Health Ontology Generator: Design And Implementation

Yip Chi Kiong[1], Sellappan Palaniappan[2], Nor Adnan Yahaya[3]
Department of Information Technology,
Malaysia University of Science and Technology,
Kelana Square, Kelana Jaya, 47301 Petaling Jaya, Selangor, Malaysia

Summary

This paper presents the design and implementation of a Health Ontology Generator (HOG) using a health database such as Microsoft Access or SQL Server. The development of the ontology generator involves building methods for creating and reading the ontology. This research performs both these tasks. In generating the ontology, database tables are treated as classes, fields as functional properties, and records as instances. The ontology generated can be read using third-party software such as Microsoft Word, Excel and Internet Explorer. HOG is implemented using C#.NET on the Windows platform.

Key words:
Ontology encoding and generation, database schema, ontology viewing, ontology information extraction and integration

1. Introduction

Most of the existing information systems are based on databases developed over years. These databases are populated from transaction records. There are software available which can create ontology by defining individually the classes, properties and even the instances. This process is tedious when it becomes necessary to create an ontology based on databases. It is more useful to be able to extract the schema of the databases and add useful semantics to generate ontology automatically. The ability to perform this task will be useful in database-ontology integration, as well as ontology-ontology integration.

1.1 Purpose

This paper discusses the development of a system to extract a database schema and create an OWL ontology file using C# in the Visual Studio environment. It also discusses the debugging and displaying of an OWL file using various resources, such as Internet Explorer, Microsoft Excel, and Microsoft Word.

With the ontology generated, it is necessary to be able to read it as a stream of data to extract information on the classes contained in it, and the properties and instances that are included.

Manuscript received February 5, 2009
Manuscript revised February 20, 2009

1.2 Review of Resources

We have searched the Web for resources to perform the task of extracting the schema of a database to create an ontology file. Protégé appears to be promising, using the DataMaster plug-in developed in the BioSTORM Project at the Stanford University School of Medicine. This was an upgrade from the DataGenie plug-in, which could not extract the schema alone, necessitating the import of the schema as well as the data [1]. This is basically a Java-based ontology editor, where one can generate ontology by extracting the database schema through an ODBC-JDBC provider. Although these programs exist, there is no documentation to describe the process of generating ontology from a database in sufficient detail to build an ontology generator.

1.3 Problem Statement

The generation of ontology involves exploring several concepts that deals with the details of file generation. These include

- How do we extract the schema of a database using OLEDB in the Visual C# environment?
- How do we encode an ontology?
- What constitutes the structure of an ontology file?
- What semantics should be included in the ontology?
- How do we verify the correctness and usefulness of the ontology generated?
- How do we extract classes, properties and instances from the ontology?

In order to answer these questions, we analyzed in depth how Protégé performs the job of creating ontology. We imported several types of databases with a known schema and studied the output generated. We found that the namespaces and the semantics generated are not within our control to specify. It was found necessary to edit the file generated using a text editor such as WordPad in order to edit the file, which can be very tedious and error-prone.

This makes it necessary to develop our own system to perform the generation of the ontology. The ability to read ontology is necessary to extract information from a given ontology.

1.4 Recent Development in Ontology

The term ontology was introduced in philosophy in the nineteenth century and have been widely used in computer science, predominantly in the area of natural language processing and knowledge representation. More recently, ontologies have become the central focus of the Semantic Web initiative, giving rise to proposals on ontology description languages and associated technologies. The most frequently quoted definition of ontology in the Semantic Web literature is the one by Gruber [2] which refers an ontology as a formal, explicit specification of a shared conceptualization. In simple terms, we can view an ontology as providing a vocabulary for the basic terms and relations used to describe certain domain or topic of interest. It consists of specific vocabulary used to describe a particular reality, together with a set of explicit assumptions regarding the intended meaning of the vocabulary.

The vision of Semantic Web was first articulated in 2000 by Tim Berners-Lee, who is the inventor of the current World Wide Web during his XML 2000 address (http://www.w3.org/2000/talks/1206-xml2k-tbl/slide1-0.html) where he envisaged the architecture of this future Web as consisting of several layers, with ontology as one of them. The Semantic Web idea is further elaborated in [3] where the Semantic Web is described as “an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in co-operation.” To date, many proposals have been made to develop the so-called Semantic Web languages to represent the various aspects associated with the future Web. In an interview with Tim Berners-Lee [4], two development were considered to be most significant ; the Resource Description Framework (RDF) [5]for representing metadata and Web Ontology Language (OWL) [6] for representing ontology.

