RDB2OWL: AN IMPROVED METHOD FOR CONVERTING RELATIONAL DATABASES INTO OWL

Pham Thi Thu Thuy*

*The Faculty of Information Technology, Nhatrang University, Khanhhoa, Vietnam

Article history
Received: January 11th, 2017 | Received in revised form: April 11th, 2017
Accepted: May 17th, 2017

Abstract

One of the biggest advantages of the Semantic web is to describe data with a well-defined meaning and link between data by using the OWL (Web Ontology Language). Today most data are stored in relational databases. In order to reuse the data on the Semantic Web, there is a need for transforming the data stored in relational databases into the form of OWL Ontology. Some approaches have been proposed; however, most of their transformation rules have not been complete. This paper proposes some improved rules for transforming relational database into OWL Ontology. Most of all, all the steps in RDB2OWL are done automatically without any user intervention.

Keywords: Databases; Ontology; OWL; Transformation.

1. INTRODUCTION

From inception to date, the World Wide Web (WWW) has become an important tool to store and share huge sources of mankind knowledge. Most data on the WWW is currently stored in form of the relational databases (RDBs). The organization of data storage of relational databases (Andrew, 2009) offers many advantages such as: Efficient storage, an ability to execute complex queries, scalability, high security. However, RDBs are distinct, heterogeneous on schemas, terminology, and identification. Thus, Ontology was born for the purpose of providing the foundation for integrating all data sources. The conversion of data from RDB into an OWL Ontology is the solution to take advantage of and exploit the huge data available on the WWW.
Currently there are several methods of transforming relational databases into a given Ontology. Guntars (2010); Lei and Jing (2011); and Edgard, Percy, Karin, José, and Marco (2013) have proposed a method for automatically building ontologies from relational databases. However, this approach ignores a number of data tables showing links. Yutao’s method (Yutao, Lihong, Fenglin, & Hongming, 2012) has not represented the table with multi-valued attributes. Mohammed’s method (Mohammed, Hicham, & Said, 2013) has added mapping rules for N-ary relationships. Mona and Esmaeil (2015) proposed some common mapping rules from RDB to OWL, especially mapping rules for triggers to OWL. However, all the four methods above have not completed the mapping for binding on the properties, namely with CHECK constraints. The method of Nguyen, Hoang, and Le (2012) has fairly completed the conversion of the full review of tables, relationships, and constraints. However, there are irrationality CHECK constraints when they use the common mapping rule for the same attributes based on the primary key values, and this rule cannot be applied to a number of databases with identical primary key values. This paper follows the methods mentioned above to improve the mapping rules and mappings CHECK constraints.

2. DATABASE TRANSFORMATION INTO OWL ONTOLOGY

2.1. Transformation diagram

Relational databases are transformed into Ontology to be represented as an OWL Ontology. The conversion process consists of two steps:

- Schema mappings: This step is to extract information from the database schemas, then transforming them into concepts and properties in Ontology. Particularly, this step generates classes from the table, creates the object properties from the foreign key attributes and creates the type of data (data property) from the attribute which is not a foreign key.

- Data mapping: This step extracts data from the relational database (records) then stores them as the instances of the OWL Ontology.
The type of database tables is divided into six categories. Classification method is based on the number of fields that are the key (foreign and primary key), the correlation between the primary key and foreign key. The method is described in Table 1.

<table>
<thead>
<tr>
<th>Table type</th>
<th>Number of fields created primary key</th>
<th>Number of fields created foreign key</th>
<th>Correlation between the primary key and foreign key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base table</td>
<td>&gt;=1</td>
<td>0</td>
<td>Primary key does not create foreign key.</td>
</tr>
<tr>
<td>The table has a usual foreign key</td>
<td>&gt;=1</td>
<td>&gt;=1</td>
<td>Primary key also creates foreign key.</td>
</tr>
<tr>
<td>Inheritance table</td>
<td>&gt;=1</td>
<td>&gt;=1</td>
<td>Foreign key also creates primary key.</td>
</tr>
<tr>
<td>Multi-value table</td>
<td>2</td>
<td>1</td>
<td>Primary key also creates foreign key.</td>
</tr>
<tr>
<td>Table represents the pluralistic relationship having attributes.</td>
<td>&gt;=2</td>
<td>&gt;2</td>
<td>Primary key also creates foreign key.</td>
</tr>
<tr>
<td>Table represents the binary relationship</td>
<td>2</td>
<td>2</td>
<td>Primary key also creates foreign key.</td>
</tr>
</tbody>
</table>

