

Ontology-based Access to Distributed Statistical Databases

Yaxin Bi¹, David Bell¹, Joanne Lamb² and Kieran Greer³

¹School of Computer Science
The Queen's University of Belfast
Belfast BT7 1NN, UK

³Faculty of Informatics,
University of Ulster, Newtownabbey,
Co. Antrim, BT37 0QB, UK

²CES, University of Edinburgh
St John's Land, Holyrood Road
Edinburgh, EH8 8AQ, UK
{y.bi, da.bell}@qub.ac.uk
krc.greer@ulster.ac.uk
J.M.Lamb@ed.ac.uk

Abstract. In this paper we describe some commonly-required functionality for constructing, visualizing and manipulating ontologies. These make extensive use of XML and DOM technologies. The functions developed are based on practical applications needing access to distributed statistical databases and intelligent content management. They fulfill the needs identified as new features for next generation ontology development and application. These system features have been implemented in Java and JAXP.

1 Introduction

In scenarios involving interoperability between distributed database systems, semantic heterogeneity is a significant problem and it will continue to be so in the future [1]. A variety of solutions to this problem have been proposed in the past decade, including the mediation approach, schema matching, and ontology-based approaches [2]. Ontologies are useful because they form a basis for integrating separate databases through the identification of semantic connections or constraints between the data or pieces of information. This paper addresses the issues of ontology structure, construction and manipulation.

The concept of “ontology” is well-known in knowledge engineering and it was originally developed for knowledge-based systems. The recognition of its usefulness has led to its applications to data integration [3], an intelligent content management [4], and the Semantic Web [5]. A solution to the problem of semantic heterogeneity is to formally specify the meaning of the terminology of each system and to define a correspondence between system terminologies, in which the terminologies will be specified using *ontologies* and the correspondence between them will be defined by *ontology mappings* [1]. Ontologies can therefore take different forms with different purposes and can be used in different phases of different applications. For example, we may have *source* and *receiver ontologies*, which define the terminology used by specific data sources and receivers, respectively. The latter makes use of *query ontologies*, which define the terminology for a range of users. Another type is *shared ontologies*, which are used as the reference terminology among different systems.

Considerable research exists in developing ontologies and their application. As described in [5], the building of ontologies was done in an *ad hoc* fashion in the past,

but more recently there have been some proposals for guiding the ontology development process. For instance, [6] gives formal guidelines for constructing a consistent and reusable ontology and a few Web-based initiatives are aimed at addressing the scale and scope of development of ontologies for the Semantic Web[7].

In contrast to these methodologies, which mostly confine their attention to the ontology itself, in our work we focus on the application-driven development and exploitation of ontologies, in particular focusing on the official statistics domain and on intelligent content management. These involve various technical aspects of ontology construction, visualization, mapping, exploitation in a query scenario. We describe our practical experiences obtained from the MISSION [3] and ICONS [4] projects. The work reported here is related to our other work: user interaction styles with distributed database systems [8] and a visual querying paradigm [9].

2 Ontologies

An ontology is defined as a shared formal conceptualization of a particular domain [10]. It can be used to specify what concepts represent and how they are related. Practically, ontologies are characterised into two types of *simple* ontologies and *sophisticated* ontologies. We concentrate on simple ontologies. A *simple* ontology should hold mandatory properties such as [11]:

- Finite controlled (extensible) terms (values)
- Unambiguous interpretation of concepts and term relationships
- Explicit hierarchical sub-concept relationships between concepts

Currently, some simple ontologies are available in many forms – some exist as freeware on the web, such as UN Classifications Registry a classification scheme for universal products and services [12]. A hierarchical structure may contain many levels in order to precisely represent the relationships between concepts, i.e. a concept may be broken down into a group of sub-concepts, each of which in turn may be divided into smaller ones, and so on. Notice that such a decomposition process is recursive, and for convenience of discussion, we assume an ontology is composed of three levels as shown in Fig. 1 (a). i.e. Ontology \rightarrow Concept \rightarrow Value. Such a hierarchical structure provides a natural way to visualize an ontology as a tree, illustrated in Fig. 1 (b).

