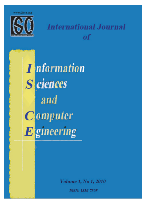
Information Sciences and Computer Engineering, Vol. 1, No. 2, (2010) 68–74

PHARUS

International Journal of
Information Sciences and Computer Engineering

journal homepage: http://www.ijisce.org



Ontology Distiller: Extracting Databases From Health Ontologies

Yip Chi Kiong, Sellappan Palaniappan*, Nor Adnan Yahaya

[a]Department of Information Technology, Malaysia University of Science and Technology, Kelana Square, Kelana Jaya, 47301 Petaling Jaya, Selangor, Malaysia

Abstract– **Many ontologies have been built over the past few years which contain a large amount of data that may be useful for specific research. However, they may not be designed to be easily processed by current database tools. This paper proposes an ontology distiller, which is a method to extract information from an ontology and create a set of databases. The database information is distilled from the encoded ontology. These databases are used to store the information for the purpose of analysis and integration.**

Keyword: Ontology encoding and generation, database schema, ontology viewing, ontology information extraction and integration, extracting ontology into database.

1. Introduction

We have presented a method to create an ontology from a database by analyzing the database schema and the data contained within the database [1]. The extraction of an ontology from a database have also been studied from the point of view of providing a conceptual view of the database [2, 3].

There are very few tools available to analyse and integrate ontologies directly. Since ontologies have a well-defined structure, it should be possible to extract information from them and store the data in databases, which have a vast array of tools to analyse and integrate. Our system, the Ontology Distiller, extracts selected information from an ontology to create databases containing information needed for a selected set of data for integration. Once the databases are created, it is possible to analyse the data using available tools. The follow-up action would be to integrate the data. There are many applications of databases extracted from an ontology. An ontology is generally generic, and extracting a database from an ontology can be used for specific data modeling [4]. After the data is integrated, an overall integrated ontology can be created using the method discussed in reference [1].

With the data integrated from various sources, that is, from extracts of ontologies and databases, the next step will be to generate an integrated ontology from the integrated database. This will be the next step in our research in ontology integration and the motivation for developing the Ontology Distiller. Information integration of existing databases across disparate departments within enterprises is a topic of great interest [5].

1.1. Purpose

This paper proposes a method to create a set of databases from a given ontology, which is the reverse process of our work to create an ontology from a database [1]. The Ontology Distiller presented will open a new tool to extract database information from an ontology and save it as various combinations of databases.

1.2. Review of Resources

There is very little work published on the process of creating databases from ontologies. After we have developed the Ontology Distiller, we have found a US patent [6], and a paper describing Knowledge Bus [7], which can create databases from an ontology using the Java platform. This paper uses application program interfaces (APIs) to reflect the entity types and relations (classes and methods) that are represented by the database.

Most of the work in the literature involves extracting an ontology from database. The Protégé system [8] has enabled us to develop our tools [1] to generate an ontology from a database. Some work on extracting ontologies from relational databases has been done in [2]. There is some work on constructing OWL ontology from XML document with the help of entity-relation mode [9]. There is an approach to deduce semantics from HTML forms while reverse engineering relational databases to ontologies [10].

1.3. Problem Statement

In our study of ontologies, several problems arose that we have to solve. Among these include:

1) How is an ontology encoded in such a way that information can be extracted from it?
2) How do we extract information from an ontology?
3) In what format should we store the extracted information?
4) What rules should be applied for the generation of the databases?

Ontologies are encoded with different forms of representing data. There are as many formats of representation as are the number of ontologies. In this paper, we present a format which was developed by Stanford University for generation of an ontology from a database [8]. The rules, in a modified and reversed format, are presented in this paper.