For each type of table, we used the priority index to mark. Priority index of the base table is 1, the table has a foreign key is usually 2, 3 for inheritance table, multi-table value is 4, the table represents the pluralistic relationship having attributes is 5, the table represents the dualistic relationship is 6.

2.2. Algorithm for transforming RDB into OWL Ontology

The algorithm for transforming RDB into OWL Ontology is presented in Figure 2. The details of the algorithm command are explained by comments in each line.
/*DefineDatatype(string datatype): Function returns the corresponding data types in OWL, with input parameter is
For class in tableClassPriority
   If PriorityIndex== 1 || PriorityIndex== 2 || PriorityIndex== 5
      Create the corresponding class in OWL Ontology
   end if
   If PriorityIndex== 3
      Create the corresponding class in OWL Ontology
      Adding attribute rdfs:subClassOf
   end if
end for
// Attribute description
For attribute in tableAttribute
   // Describe Domain of the property
   If PriorityIndex!= 4 || PriorityIndex!= 6 && constraint != FOREIGN KEY
      Create the corresponding property in OWL Ontology
      Domain = domain[attribute]
   End if
   // Describe Range of the property
   If PriorityIndex!= 4 || PriorityIndex!= 6  && constraint != CHECK && constraint != FOREIGN KEY
      Create the corresponding data type property in OWL Ontology
      Range = DefineDataType(range[attribute])
   End if
   // Describe constraint UNIQUE and PRIMARY KEY
   If PriorityIndex!= 4 || PriorityIndex!= 6  && constraint == PRIMARY KEY || constraint == UNIQUE  && constraint != FOREIGN KEY
      Set the Functional for the corresponding property in OWL Ontology
   End if
   // Describe constraint NOT NULL
   If  PriorityIndex!= 4 || PriorityIndex!= 6  && isNullAttribute[attribute] == NO && constraint != FOREIGN KEY
      Set the constraint of minCardinality equal 1 for the corresponding property in OWL Ontology
   End if
   // Describe CHECK constraint
   If PriorityIndex!= 4 || PriorityIndex!= 6  && constraint == CHECK && constraint != FOREIGN KEY
      string checkClause :: conditional clause of CHECK constraint
      Consider checkClause to determine the type of CHECK constraint
      // CHECK (attribute IN (value1, value2, ...))
      If it is CHECK (attribute IN (value1, value2, ...))
         Assign the constraint owl:oneOf and owl:DataRange on the range of the corresponding property in OWL Ontology.
      end if
      // CHECK (attribute = value)
      If it is CHECK (attribute = value)
         Assign the constraint owl:hasValue as the same value as the corresponding value on corresponding property in OWL
      End if
      DefineDatatype(range[attribute]) == integer
      // CHECK (attribute > 0)
      If it is CHECK (attribute > 0)
         Describe the range by xsd:positiveInteger for the corresponding property in OWL Ontology.
      end if
      // CHECK (attribute > 0)
      If it is CHECK (attribute <= 0)
         Describe the range by xsd:nonPositiveInteger for the corresponding property in OWL Ontology.
      End if
      // CHECK (attribute < 0)
      If it is CHECK (attribute <= 0)
         Describe the range by xsd:nonNegativeInteger for the corresponding property in OWL Ontology.
      End if
      // CHECK (attribute != value)
      If it is CHECK (attribute != value)
         Assign the constraint owl:hasValue as the same value as the corresponding value on corresponding property in OWL
      End if
   end if
   // Describe constraint FOREIGN KEY
   If PriorityIndex!= 4 || PriorityIndex!= 6  && constraint != FOREIGN KEY
      Create the corresponding object property in OWL Ontology and set the minCardinality constraint equal 1 for this property
      Domain = domain[attribute]
      Range = range[attribute]
   End if

Figure 2. The algorithm for transforming RDB into OWL Ontology
Figure 2. The algorithm for transforming RDB into OWL Ontology (cont.)