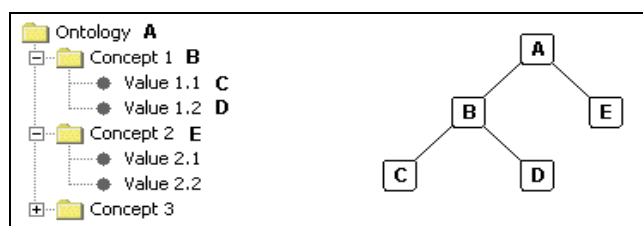


Fig. 1. An abstract structure for ontologies (left: a, right: b)

Ontologies can be represented in various ways, such as description logic and RDF [7]. To address issues in the content representation of databases, ontology construction and visualization, we employ XML to represent ontologies, because it deals well with

hierarchies, and covers various types of data. In particular, it is well suited to the generation of hierarchies as required by different kinds of operations involved in data integration and the Semantic Web services.

Our ontologies are simple and each consists of a set of variables (concepts or nomenclatures). These can have hierarchical structure, and each variable may comprise a set of category values.

3 Developing ontologies

The problem of developing ontologies has been well studied. A comparative analysis of relevant methodologies can be found in [5]. One of the major conclusions of that study was that the best approach to take in developing an ontology should be determined by the eventual purpose of the application.

In our work, we develop a *content-driven* approach for developing ontologies. This approach defines a template with XML syntax, being composed of two major parts: keywords and hierarchical structures. The template plays a vital role in mapping data with a flat structure into a hierarchical structure. To store metadata, we make use of the flexibility of a 3-ary relational schema (for data dictionary). The advantage of such a schema and its application in e-commerce has been described in [13]. We describe how to use a template to construct an ontology below.

3.1 Metadata and data dictionary

It is frequently the case in scientific data archives, for example, that raw (micro) data is of value encoded form, which are accompanied with a considerable amount of metadata for interpretation (see Table 1). The metadata can be regarded as a content interpretation of databases. For example, the attribute *Land* – physical name – occurring in a relation has a logical name *Country* and its domain consists of a set of values {Scotland, France, Sweden, Netherlands, Ireland}. All of the attributes and values constitute granularities of ontologies.

Attribute	Logical name	Type	Label
Land	Country	Categorical (geographical)	{Scotland, France, Sweden, Netherlands, Ireland}
Sex	Gender	Categorical	{male, female, Not answered}

Table 1. A piece of metadata

To store metadata, we develop a 3-ary schema, consisting of attributes *Role*, *Attr* and *Ext*. In general, *Attr* and *Ext* represent mnemonic and logical names of attributes and domain values, respectively, and *Role* represents equivalence relations of *Attr* and *Ext* in terms of keywords. Fig. 2 shows a piece of metadata held in a 3-ary relation (table), called a data dictionary. The domain of attribute *Role* comprise *frame*, *geo* and *map*, *label* (*categorical* is treated as a constant value), representing different relations between pairs of *Attr* and *Ext*. For example, *map* means that *Land* and *Country* are semantically equivalent. To extract content from a data dictionary, we specify the content of the data dictionary by using the keywords and structure to be imposed on the content in a template (see Section 3.2).

Role	Info	Ext
frame	Catewe	*
geo	Ireland	Country
map	Land	Country
categorical	0	*
label	Scotland	Country
label	France	Country
...		

Fig. 2. A fragment of data dictionary

3.2 Ontology construction using content-driven extraction

Given a hierarchical structure of ontologies and a data dictionary, the idea of content-driven extraction is to retrieve data from a data dictionary using keywords specified in a template and fill them into the structure defined in the template. This structure strictly complies with the structure of a given ontology. A template is defined with two elements of *element* and *attribute*, and it conforms to XML syntax.

In a template, *element* is used to define tag names via its attribute, and *attribute* is for defining attributes of elements with either the keywords or constant values. If a keyword is used with the parameter *column*, that means that attribute values of elements and values of attributes will be extracted from the corresponding column in a data dictionary, otherwise these values are constant values, such as *Catewe* and *categorical* at line 2 and 4 in Fig. 3.