*Corresponding author:
Email address: sell@must.edu.my, Ph: +603 7880 1777 (ext 131)

1.4. Encoding an Ontology

An ontology represents a set of concepts within a specific domain of knowledge, and the relationships among these concepts. These concepts are described in terms of individuals, classes, attributes, relations and events. Individuals form the basic component on the ontology. These individuals or instances are grouped into collections called classes. Each class has its own properties or attributes. [11]

An ontology within a certain domain is a concept. The concept must be implemented in the form of an encoding. Current technology defines the encoding of an ontology as rdf tags in a txt file, often with an .owl extension. This text file is viewed as having a header, the body and the trailer. Within the body of this file lies the information from which data can be extracted to produce a database. The method of doing this is described in this paper.

2. Design of the Ontology Distiller

The main task of the Ontology Distiller is to extract relevant information from an ontology. To achieve this task, there are certain rules that we formulate that enables us to distinguish what should be included in the database. In our first prototype, these rules are very specific, and must be explicitly specified in the ontology.

The Process

The process of extracting database information follows the steps shown in Figure 1.

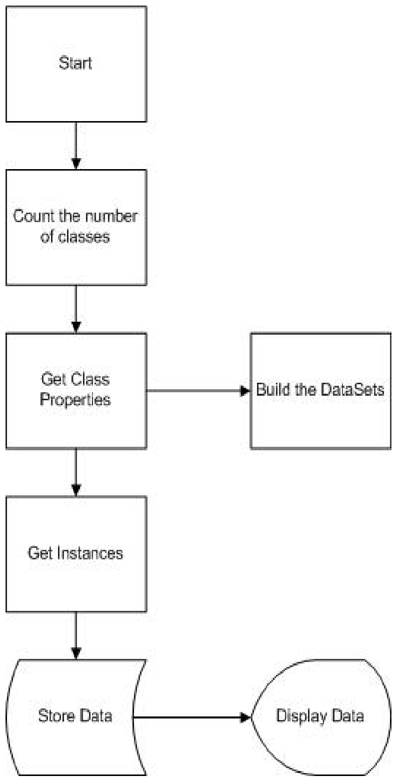


Fig. 1. The overall process.

Counting the number of classes is necessary to declare the amount of storage space to be allocated for the array of datasets. This is the first pass of the entire encoded owl file before any data is kept. The second pass involves allocating the actual classes and their properties and building the datasets. The third pass would read in the instances, their data types and their values. The resulting data would be stored. It may be displayed if necessary. Figure 1 shows these passes as the overall process.

As can be seen in Figure 2, there are three main sub-processes, namely, looking for the classes, looking for the properties, especially the functional properties, and looking for the instances. Each of these sub-processes require certain rules

2.1. General rules

Rule 1: It is expected that the ontology is encoded as an owl file which begins with the xml tag, namely "**?xml version**", with the version specified. This is part of the header of an xml file. Within the header, the namespaces should be declared. At the end of the header, there should be a tag for **owl:Ontology**

Rule 2: The next rule involves the tag needed to identify which are the database-specific items in the ontology. In the current version, this tag has to be specifically stated. In this case it is the **¡db¿** tag. Future versions of the Ontology Distiller will allow mapping of this tag to read various versions of ontologies.

Rules for Classes

Rule 3: The classes encoded in the ontology file are stored as tables in the database.

Rule 4: The classes need to be tagged to indicate their identity. Here the tag expected is **owl:Class**

Rules for Properties

Rule 5: The functional properties of a particular class with the semantics of "has original column" will be stored as properties or fields in the respective table. One example of tagging this property is **db:hasOrigColumnName**

Rule 6: The functional properties should include the tag **owl:FunctionalProperty.** This will identify the functional properties.

Rule 7: The data type of the property must be declared. One such tag would be **rdf:datatype**

Rules for instances

Rule 8: The instances from the ontology are stored as rows in the database.

Rule 9: The instances should enumerate themselves. One obvious method is to tag them as **Instance_** instance_number.