3. EXPERIMENTAL RESULTS AND CONCLUSIONS

3.1. Experimental results

To simulate the conversion algorithm from RDB to OWL Ontology, we use the university sample database, namely Nhatrang University. The software used are Microsoft Visual Studio 2012 and Microsoft SQL Server 2012.

Table 2. Describing information for the university sample database

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Domain</th>
<th>Range</th>
<th>Constraint</th>
<th>Reference table</th>
<th>NULL acceptance</th>
<th>Conditional clause</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Khoa. MaKhoa</td>
<td>Khoa</td>
<td>varchar</td>
<td>PRIMARY KEY</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>1</td>
</tr>
<tr>
<td>Khoa. SoLuongGV</td>
<td>Khoa</td>
<td>int</td>
<td>CHECK</td>
<td>NONE</td>
<td>YES</td>
<td>([SoLuongGV] &gt;0())</td>
<td>1</td>
</tr>
<tr>
<td>Khoa. TenKhoa</td>
<td>Khoa</td>
<td>nvarchar</td>
<td>ATTRIBUTE</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>1</td>
</tr>
<tr>
<td>MonHoc. MaMonHoc</td>
<td>MonHoc</td>
<td>varchar</td>
<td>PRIMARY KEY</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>1</td>
</tr>
<tr>
<td>MonHoc. SoTC</td>
<td>MonHoc</td>
<td>int</td>
<td>CHECK</td>
<td>NONE</td>
<td>YES</td>
<td>([SoTC] &gt;=0())</td>
<td>1</td>
</tr>
<tr>
<td>MonHoc. TenMonHoc</td>
<td>MonHoc</td>
<td>nvarchar</td>
<td>ATTRIBUTE</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>1</td>
</tr>
</tbody>
</table>
### Table 2. Describing information for the university sample database (cont.)