The nested relation between *element* and *attribute* defines hierarchical relations, which reflect ontology structures to be formed, i.e. the *attribute* of an element is always nested at one level lower than that element. Fig. 3 illustrates a fragment of a template. Three tag names *Ontology*, *CONCEPT* and *VALUE* are specified by *element* in line 1, 3 and 7, respectively, and the nested relations between tags are restricted by *attributes*, and the resulting hierarchical structure is the same as one described in Fig. 1(a).

```

1. <element name="Ontology">
2.   <attribute name="name">Catewe</attribute>
3.   <element name="CONCEPT">
4.     <attribute name="vartype">categorical</attribute>
5.     <attribute pname="mnemonic" column="2">map</attribute>
6.     <attribute lname="name" column="3">map</attribute>
7.     <element name="VALUE">
8.       <attribute name="value" column="2">label</attribute >
...

```

Fig. 3. A fragment of template

With the template and data dictionary, we now briefly look at how ontologies can be constructed. The construction algorithm has been implemented in MISSION as a functional module called XML Extractor to extract partial information from a data dictionary to generate ontology as shown in Fig. 4 (a). The key idea of XMLExtractor is to take as input a template, remotely connect to a data dictionary, and retrieve a data dictionary using the keywords specified in the template, and then recursively generate XML fragments based on the structure specified in the template. A fragment of the ontology generated from the data dictionary in Fig. 2 is shown in Fig. 5.

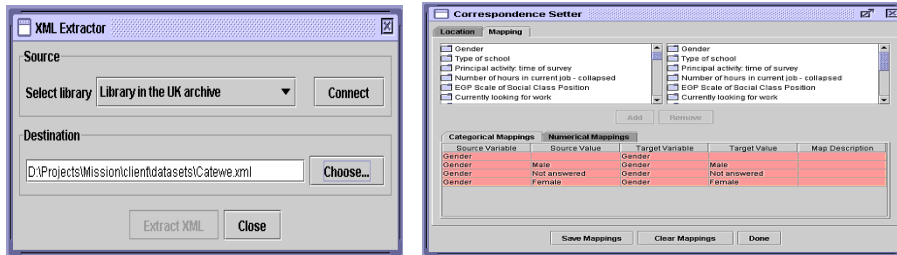


Fig. 4. Ontology construction (left: a) and mapping (right: b)

```

<Ontology name="Catewe">
<CONCEPT datatype = "" vartype = "geographical" mnemonic = "LAND" name = "Country">
  <VALUE value="Ireland"/>
  <VALUE value="Netherlands"/>
  ...

```

Fig. 5. A XML fragment of the ontology Catewe

The advantage of this approach is twofold: 1) it provides a means of converting from data with a flat structure to a hierarchical structure and 2) it has a generic character which is capable of coping with different structures defined in the templates, without having to change the extraction algorithm. Our approach is different from most methods which are built on the mapping between database schemas and XML documents [14]. The underlying difference lies in that our approach does not utilize the structure of database schema in generating XML documents. Instead, it is based on the content reflected in the database and structure reflected in ontologies, as defined in the templates.

4 Ontology visualization

As mentioned previously, ontologies can be used for different purposes. To use an ontology as a user interface component, the visualization of ontologies is important. In an ontology-based query scenario, a *query ontology* can be treated as a view of data sources – a content representation. As indicated in [15], one of the difficult problems related to such a query model is how to visualize the ontologies as query views in terms of their structure and content. The issue associated with ontology visualization is also pointed in [6], i.e. in the future, ontology visualization will be seen as an important facility in applying ontologies.

In this work, we describe an approach to visualizing ontologies by means of the DOM (Document Object Model) [16]. DOM is a platform- and language-neutral application programming interface. It allows programs to dynamically access and manipulate the content and structure of XML documents. DOM is used to define the tree-like structure of documents and provide a standard set of objects for representing XML documents, along with a standard interface for accessing and manipulating them. Therefore the use of DOM is an ideal way for managing and visualizing XML documents.