RDF allows metadata to be represented in the form of <subject, property, object> triples using XML syntax. Although it is particularly intended for representing metadata about Web resources, it can also be used to represent information about objects that can be identified on the Web. RDF is also regarded as a lightweight ontology language where its lack of expressiveness was partly eased with the introduction of RDF Schema (RDF-S) [7] through additions of terms for defining application-specific classes and properties. OWL extends the RDF/RDF Schema vocabulary further with richer semantics to allow for descriptions of classes, properties, and relations among conceptual objects in a way that facilitates machine interpretability of Web content. Like RDF and RDF-S, OWL itself is defined as a vocabulary where an ontology described in OWL essentially is a collection of RDF triples using such a vocabulary [8].

There are three dialects of OWL: OWL Lite, OWL DL and OWL Full.. OWL-Lite is the syntactically simplest sub-language. OWL-DL is much more expressive than OWL-Lite and is based on Description Logics (hence the suffix DL). Description Logics are a decidable fragment of First Order Logic and are therefore amenable to automated reasoning. It is therefore possible to automatically compute the classification hierarchy and check for inconsistencies in an ontology that conforms to OWL-DL. OWL-Full is the most expressive OWL sub-language. It is intended to be used in situations where very high expressiveness is more important than being able to guarantee the decidability or computational completeness of the language. It is therefore not possible to perform automated reasoning on OWL-Full ontologies [9].

2. Design of Ontology Generator

The system we have built is called Health Ontology Generator (HOG). It is built to conform to the needs of the healthcare domain (specifically healthcare records). The prototype is built as a Windows Application. This is easier to install on another notebook for purposes of demonstration than designing it as Web Service and Web Client. However, the coding is done in such a manner that it can be readily to converted into Web Service. Each module transfers data using DataSets and passes data as if it is Web Service. This is a more complex Windows Application but convertibility to Web Service has future benefits. The final design will use Web Services and may even include Windows Communication Framework version.

2.1 Algorithm to Extract Schema from Database

We have chosen to develop the system using C#, making rapid development smoother. The first step involves selecting the type of database. We started with Microsoft Access and SQL Server. The connection string is created to connect to the database. For Microsoft Access databases, we use the `Microsoft.Jet.OLEDB.4.0` provider, for SQL Server, we use the OLEDB provider. HOG’s initial interface is shown in Figure 5 in Section 3. Currently the system supports only Microsoft Access and SQL Server. Support for MySQL will be added later.

HOG uses the `schemaTable` method to query the tables in a database and returns the result as a DataSet. After the table names are obtained, it extracts the column names and their data types and puts them into another DataSet. Finally, if the user chooses, the row data is extracted into a third DataSet.

2.2 Method for Encoding Ontology

The ontology generation is an automatic process which encodes and stores the ontology physically as an RDF file that includes declarations of classes, properties and instances. In addition, the ontology also includes the semantics that describe the meaning of the data included in it. Typically, the file is given the file extension owl.

The first part of the encoding process of ontology is the generation of the header. The body of the ontology includes the classes, the properties and the instances. The final part is the trailer. Figure 1 show these stages diagrammatically.

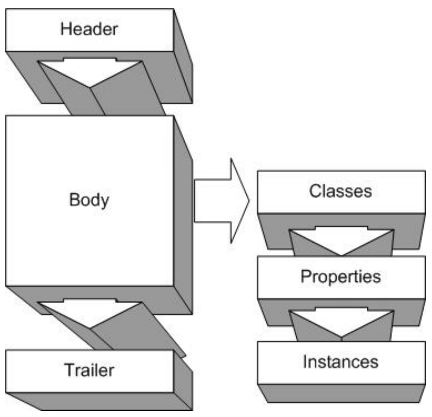


Figure 1: Stages in the encoding of ontology

2.3 Encoding the Header

The header specifies the RDF start tag (with namespace attributes) and the ontology element. It starts with the version information of the XML encoding. This is followed by some standard namespaces, which includes