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Domain</th>
<th>Range</th>
<th>Constraint</th>
<th>Reference table</th>
<th>NULL acceptance</th>
<th>Conditional clause</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>NghienCuu. MaDeTai</td>
<td>NghienCuu</td>
<td>varchar</td>
<td>PRIMARY KEY</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>1</td>
</tr>
<tr>
<td>NghienCuu. TenDeTai</td>
<td>NghienCuu</td>
<td>ntext</td>
<td>ATTRIBUTE</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>1</td>
</tr>
<tr>
<td>NhanVien. DiaChi</td>
<td>NhanVien</td>
<td>nvarchar</td>
<td>ATTRIBUTE</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>1</td>
</tr>
<tr>
<td>NhanVien. Email</td>
<td>NhanVien</td>
<td>nvarchar</td>
<td>UNIQUE</td>
<td>NONE</td>
<td>YES</td>
<td>NONE</td>
<td>1</td>
</tr>
<tr>
<td>NhanVien. HoNhanVien</td>
<td>NhanVien</td>
<td>nvarchar</td>
<td>ATTRIBUTE</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>1</td>
</tr>
<tr>
<td>NhanVien. MaNhanVien</td>
<td>NhanVien</td>
<td>nvarchar</td>
<td>PRIMARY KEY</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>1</td>
</tr>
<tr>
<td>NhanVien. TenNhanVien</td>
<td>NhanVien</td>
<td>nvarchar</td>
<td>ATTRIBUTE</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>1</td>
</tr>
<tr>
<td>BoMon. MaBoMon</td>
<td>BoMon</td>
<td>varchar</td>
<td>PRIMARY KEY</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>2</td>
</tr>
<tr>
<td>BoMon. MaKhoa</td>
<td>BoMon</td>
<td>Khoa</td>
<td>FOREIGN KEY</td>
<td>Khoa</td>
<td>YES</td>
<td>NONE</td>
<td>2</td>
</tr>
<tr>
<td>BoMon. TenBoMon</td>
<td>BoMon</td>
<td>nvarchar</td>
<td>ATTRIBUTE</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>2</td>
</tr>
<tr>
<td>GiangDay. HocKy</td>
<td>GiangDay</td>
<td>int</td>
<td>CHECK</td>
<td>NONE</td>
<td>YES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GiangDay. MaGiangDay</td>
<td>GiangDay</td>
<td>varchar</td>
<td>PRIMARY KEY</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>2</td>
</tr>
<tr>
<td>GiangDay. MaGiangVien</td>
<td>GiangDay</td>
<td>GiangVien</td>
<td>FOREIGN KEY</td>
<td>GiangVien</td>
<td>YES</td>
<td>NONE</td>
<td>2</td>
</tr>
<tr>
<td>GiangDay. MaMonHoc</td>
<td>GiangDay</td>
<td>MonHoc</td>
<td>FOREIGN KEY</td>
<td>MonHoc</td>
<td>YES</td>
<td>NONE</td>
<td>2</td>
</tr>
<tr>
<td>GiangDay. NamHoc</td>
<td>GiangDay</td>
<td>varchar</td>
<td>ATTRIBUTE</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>2</td>
</tr>
<tr>
<td>SinhVien. GioiTinh</td>
<td>SinhVien</td>
<td>nvarchar</td>
<td>ATTRIBUTE</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>2</td>
</tr>
<tr>
<td>SinhVien. HoSinhVien</td>
<td>SinhVien</td>
<td>nvarchar</td>
<td>ATTRIBUTE</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>2</td>
</tr>
<tr>
<td>SinhVien. MaKhoa</td>
<td>SinhVien</td>
<td>Khoa</td>
<td>FOREIGN KEY</td>
<td>Khoa</td>
<td>YES</td>
<td>NONE</td>
<td>2</td>
</tr>
<tr>
<td>SinhVien. MaSinhVien</td>
<td>SinhVien</td>
<td>varchar</td>
<td>PRIMARY KEY</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>2</td>
</tr>
<tr>
<td>SinhVien. TenSinhVien</td>
<td>SinhVien</td>
<td>nvarchar</td>
<td>ATTRIBUTE</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>2</td>
</tr>
<tr>
<td>GiangVien. MaBoMon</td>
<td>GiangVien</td>
<td>BoMon</td>
<td>FOREIGN KEY</td>
<td>BoMon</td>
<td>YES</td>
<td>NONE</td>
<td>3</td>
</tr>
</tbody>
</table>
### Table 2. Describing information for the university sample database (cont.)

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Domain</th>
<th>Range</th>
<th>Constraint</th>
<th>Reference table</th>
<th>NULL acceptance</th>
<th>Conditional clause</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>GiangVien. MaGiangVien</td>
<td>GiangVien NhanVien</td>
<td>FOREIGN KEY</td>
<td>NhanVien</td>
<td>NO</td>
<td>NONE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>DienThoai. MaNhanVien</td>
<td>DienThoai NhanVien</td>
<td>FOREIGN KEY</td>
<td>NhanVien</td>
<td>NO</td>
<td>NONE</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>DienThoai. SoDienThoai</td>
<td>DienThoai varchar</td>
<td>PRIMARY KEY</td>
<td>NONE</td>
<td>NO</td>
<td>NONE</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>KetQua. DiemTongKet</td>
<td>KetQua float</td>
<td>CHECK</td>
<td>NONE</td>
<td>YES</td>
<td>([DiemTongKet] &gt;= (0))</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>KetQua. MaGiangDay</td>
<td>KetQua GiangDay</td>
<td>FOREIGN KEY</td>
<td>GiangDay</td>
<td>NO</td>
<td>NONE</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>KetQua. MaSinhVien</td>
<td>KetQua SinhVien</td>
<td>FOREIGN KEY</td>
<td>SinhVien</td>
<td>NO</td>
<td>NONE</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>TacGia. MaDeTai</td>
<td>TacGia NghienCuu</td>
<td>FOREIGN KEY</td>
<td>NghienCuu</td>
<td>NO</td>
<td>NONE</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>TacGia. MaGiangVien</td>
<td>TacGia GiangVien</td>
<td>FOREIGN KEY</td>
<td>GiangVien</td>
<td>NO</td>
<td>NONE</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

![Figure 3. University sample database](image_url)
RDB2OWL program allows converting relational databases into a given Ontology. The conversion can be applied to any relational database. OWL file created can be opened by using the Ontology editor.