Rule 10. The data type should be declared. One possible way is to declare the datatype as a postscript to **XMLSchema#System** An example of such a tag would be **XMLSchema#System.String** in order to declare it as the *string* type. There are differences in the definitions of data types in the XML Schema and the database data types. To resolve this isuue, a mapping should be done to match the ontology data types to the database data type. Table 1 shows an example of a mapping between the XML Schema Type to Database Type.

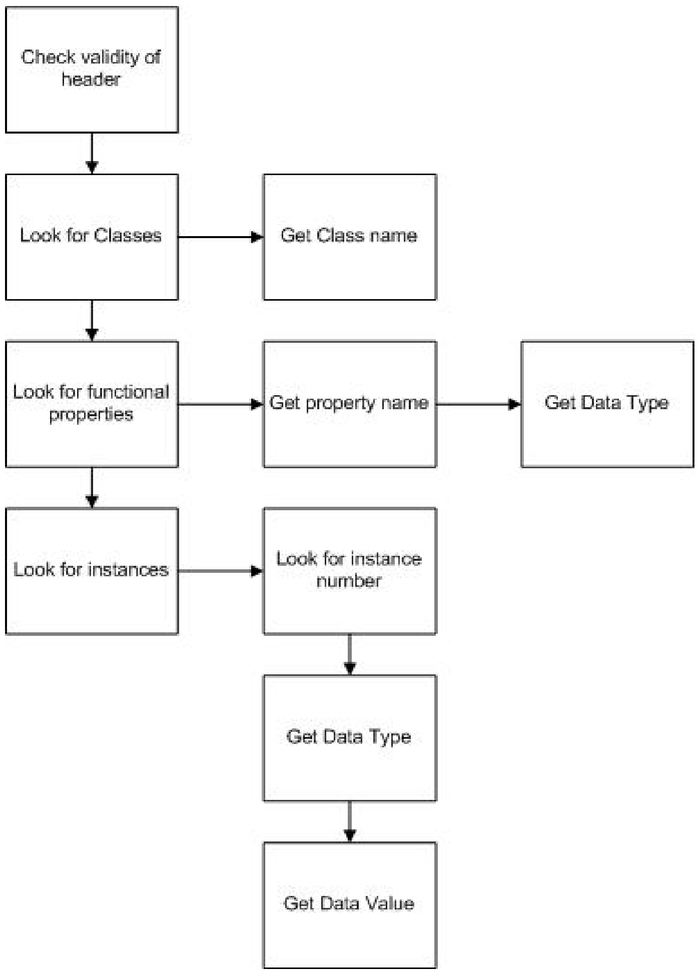


Fig. 2. The process of getting the data.