- XML schema (for data types)
- RDF
- RDFS
- OWL

Each of these standard namespaces is declared using their usual URIs. For example, XMLS is declared as

`xmlns:xsd="http://www.w3.org/2001/XMLSchema#".` The ontology's own namespace is declared as `xmlns: db="http://zhiq.tripod.com/db_table_classes?DSNtype=Access:dbHealth_1#",` which is a reference to the database to link to the ontology. Finally, the ontology element is declared simply as `<owl:Ontology rdf:about=""/>.`

Figure 10 (in Section 4) shows the header as viewed through Internet Explorer.

2.4 Encoding the Body

In the ontology, tables are converted to classes. This is done by constructing the RDF statement as an OWL class. Figure 11 (in Section 4) shows the encoding of the class. Field names (or column names) in the table are converted to functional attributes. Figure 12 shows the encoding of the field names. In addition, other functional attributes are added, which describe the semantics of the ontology. These include

```
#hasFKName
#isBridgeTable
#hasLocTableClass
#hasLocalField
#hasRefTableClass
#hasLocFieldProperty
#hasReferenceField
#hasOrigColumnName
#hasRefFieldProperty
#hasReferenceTable
```

The rows of each table are converted into instances, beginning the first instance in the form of `instance_1`. The annotation properties, such as `#hasForeignKeys` are then added.

2.5 Encoding the Trailer

The trailer consists of the closing RDF tag and information about the creator of the ontology.
`</rdf:RDF><!—creator -->`

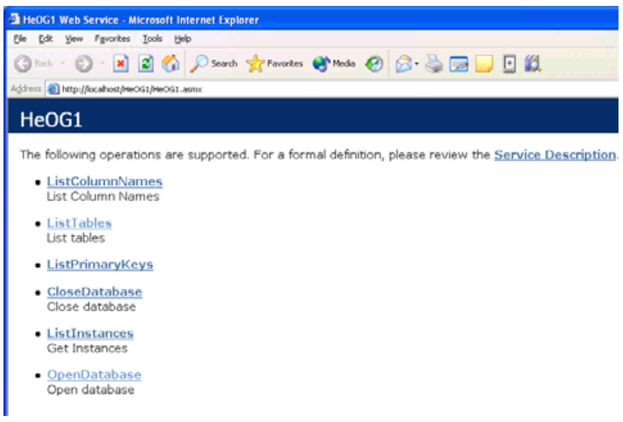
2.6 Web Service Ready Functions

The following functions are implemented as Web Service ready functions:

a. Connect to database
b. Extract Tables from database
c. Extract Fields from each table
d. Extract Rows from each table
e. Extract Foreign Keys from each table

Figure 2 shows some of the Web Services available. These functions use the respective provider's features to extract required data as listed above. They are implemented by using DataSets to pass data from the database tier to the client tier in the three-tier architecture. When we implement the system as Web Service at a later stage of development, these functions will be implemented at the server. The User tier will call these functions either from a Windows or Web

Application. A Web Application will make the layer platform independent.



HeOG1 Web Service - Microsoft Internet Explorer

http://localhost/HeOG1/HeOG1.asmx

HeOG1

The following operations are supported. For a formal definition, please review the Service Description.

- ListColumnNames
  List Column Names
- ListTables
  List tables
- ListPrimaryKeys
- CloseDatabase
  Close database
- ListInstances
  Get Instances
- OpenDatabase
  Open database

Figure 2: Some of the services available as Web Services

2.7 Web Service Architecture of HOG

HOG is implemented as a three-tier architecture. The data tier stores and retrieves data from the database. These functions are processed by the database management system, and are transparent to the system. The information is then passed to the logic tier. Here, the application, in this case the Web Service module itself coordinates and processes commands that come from the user. Finally, we have the top-most presentation layer, which is the user interface. Its function is to translate and interpret tasks from the user. Figure 3 shows these three layers in perspective.