During the implementation process, there are five files that are created, including: _Attributes.xml, _ClassPriority.xml, _PkeyInheritance.xml, _MTRDB.xml, Ontology.owl. The content of those files is the results after converting Nhatrang University database. The content of those files is very long, so in this section, we present only a small section of the conversion results when transforming BoMon table into owl file (Figure 4).

```xml
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [ 
<!ENTITY owl "http://www.w3.org/2002/07/owl#" > 
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" > 
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" > 
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" ]>
<owl:Ontology rdf:about="http://example.com/2016onto" />
<owl:Class rdf:about="http://example.com/2016onto#BoMon">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://example.com/2016onto#BoMon.MaBoMon"/>
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
    </owl:equivalentClass>
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://example.com/2016onto#BoMon.TenBoMon"/>
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
    </owl:equivalentClass>
</owl:Class>
<owl:ObjectProperty rdf:about="http://example.com/2016onto#BoMon.MaKhoa">
  <rdf:domain rdf:resource="http://example.com/2016onto#BoMon"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://example.com/2016onto#inverseOfBoMon.MaKhoa">
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:about="http://example.com/2016onto#BoMon.MaBoMon">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdf:domain rdf:resource="http://example.com/2016onto#BoMon"/>
  <rdf:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://example.com/2016onto#BoMon.TenBoMon">
  <rdf:domain rdf:resource="http://example.com/2016onto#BoMon"/>
  <rdf:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>
<owl:NamedIndividual rdf:about="http://example.com/2016onto#BoMon-INS">
  <rdf:type rdf:resource="http://example.com/2016onto#BoMon"/>
  <BoMon.TenBoMon rdf:datatype="&xsd;string">Hệ thống thông tin</BoMon.TenBoMon>
  <BoMon.MaBoMon rdf:datatype="&xsd;string">INS</BoMon.MaBoMon>
</owl:NamedIndividual> 
</rdf:RDF>
```

Figure 4. The contents of into.owl file
3.2. Comparing results with other studies

We evaluated our proposed converting method by matching a relational database with an OWL file to determine the true matches and compared our results with other methods. To assess the quality of the matching system, we used precision and recall (Wikipedia, 2016). Given the set of expected matching pairs, R (produced by a human), the set of alignment pairs, T (produced by the matching system for the proposed methods), the precision is computed as in the following equation:

\[
\text{precision}(R,T) = \frac{|R \cap T|}{|T|} \tag{1}
\]

Recall specifies the share of real correspondences:

\[
\text{recall}(R,T) = \frac{|R \cap T|}{|R|} \tag{2}
\]

Although precision and recall are the most widely used measures, when comparing matching systems, one may prefer to have only a single measure. For this reason, F-measure (Wikipedia, 2016), is introduced to aggregate the precision and recall.

\[
F - \text{measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{3}
\]

To obtain practical evidence, we applied our transformation to two sample databases produced by Microsoft, particularly Microsoft (2011) and Microsoft (2013).

We compared the precision, recall, and F-measure values between our proposed method and the results of other studies, such as Edgard et al. (2013); Nguyen et al. (2012); Mona and Esmail (2015); and Yutao et al. (2012). The matching system is also implemented by using Visual Studio (C#). The compared results are shown in the following Figure 5 and Figure 6.
Figure 5. Matching comparison between our method and others’ on Northwind database

Figure 6. Matching comparison between our method and others’ on Pub Database

Figure 5 and Figure 6 show that our matching quality is the highest when compared to those of other studies. Nguyen (2012) is ranked second, followed by Edgard et al. (2013); Mona and Esmaeil (2015); and Yutao et al. (2012). The main reason is that our method (RDB2OWL) and Nguyen et al. (2012) transform all the CHECK constraints whereas the other three methods ignore this condition. Moreover, our method maintains the relationships between the foreign key and primary key among relations whereas other compared methods do not.