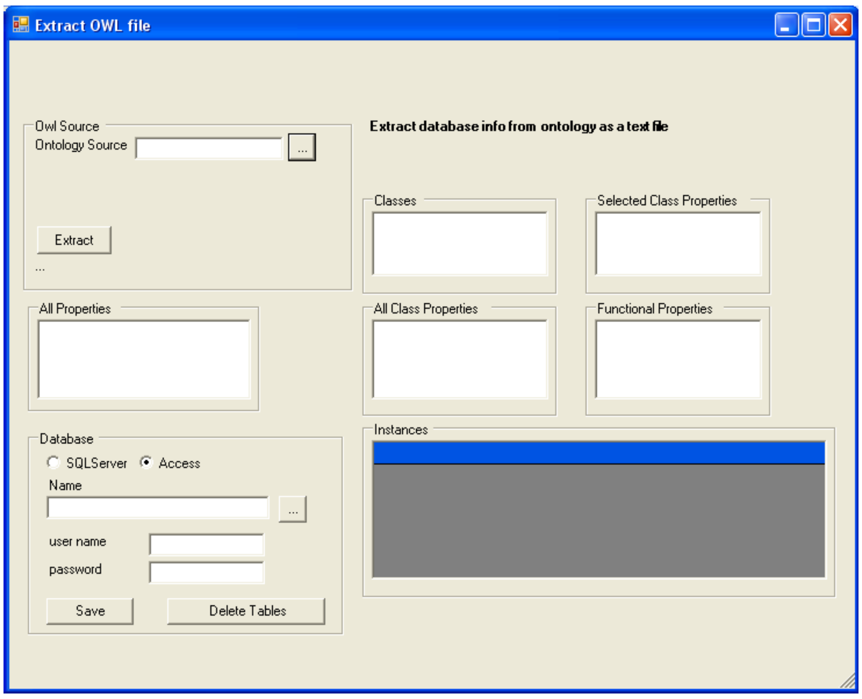
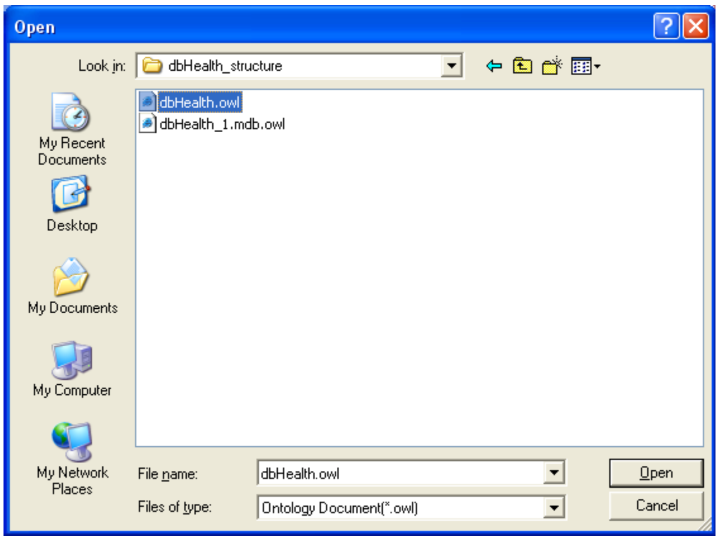


Fig. 3. The extract screen of the Ontology Distiller.

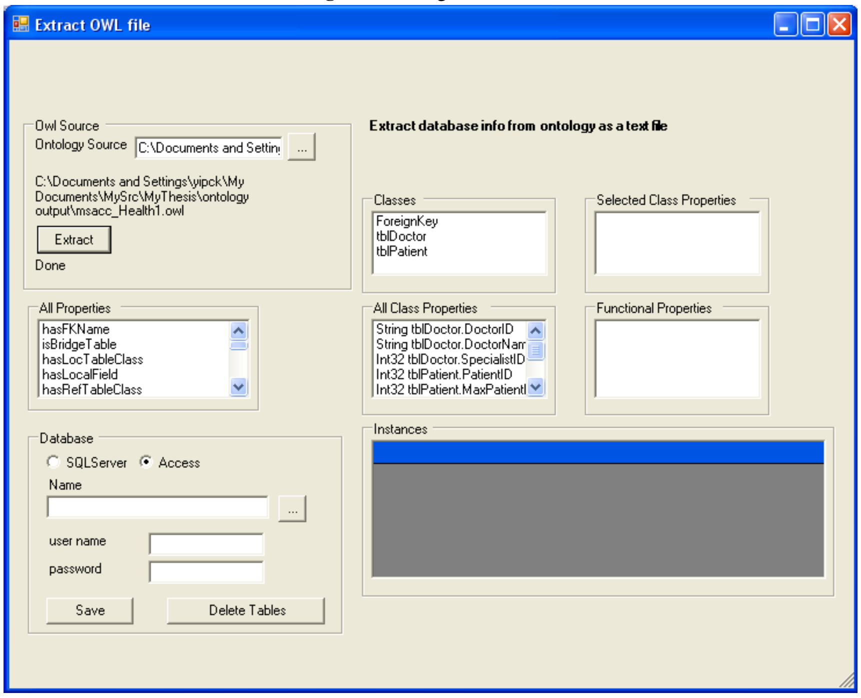


Look in: dbHealth_structure

dbHealth.owl
dbHealth_1.mdb.owl

File name: dbHealth.owl
Files of type: Ontology Document(*.owl)

Fig. 4. Selecting the owl file.