2.8 Architecture of Ontology Generator

How does ontology relate to the user interface and the underlying database? Figure 4 illustrates one way of looking at the links between these resources. The ontology is the theory of what exists, and is expressed in a language such as OWL. The database (which stores the facts) is defined and accessed using this language. The information requirements are the specification of what information we wish to keep. The human-computer interface (HCI) and the computer-computer interface (CCI, which is the reason for the Semantics) are applications that view, interpret, modify or request information/data from the ontology. Sometimes, the ontology is stored in the database. Ontology tools are used to create and modify the ontology. This is an area which needs more research to discover the implementation of these links.

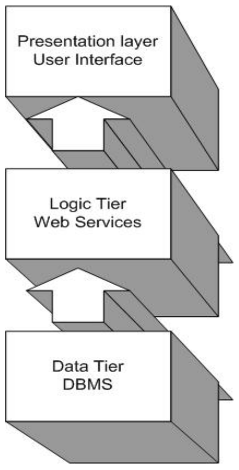


Figure 3: Three-tier Web Services model of HOG

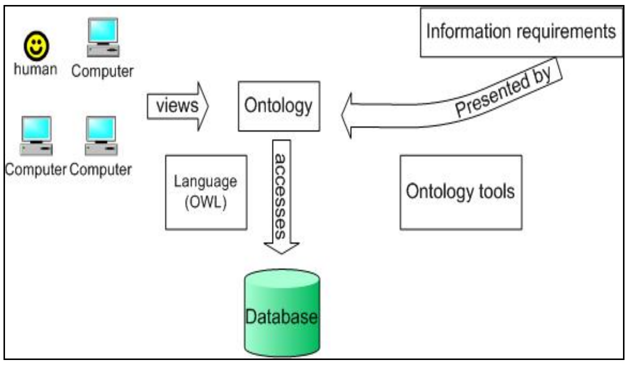


Figure 4: The ontology-database architecture

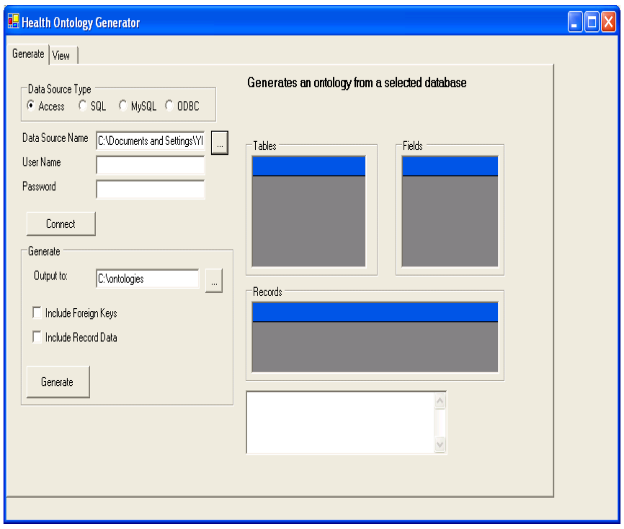
3. The Health Ontology Generator

This section deals with handling the user interface used to generate the ontology. Figure 5 shows the user interface as the system starts. Currently it displays only two tabs, one for ontology generation and one for viewing the resulting ontology.

3.1 Connecting to Database

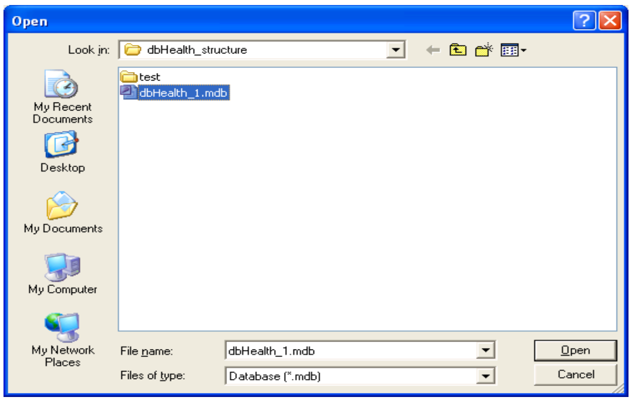
The first selection is to decide on the type of database to open, using the radio buttons provided. It is necessary to select the Data Source Name using the Select button, which opens the file-open dialog box shown in Figure 6.

The User Name and Password will have to be added if the database is locked by a password, before proceeding to click the Connect button. Once the connection is made, the system opens the database and displays the tables, fields and the records in the database as shown in Figure 7.