There are some small differences between Figure 5 and Figure 6 due to, the differences of Northwind and Pubs databases. Northwind database has 13 relations in comparison with 11 relations in Pubs database. Among those relations, there are relationships between foreign keys and primary keys. In this experiment, the total number of the relationships in the Northwind database is higher than that of Pubs database. Therefore, for those methods which do not maintain the foreign key and primary key
relationship, their matching results in the Northwind database are lower than those in the Pubs database.

3.3. Conclusions

Compared with other methods of conversion of reference, our method was more complete in mapping of CHECK constraint (CHECK form (attribute> 0), CHECK (attribute> = 0), CHECK (attribute <0), CHECK (attribute <= 0)) and the way to name the class.

First, the CHECK constraint (CHECK form (attribute> 0), CHECK (attribute> = 0), CHECK (attribute <0), CHECK (attribute <= 0)) in relational databases can apply under data type property about numbers (integers, real numbers) whereas, Nguyen et al. (2012) mapping rule is only used for integer data type. Therefore, when mapping this kind of constraint, we will review the data type of the property. If the type attribute is integer, then the mapping follows the rules specified by Nguyen et al. (2012), otherwise the attribute type in the Ontology is the corresponding data type in SQL.

Second, about the way to name instances for the class. In most of the related works, naming for instances will get by the value of the primary key. However, in a number of databases, the data type of the primary key is automatic number. That means the key values are the ascending integer. Therefore, when mapping this value there occurs the same name, so we cannot identify the class of this instance. So, when naming the instance of the class, we put the name of the class before primary key values to avoid having the same (by identical primary key values) because OWL Ontology requires that the name of the class in the Ontology is unique.

Finally, the transformation program into OWL Ontology is done automatically and the OWL file result complies with the format and syntax of the W3C and can be used directly by the application program without any supplements.
REFERENCES


RDB2OWL: MỘT PHƯỚNG PHÁP CẢI TIẾN TRONG VIỆC CHUYỂN ĐỔI CƠ SỞ DỮ LIỆU QUAN HỆ SANG OWL

Phạm Thị Thu Thúy*

*Khoa Công nghệ Thông tin, Trường Đại học Nha Trang, Khánh Hòa, Việt Nam
*Tác giả liên hệ: Email: thuthuy@ntu.edu.vn

Lịch sử bài báo
Nhận ngày 11 tháng 01 năm 2017 | Chỉnh sửa ngày 11 tháng 04 năm 2017
Cháp nhận đăng ngày 17 tháng 05 năm 2017

Tóm tắt

Một trong những lợi thế của Semantic Web là để mô tả dữ liệu với một ngữ nghĩa rõ ràng và liên kết giữa các dữ liệu bằng cách sử dụng ngôn ngữ OWL (Web Ontology Language). Ngày nay hầu hết các dữ liệu được lưu trữ trong cơ sở dữ liệu quan hệ. Để tận dụng lại các dữ liệu này, cần thiết phải có phương pháp chuyển dữ liệu lưu trữ trong cơ sở dữ liệu quan hệ vào định dạng của OWL Ontology. Một số phương pháp đã được đề xuất, tuy nhiên, hầu hết các quy tắc chuyển đổi đã không được hoàn chỉnh. Bài báo này đề xuất một số quy tắc cải thiện trong việc chuyển đổi cơ sở dữ liệu quan hệ sang OWL Ontology. Ngoài ra, tất cả các bước chuyển đổi trong suốt toàn RDB2OWL được thực hiện tự động mà không cần bất kỳ sự can thiệp của người dùng.

Từ khóa: Biến đổi; Cơ sở dữ liệu; Ontology; OWL.