Extract OWL file
Extract database info from ontology as a text file
Extract
Save
Delete Tables
Instances
Selected Class Properties
Functional Properties

Owl Source
Ontology Source: C:\Documents and Settin

C:\Documents and Settings\vipck\My Documents\MySrc\MyThesis\ontology output\msacc_Health1.owl

Done

Classes:
ForeignKey
tblDoctor
tblPatient

All Properties:
hasFKName
isBridgeTable
hasLocTableClass
hasLocalField
hasRefTableClass

All Class Properties:
String tblDoctor.DoctorID
String tblDoctor.DoctorNam
Int32 tblDoctor.SpecialistID
Int32 tblPatient.PatientID
Int32 tblPatient.MaxPatientI

Database: SQLServer [✓ Access]
Name:
user name:
password:

Fig. 5. Extracting from a selected ontology.

Table 1. Mapping of XML Schema Type to Database Type.

XML Schema Type	Database Type
Int32	int
String	string
DateTime	string

3. Implementation of the Ontology Distiller

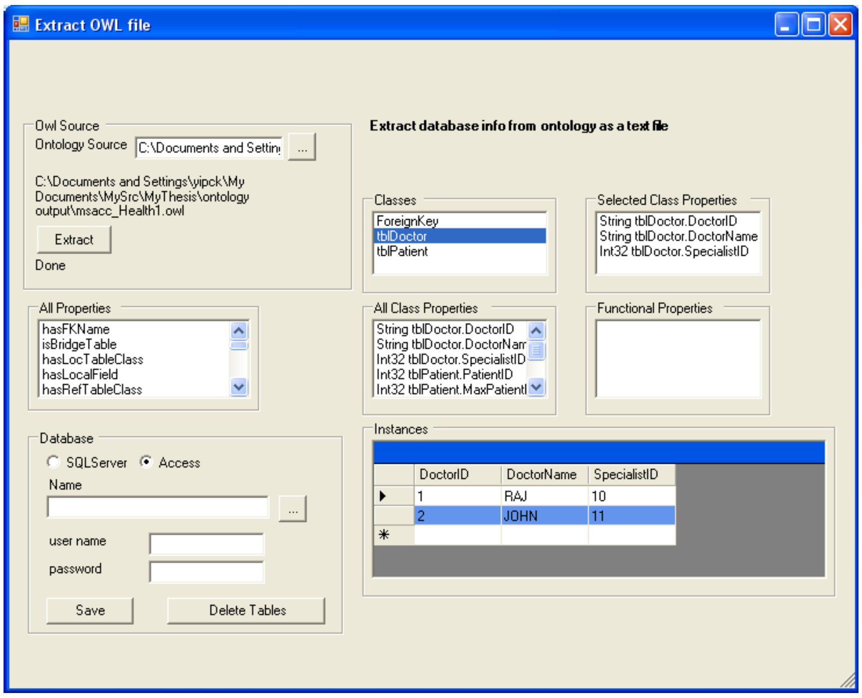
The Ontology Distiller is implemented using Microsoft Visual Studio 2008; the language selected for the coding is C#. This platform enables us to develop the system within a short time of development, as the tools provided assists in quick prototypes to model our ideas. As compared to a Java platform, many components must be developed to produce a new prototype. The part of the distiller that extracts data from an ontology into a database is shown in Figure 3.

Clicking on the owl sources button will open the dialogue box to select the owl file to extract, as shown in Figure 4.

Clicking the Extract button will display the classes in the Classes List Box. As can be seen on Figure 5, The classes are shown in the Classes listbox. The All Properties listbox will show all the properties extracted. The All Class Properties show the properties related to each class, with their data types.

Clicking on a selected class will display the Selected Class Properties and the instances as rows in the data grid. Figure 6 shows the results of clicking the tblDoctor Class.