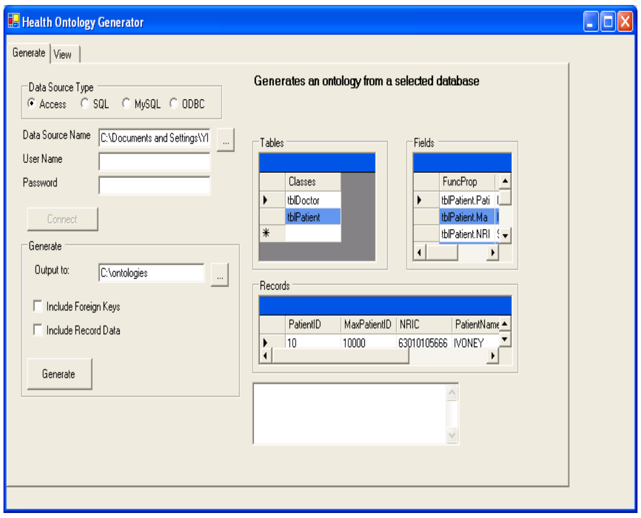
Data Source Type: [✓ Access]
Data Source Name: C:\Documents and Settings\Y\
Output to: C:\ontologies

Figure 5: Initial User interface of the Health Ontology Generator



Look in: dbHealth_structure
File name: dbHealth_1.mdb
Files of type: Database (*.mdb)

Figure 6: File open dialogue



Data Source Type: [✓ Access]
Data Source Name: C:\Documents and Settings\Y\
Output to: C:\ontologies

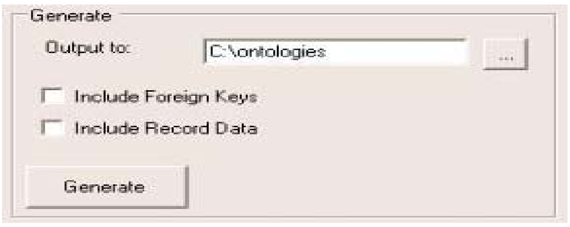
Tables: Classes, tblDoctor, tblPatient
Fields: FuncProp, tblPatient.Pati, tblPatient.Ma, tblPatient.NRI

PatientID	MaxPatientID	NRIC	PatientName
10	10000	63010105666	IVONEY

Figure 7: Displaying the Tables, Fields and Records

3.2 Generating the ontology

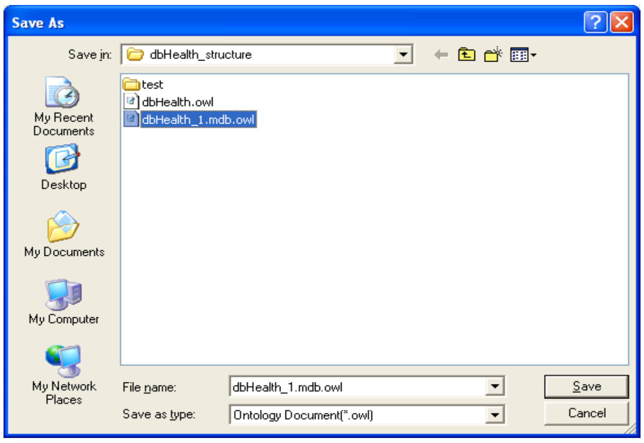
The next task is to generate the ontology from the selected database. It requires a few selections as shown in Figure 8.



Output to: C:\ontologies

Figure 8: Generating the ontology

The first step involves selecting the output file, which in this case is an OWL file. It is basically an RDF file.



Save in: dbHealth_structure
File name: dbHealth_1.mdb.owl
Save as type: Ontology Document(*.owl)

Figure 9: Selecting the output file

After selecting the output filename, we have to decide whether to include the foreign keys, and whether to include the data from each record in the database. However, the Access data provider does not give information about foreign keys, so this function does not work for Access databases.

Clicking the Generate button will generate the OWL file according the selections made.

4. Output of the Ontology Generated

In C#, the ontology is generated as a text file conforming to the syntax rules of OWL 1.1 [10]. In the design of the ontology, the first part of the OWL file declares it as an XML file. This is followed by the RDF declarations and the namespaces. The namespaces for the dbs, db, rdf, xsd, rdfs, and owl are declared. We have generated this part as shown in Figure 10.