A DOM instance can be defined as a tree model and implemented as a tree. According to the DOM, a XML document is a collection of nodes with hierarchical

structure. Nodes are of 12 types, but there are only three types that are important to our discussion: element node, text node and attribute node. Fig. 6 (a) shows how an XML fragment of the ontology in Fig. 4 is displayed using DOM. In the figure the element nodes may have element, attribute and text nodes. For example Ontology has the element node CONCEPT, attribute node ATTRS and #text, but the attribute and text nodes only carry texts, which are terminals. It is possible to visualize element nodes as proper tree nodes, acting as an accurate depiction of the contents of distributed data, rather than being an accurate description of the data structure as represented by the DOM., as illustrated in Fig. 6 (b). Instead of filtering attribute and text nodes out, there is a need to encapsulate element, attribute and text nodes into objects since they are important constraints to respective elements. We have developed an algorithm to meet this requirement.

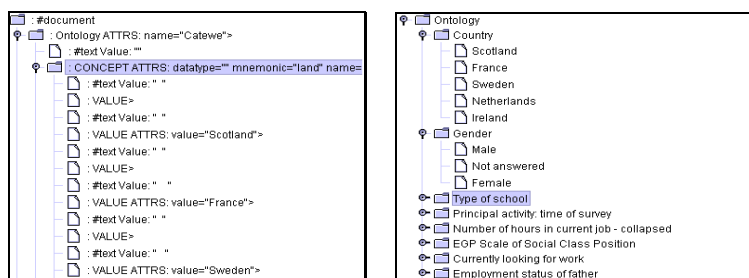


Fig. 6. Visualizing ontologies (left (a): a DOM tree; right (b): an ontology tree)

5 Mappings between schemas and ontologies

In a relational database system, a schema defines a relational table. An ontology, on the other hand, defines the meanings of the terms used in the schema, and hierarchical relationships between them as described in Fig. 1. It is not always feasible to directly map an ontology to a schema because of this hierarchical structure. We confine our attention to the simple case where a schema corresponds only to an ontology, i.e. one-to-one mapping. For example, we define a schema and construct an ontology based on Table 1. It is straightforward to establish the mapping between the schema and the ontology through mapping the attributes in the schema to the meanings of the corresponding attributes within the ontology, e.g. *Land* mapping *Country*.

Defining mappings between ontologies is another aspect in the integration of diverse data sources. An ontology normally includes more semantics than a schema, the existence of such semantics provides an effective means for coping with semantic heterogeneity reflected in data sources and makes it possible to implement an automated approach to creating mappings between ontologies. In the MISSION project, we have implemented an ontology mapping function, called *Correspondence Setter* for data providers in order to define mapping rules between source ontologies. A screenshot of this function module is shown in Fig. 4(b).

In this work, we propose two ways to define mappings to indicate how terms in one ontology correspond to terms in another [17].

- *peer to peer mapping* – mapping rules are defined for each pair of ontologies which directly map from one ontology into another.
- *Mapping via a standard reference classification (taxonomy)* – mapping rules are defined for each source ontology into a shared reference classification to be derived from public classification repositories as described in [12].

Both mapping rules can be defined manually using the function of the *Correspondence Setter* as illustrated in Fig. 4 (b). For example, peer-to-peer mappings are possible of the corresponding terms in the source ontologies Catewe Ireland and Catewe Scotland via this function. Notice that mappings occur at the levels of both variables such as *Gender* \leftrightarrow *Gender* and the value set such as Male \leftrightarrow Male. Resulting mappings will be stored in relations called *correspondence tables* for use in query processing. In addition to this, we have implemented a function to allow ontologies to be merged together into a master ontology based on the amount of semantic overlap between ontologies. In fact, mappings can occur at multi-levels, such as ontology, variable, classification and value, etc. The support for complicated and dynamical mappings between ontologies is currently under development.

6 Queries against ontologies

Queries are composed using the terms defined in the ontologies, and posed against the ontologies, instead of schemas or views, as in an alternative to schema and the views-based systems. To answer them, a relationship is defined between the ontology and the schema. Processing these efficiently converts a query expression derived from the ontology into the internal schema-based expression of the query.

Any query composed of the terms defined in the ontology must be converted to an equivalent executable query against the schema. In a system, queries are visually specified over query ontologies and expressed in a high-level query language. These high-level user queries are translated into queries over the definitions at the ontology mapping layer, and ontology terms are replaced by schema attributes. More detail is described in our internal paper [9].