After a class is selected, it is now ready to be saved into a database. In the current prototype, a database may be selected in Microsoft Access or SQL Server. Clicking the Database Name button will open the dialogue box to select the database, as shown



Ontology Source: C:\Documents and Settin

C:\Documents and Settings\vipck\My Documents\MySrc\MyThesis\ontology output\msacc_Health1.owl

Done

Classes: ForeignKey, tblDoctor, tblPatient

Selected Class Properties: String tblDoctor.DoctorID, String tblDoctor.DoctorName, Int32 tblDoctor.SpecialistID

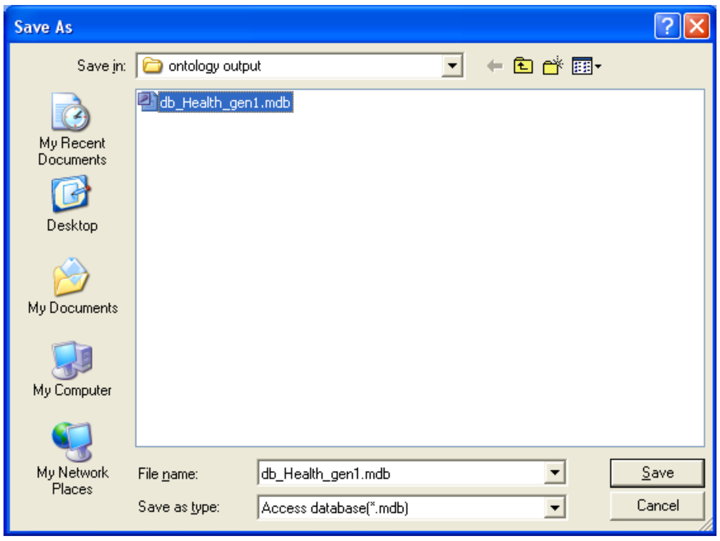
All Properties: hasFKName, isBridgeTable, hasLocTableClass, hasLocalField, hasRefTableClass

All Class Properties: String tblDoctor.DoctorID, String tblDoctor.DoctorNam, Int32 tblDoctor.SpecialistID, Int32 tblPatient.PatientID, Int32 tblPatient.MaxPatientI

Database: SQLServer [✓ Access]

	DoctorID	DoctorName	SpecialistID
▶	1	RAJ	10
	2	JOHN	11
*			

Fig. 6. The Selected Class Properties and the Instances for the tblDoctor class.



Save in: ontology output

db_Health_gen1.mdb

File name: db_Health_gen1.mdb

Save as type: Access database(*.mdb)

Fig. 7. Selecting the database.

in Figure 7. The tblDoctor table will be saved by clicking the **Save** button. This can be repeated for any selection of tables. The tables are *selectively* added and saved. Tables may also be deleted from the database.

4. Output of the Ontology Distiller

Figure 8 shows the tblDoctor and tblPatient tables generated by the Ontology Distiller. Figure 9 shows one of the tables, tblPatient data being displayed in Microsoft Access.

Tables can be generated in either Microsoft Access of SQL Server or both, showing the flexibility of the Ontology Distiller in the path towards integration of ontologies.

One interesting feature of the combined use of the Ontology Generator (presented in reference [1]) and Ontology Distiller discussed in this paper is the ability to generate an ontology from a SQL Server, then to distill it into a Microsoft Access database. Similarly, an ontology can be generated from a Microsoft Access database, and distilled into a SQL database.

This combination enables ontologies to be generated from various and diverse databases and distilled into one type of database. The next step in our study is to integrate all these ontologies into one master ontology.