Tables in the ontology are generated as classes, as shown in Figure 11.

Fields in each table are generated as functional properties. Figure 12 shows the statement declaring DoctorID as a functional property of the class called tblDoctor.

The Records of the data are generated as instances. Figure 13 shows the first instance of the class tblDoctor having three fields: DoctorID, Doctor Name and Specialist ID. The functional property data types and the instance values are inserted.

5. Reading the OWL File

There are several ways to view the generated ontology, using Internet Explorer, Microsoft Excel or Word. Subjecting the ontology to be read by these different programs shows the consistency of the data generated. It is also a quick way to debug the generated output before we had a method to read and verify the ontology.

5.1 Debugging OWL file

Internet Explorer is useful for debugging the RDF file generated as it gives useful debugging information. Figure 14 shows a sample of the debug message displayed during the process of debugging the output of the ontology. The error is indicated clearly using a caret at the part of the statement which is erroneous.

```
<?xml version="1.0" ?>
-             <rdf:RDF            xmlns:dbs="http://www.univ_ontology.com.my/RDF/relational.owl#"
  xmlns:db="http://zhiq.tripod.com/db_table_classes?DSNtype=Access:dbHealth_1#"
  xmlns="http://www.owl-ontologies.com/Ontology1227149346.owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
_xml:base="http://www.owl-ontologies.com/Ontology1227149346.owl">
  <owl:Ontology rdf:about="" />
```

Figure 10: Viewing the ontology using Internet Explorer

```
-
<owl:Class
  rdf:about="http://zhiq.tripod.com/db_table_classes?DSNtype=Access:dbHealth_1#tblDoctor">

<db:isBridgeTable
  rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</db:isBridgeTable>
  </owl:Class>
```

Figure 11: Tables are classes in the ontology

```
-
<owl:FunctionalProperty
rdf:about="http://zhiq.tripod.com/db_table_classes?DSNtype=Access:dbHealth_1#tblDoctor.Doctor
ID">
```

Figure 12: Fields are functional properties

```
-
<db:tblDoctor
rdf:about="http://zhiq.tripod.com/db_table_classes?DSNtype=Access:dbHealth_1#tblDoctor_Insta
nce_1">

<db:tblDoctor.DoctorID
rdf:datatype="http://www.w3.org/2001/XMLSchema#System.String">1</db:tblDoctor.DoctorID>

<db:tblDoctor.DoctorName
rdf:datatype="http://www.w3.org/2001/XMLSchema#System.String">RAJ</db:tblDoctor.DoctorName
>

<db:tblDoctor.SpecialistID
rdf:datatype="http://www.w3.org/2001/XMLSchema#System.Int32">10</db:tblDoctor.SpecialistID>
  </db:tblDoctor>
```

Figure 13: Instance Number One of the tblDoctor Table

```
Missing equals sign between attribute and attribute value. Error processing resource
'file:///C:/Documents and Settings/YIP/My
Documents/MySrc/MyThesis/dbHealth_structure/test/dbHealth_1.mdb.owl'. Line 37, Position 16
<db: tblDoctor
rdf:about="http://zhiq.tripod.com/db_table_classes?DSNtype=Access:dbHealth_1#tblDoctor_Instance_1">
---------------^
```

Figure 14: Debugging OWL file using Internet Explorer

ns1:about10	ns5:tblPatient.PatientID	ns1:datatype11	ns5:tblPatient.DoctorID	ns1:datatype12
....				
http://biostorm.stanford.edu/db_table_classes?DSN=jdbc:odbc:dbHealth_1#tblPatient_Instance_2	11	http://www.w3.org/2001/XMLSchema#int	2	http://www.w3.org/2001/XMLSchema#string
http://biostorm.stanford.edu/db_table_classes?DSN=jdbc:odbc:dbHealth_1#tblPatient_Instance_1	10	http://www.w3.org/2001/XMLSchema#int	1	http://www.w3.org/2001/XMLSchema#string

Figure 15 – Reading the ontology using Excel

5.2 Viewing Ontology using Microsoft Excel

Microsoft Excel 2003 can read the ontology file as an XML List, creating a schema based upon the XML source data. Each section is shown in its own column. The instance (i.e., the record data) is presented as in the snapshot in Figure 15. It displays the instance number and the instance value for each field.