7 Summary

Ontologies play a key role in coping with semantic heterogeneity occurring in a variety of applications. Some development environments exist for building ontologies, such as Ontolingua [6] and OntoEdit [7], and there exist some repositories of simple ontologies as well, but application-oriented methods and techniques in relation to specific applications are still under development.

Experience with the ontology development and exploitation in connection with official statistics and general intelligent content management, and an exhaustive analysis of current ontology research, have led us to develop functionality such as content-driven construction, visualization based on DOM, and ontology mappings, all of which are regarded as essential features of next-generation ontology construction and applications. Although these have been developed specifically for statistical

databases in the MISSION project, an attempt is being made to tailor these features to general applications, such as those using the ICONS.

Acknowledgement

The work is partially supported by the MISSION project (IST 1999-10655) and partially supported by the ICONS project (IST-2001-32429), which are funded by the European Framework V.

References

1. Zhan, C., Jones, D., and O'Brien, P. Semantic B2B integration: issues in ontology-based approaches. ACM SIGMOD Record, Vol.31 (1), (2002) 43-48.
2. Mena, E., Illarramendi, A., Kashyap, P., Sheth, A. P.: OBSERVER: An Approach for Query Processing in Global Information Systems Based on Interoperation Across Pre-Existing Ontologies. Distributed and Parallel Databases 8(2): (2000) 223-271.
3. Specification of the MISSION system (Multi-Agent Integration Of Shared Statistical Information Over The (Inter)Net) (Deliverable 6), 2001.
4. Bell, D, Bi, Y., *et al.* Analysis and selection of the ICONS (Intelligent Content Management System) project research base (Deliverable 6), June 2002.
5. Staab, S. Methodology for Development and Employment of Ontology based Knowledge Management Applications in Semantic Web, Database Management and Information Systems: Overview of the Special Issue, SIGMOD Record, No.4, December 2002.
6. Fikes, R., Farquhar, A. Large-Scale Repositories of Highly Expressive Reusable Knowledge; IEEE Intelligent Systems, Vol. 14, No. 2, March/April 1999.
7. D. Fensel. On-To-Knowledge: Semantic Web Enabled Knowledge Management, IEEE Computer, 35(11), 2002.
8. Bi,Y., Bell, D. and Lamb, J. Aggregate Table-driven Querying Via Navigation Ontologies in Distributed Statistical Databases (to appear in BNCOD 2003).
9. Bi,Y., Bell, D. and Lamb, J., Greer, K. Visual Query with Ontologies for Distributed Databases (internal paper).
10. Gruber,T.: A translation Approach to Portable Ontology Specifications. Knowledge Acquisition. Vol. 5(2), (1993)199-220.
11. McGuinness, D., L. Ontologies Come of Age. In Fensel, D., Hendler, J., Lieberman, H. and Wahlster, W., (eds). Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. MIT Press, 2003.
12. UN Classifications Registry, esa.un.org/unsd/cr/registry, (April 2002).
13. Rakesh Agrawal, Amit Somani, Yirong Xu: "Storage and Querying of E-Commerce data", 27th Int'l Conference on Very Large Data Bases Rome, September 2001.
14. Shanmugasundaram, J., Shekita, E., Barr, R., Carey, M., Lindsay, B., Pirahesh, H. and Reinwald, B. Efficiently publishing relational data as XML documents. VLDB Journal: Very Large Data Bases, vol (10) 2-3, (2001)133-154.
15. Wache, H. V ogele, T. Visser, U. Stuckenschmidt, H. Schuster, G. Neumann, H., H ubner, S.: Ontology-based integration of information – a survey of existing approaches. In Stuckenschmidt, H. (ed.): IJCAI-01 Workshop: Ontologies and Information Sharing, (2001) 108-117.
16. Document Object Model, <http://www.w3.org/DOM/> (October 2002).
17. McClean, S., Páircéir, R., Scotney, B., Greer, K. A Negotiation Agent for Distributed Heterogeneous Statistical Databases in SSDBM (2002) 207-217.