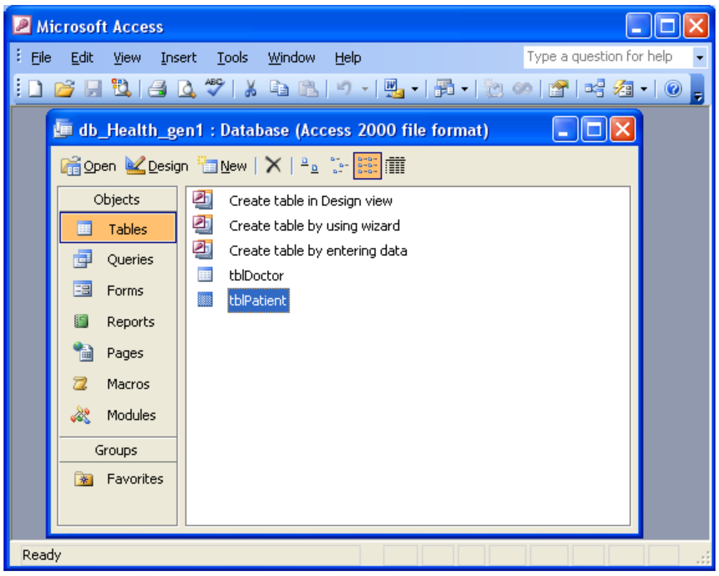
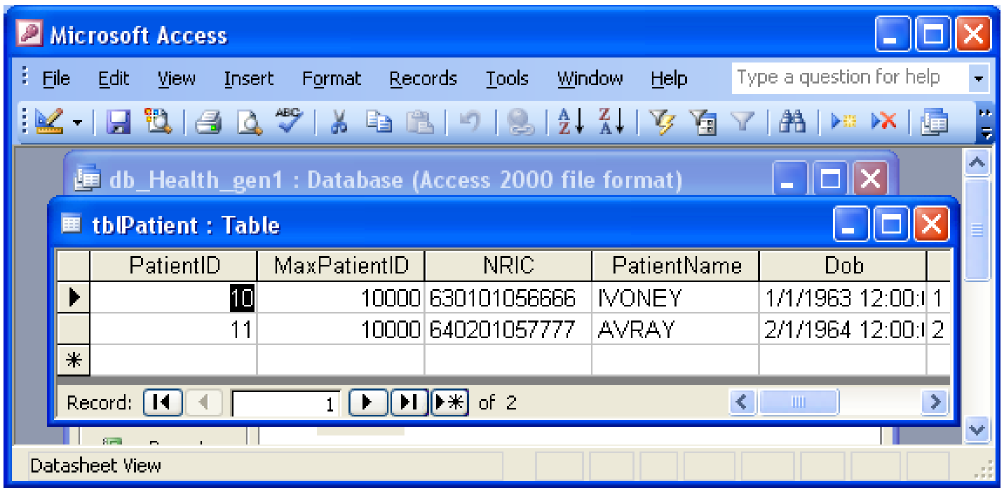

Microsoft Access
File Edit View Insert Tools Window Help
Type a question for help
db_Health_gen1 : Database (Access 2000 file format)
Open Design New
Objects
Tables
Queries
Forms
Reports
Pages
Macros
Modules
Groups
Favorites
Create table in Design view
Create table by using wizard
Create table by entering data
tblDoctor
tblPatient
Ready

Fig. 8. Tables generated by Ontology Distiller.


Microsoft Access
File Edit View Insert Format Records Tools Window Help
Type a question for help
db_Health_gen1 : Database (Access 2000 file format)
tblPatient : Table
Record: 1 of 2
Datasheet View

PatientID	MaxPatientID	NRIC	PatientName	Dob	
10	10000	630101056666	IVONEY	1/1/1963 12:00:I	1
11	10000	640201057777	AVRAY	2/1/1964 12:00:I	2

Fig. 9. The tblPatient table.

