	dankai.liu@jpl.n
lastname	halu
所屬類別	Person
name	Yang halu

Figure 4-2: The result of internal query language

For conceptual search, we design a graph interface for user to input their data. In this method of search, we not only use property and value to find the answer but we are according to the class of concept to find the results. The search engine finds all relational concept with certain class and then put the answer to the user. For example, you want to find a person but you do not remember the name of this person. You can search the class of person and use the name of predicate to find the relation data.The semantic navigation helps people to find the resource under some classes. The users can navigate the concept on the repository. The label of tree list shows the concept’s name. We know the relation of concept with the ontology. According the description of ontology, the tree’s relation can be built. So the navigation tree describes the relationship between class and class in the ontology. The semantic navigation will accept the user’s request and send this request to the conceptual search engine; then product the answer. Each of queries is the query a class which you choose. It will show all the instances under this class.

Conclusions

We have built an ontology-based community portal for scientific application to test the functionality of the system we developed previously. The metadata for describing information with various degrees of structures are created using RDF. The created results are loaded into the knowledge warehouse of the having ontology schema in it. We have tested the conceptual search and semantic navigation services. They can provide precise way of getting information of interest. We have made a comparison of performance by using Prolog and frame-based system. The result shows that the latter achieve better result. In the future we will employ knowledge engineering methodology, CommonKADS, as the basis to develop knowledge management system using our system.

Acknowledgements

This research was supported by the Taiwan National Science Council under Contract No. NSC92-2422-H-036-322.

References

[1] Brian McBride, *Resource Description Framework (RDF): Concepts and Abstract Syntax*, W3C Recommendation, http://www.w3.org/TR/rdf-concepts/, 10 February 2004.

[2] Steve Pepper and Graham Moore, *XML Topic Maps (XTM) 1.0 Topic Maps. Org Specification*, http://www.topicmaps.org/xtm/index.html.

[3] DAML Ontology Library, http://www.daml.org/ontologies/

[4] York Sure, *KA2-Knowledge Acquisition Community Ontology*, http://ontobroker.semanticweb.org/ontos/ka2.html, 2000.

[5] R. Scott Cost et al. *ITTALKS: A Case Study in the Semantic Web and DAML, IEEE Intelligent Systems*, Vol. 17, No. 1,pp. 40-47, January/February 2002.

[6] An annotation tool, Protégé-2000, http://protege.stanford.edu/.

[7] Clive Spenser, Flex Tutorial, http://www.lpa.co.uk/ind pro.htm, 2002.

[8] Natalya F. Noy and Deborah L. McGuinness, *Ontology Development 101: A Guide to Creating Your First Ontology,* http://protege.stanford.edu/publications/ontology_development/ontology101.html

[9] KAON, http://km.aifb.uni-karlsruhe.de/kaon2.

[10] Chimara, Knowledge Systems Laboratory, Stanford University, http://www.ksl.stanford.edu/software/chimaera/

[11] OntoEdit, http://www.ontoprise.de/products/ontoedit.

[12] Institute AIFB, *TUTORIAL Development and Applications of Ontologies*, September 2000.

[13] WordNet, http://www.cogsci.princeton.edu/wn/.

[14] Ching-Long Yeh, Development of an ontology-based portal for digital archive services, International Conference on Digital Archive Technologies (ICDAT2002), Academia Sinica, Nankang, Taipei, Taiwan, 2002.