5.3 Viewing Ontology using Microsoft Word

Microsoft Word 2003 can read the ontology in Web Layout View. It can display the property names alongside the actual data in each instance of the classes. This indicates the usefulness of the ontology that is generated using HOG.

5.4 Viewing Methods in HOG

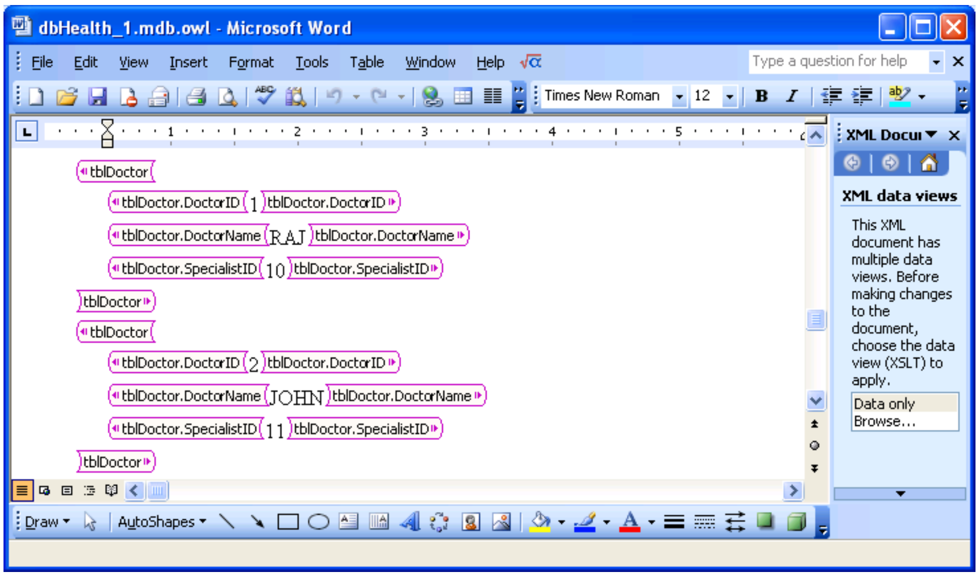
HOG has a tab which can view the ontology read as a stream, and place the classes and properties in list boxes.

The viewing of the ontology in this tab involves treating the ontology file as a stream. As the stream is read, it is parsed for specific keywords, such as

- the XML header,
- the RDF and namespace declarations,
- the class declarations,
- the functional properties declarations, and
- the instances in the ontology.

As they are parsed, the system adds the items in the various list boxes.

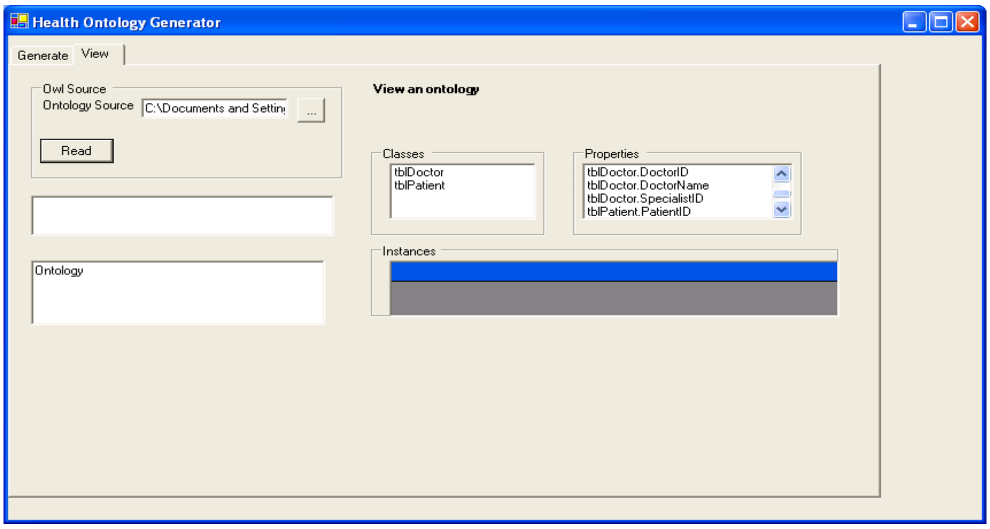
This functionality of reading ontology will be used to extract data from third party ontologies for the purpose of ontology integration and data-mining. Treating the ontology as a text file to be parsed appears to serve this purpose.



```
tblDoctor
    tblDoctor.DoctorID 1 tblDoctor.DoctorID
    tblDoctor.DoctorName RAJ tblDoctor.DoctorName
    tblDoctor.SpecialistID 10 tblDoctor.SpecialistID
tblDoctor
tblDoctor
    tblDoctor.DoctorID 2 tblDoctor.DoctorID
    tblDoctor.DoctorName JOHN tblDoctor.DoctorName
    tblDoctor.SpecialistID 11 tblDoctor.SpecialistID
tblDoctor
```