5. Related Work

Andersen, et al.'s work on "*Generating Application-focused Databases from Large Ontologies*", [6, 7], which is called Knowledge Bus, is a system which generates information systems – databases and programming interfaces – from application-focused subsets of a large ontology. The system uses a Java API to contain constraints and axioms. The sub-ontology extracts from the Cyc ontology, all of the concepts required to support representation and reasoning. They chose the XSB system as the basis for the underlying database. Knowledge Bus is tightly coupled with the Cys ontology and XSB system. Our Ontology Distiller is designed to eventually extract database information from more than one type of ontology, and extract it to more than one type of database. However, our work is developed in C#; hence we have to create our own system to process this conversion, using a different set of rules. The Microsoft Visual Studio platform enables quick prototypes to be developed.

Protégé is a platform that provides tools to construct domain models and knowledge-based applications with ontologies. A tutorial is available which gives a useful guide to the use of Protégé for creating an ontology [8]. Version 3.3.4 has a JDBC database backend that uses a plugin to create an ontology from a database. The study of the output generated has enabled us to design a tool to generate an ontology from a database, which was the focus of our earlier paper [1]. However, this is the reverse of the procedure researched in this paper, which is to distill a database from an ontology.

6. Conclusions

The Ontology Distiller provides a method to extract database information from an ontology and store it in any combination of databases and tables. Together with the Ontology Generator, we have a powerful set of tools approaching the path of ontology integration.

References

[1] Y. C. Kiong, S. Palaniappan, and N. A. Yahaya, “Health ontology generator: Design and implementation, malaysia university of science and technology,” *International Journal of Computer Science and Network Security*, vol. 9, no. 2, pp. 104–112, 2009.

[2] L. Lubyte and S. Tessaris, “Extracting ontologies from relational databases,” *Faculty of Computer Science, Free University of Bozen-Bolzano, Italy, http://www.inf.unibz.it/~lubyte/pub/dl07.pdf*, pp. 1–3, 2007.

[3] C. Batini, S. Ceri, and S. B. Navathe, “Conceptual database design. an entity-relationship approach,” *Benjamin/Cummings Publishing Company, Inc., ISBN 0805302441*, 1992.

[4] C. He-Ping, H. Lu, and C. Bin, “Research and implementation of ontology automatic construction based on relational database,” pp. 1078–1081, 2008.

[5] V. Alexiev, M. Breu, J. de Bruijn, D. Fensel, R. Lara, and H. Lausen, *Information Integration with Ontologies*. England: John Wiley & Sons Ltd., 2005.

[6] W. A. Andersen, P. M. Brinkley, J. F. Engel, and B. J. Peterson, “Ontology for database design and application development,” *United States Patent 6640231*, 2003.

[7] W. A. Andersen, P. M. Brinkley, J. F. Engel, and B. J. Peterson, “Knowledge bus: Generating application-focused databases from large ontologies,” pp. 2–3, 2003. US Department of Defense and Department of Computer Science, University of Maryland, College Park, U.S.A.

[8] M. Horridge, H. Knublauch, A. Rector, R. Stevens, and C. Wroe, “Protégé owl tutorial,” *The University Of Manchester, Stanford University, United Kingdom*, pp. 11–14, 2004.

[9] J. Xu and W. Li, “Using relational database to build owl ontology from XML data sources,” pp. 124–127, 2007.

[10] I. Astrova and B. Stantic, *Reverse Engineering of Relational Databases to Ontologies: An Approach Based on an Analysis of HTML Forms*. 2005. Tallinn University of Technology, Griffith University, Australia, http://olp.dfki.de/pkdd04/astrova-final.pdf.

[11] J. Cardoso and A. P. Sheth, *Semantic Web Services, Processes and Applications*. Springer, 2006.

Yip Chi Kiong obtained his Masters in Information Technology from University of Malaya. He is currently doing his PhD at the Department of Information Technology, Malaysia University of Science and Technology. His current research interests include database systems, ontology development, data mining, web services and programming in C#.

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 a Professor at the Department of Information Technology, Malaysia University of Science and Technology. His current 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. He is a Professor at the Department of Information Technology, Malaysia University of Science and Technology. 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.