This XML document has multiple data views. Before making changes to the document, choose the data view (XSLT) to apply.

Figure 16: Web Layout View



Ontology Source: C:\Documents and Settin

Classes	Properties
tblDoctor	tblDoctor.DoctorID
tblPatient	tblDoctor.DoctorName
	tblDoctor.SpecialistID
	tblPatient.PatientID

Figure 17: Viewing ontology in HOG

6. Conclusion

In this research, we have built a system that from a database it has generated an ontology which can be read using third-party sources such as Microsoft Word, Excel and Internet Explorer. We have also built a stream reader to read and extract information from ontology. These are the building blocks that we have successfully built to proceed to the next stage in our study, that is, the integration of ontologies.

References

[1] Nyulas, C., O'Connor, M., Samson Tu, *DataMaster – a Plug-in for Importing Schemas and Data from Relational Databases into Protégé*, Stanford University School of Medicine, Stanford, CA 94305

[2] Gruber, T.R., A translation approach to portable ontologies, *Knowledge Acquisition,* Vol 5, No. 2, pp 199-220, 1993.

[3] Lee, T.B., Hendler, J., and Lasilla, O. , "The Semantic Web," *Scientific American*, May 2001.

[4] Updegrove, A., "The Semantic Web : An Interview with Tim Berners-Lee," *Consortium Standards Bulletin*, June 2005.

[5] Manola, F. and Miller, E. (Eds), RDF Primer, W3C Recommendation, 10 February 2004. Available at : http://www.w3.org/TR/rdf-primer/

[6] McGuiness, D. L. and Harmelen, F. V. (Eds), OWL Web Ontology Language Overview, W3C Recommendation, 10 February 2004. Available at : http://www.w3.org/TR/owl-features/

[7] Brickley, D. and Guha, R. V. (Eds), RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation, 10 February 2004. Available at : http://www.w3.org/TR/rdf-schema/

[8] Breitman, K. K., Casanova, M. A., and Truszkowski, W., *Semantic Web – Concepts, Technologies and Applications*, Springer-Verlag, London, 2007.

[9] Horridge, M., Knublauch,H., Rector,A., Stevens,R., Wroe, C. (2004) , *Protégé OWL Tutorial*, The University Of Manchester, Stanford University

[10] Motik, B., Patel-Schneider, P., Horrocks, I. (2007) *OWL 1.1 Web Ontology Language Structural Specification and Functional-Style Syntax*, http://www.webont.com/owl/1.1/owl_specification.html

Yip Chi Kiong obtained his Masters in Information Technology from University of Malaya. He is a Research Officer at the Department of Information Technology, Malaysia University of Science and Technology. His research interests include database systems, ontology development, data mining and web services.

Sellappan Palaniappan obtained his PhD in Interdisciplinary Information Science from University of Pittsburgh and a MSc in Computer Science from University of London. He is an Associate Professor at the Department of Information Technology, Malaysia University of Science and Technology. His research interests include information integration, clinical decision support systems, OLAP and data mining, web services and collaborative CASE tools.

Nor Adnan Yahaya obtained his PhD in Computer Science from Northwestern University, USA in 1987. He is an Associate Professor of IT at the Malaysia University of Science and Technology (MUST). His current research activities are focused on the development of tools and innovative applications related to emerging Web technologies such as web aggregation, web services, web agents, and the Semantic Web