Automatic Mapping of Relational Databases to OWL Antology

Larbi Alaoui
International University of Rabat
11100 Sala Al Jadida, Morocco

Oussama El Hajjamy, Mohamed Bahaj
Department of Mathematics and Informatics
University Hassan I, FSTS
Settat, Morocco

***Abstract*—Nowadays relational models are most frequently used to store, treat and extract data. Yet, the increase of semantic web technologies and the fast development of its applications based on ontology have made the problem of migrating RDB to OWL an active research domain. The difficulties with this problem lay especially in the treatment of semantic constraints of the data stored in RDB which let the transition from RDB to OWL require a thorough study of all characteristics of the data structures to be converted. In this context, we give a state of the art comparison of existing mapping methods from RDB to RDF/OWL and propose a novel migration solution that generalizes these methods, optimizes constraints extraction and retains the RDB source schema characteristics. A tool based on our approach has also been developed and tested to demonstrate the effectiveness and power of our strategy.**

Keywords— web ontology; semantic web; relational database RDB; OWL

I. INTRODUCTION

The use of ontology has been rapidly growing since the emergence of the semantic web. However, very large volumes of data are always stocked in relational databases and companies do always wish to keep them into existing systems having in mind the time and money already spent on them and the multiple associated software tools. Thus, instead of rebuilding the source databases and in order to make the existing data available for the semantic web, it is more suitable to find good solutions for the migration from relational databases (RDBs) to web databases. In this sense the research area of migrating relational databases to OWL ontology has attracted many researchers during the last years ([1]-[12]). This is because of the importance of the OWL (Ontology Web Language) as a semantic web language. Indeed OWL which was standardized by W3C in 2004 ([11], [12]) with its conceptual vocabulary and formal semantics is built to enrich the transition to semantic web by facilitating machine interpretability of web contents and making it possible for applications to process the semantic contents of web information.

However the existing studies do not provide a complete solution to the problematic of migrating RDBs to OWL ontology, and so far there still be no effective proposals that could be considered as a standard method that preserves the original structure and constraints of the relational database.

Our aim in this work is to analyze and provide an overview of RDB to OWL migration issues to identify the weaknesses and limitations of the different techniques and proposals, and to identify the differences between them in order to give a general mapping algorithm that covers all the constraints, preserves the semantics of RDB data and keeps the consistency and integrity of data. Our mapping model involves both the conversion of relational schemas and of relational data instances. To validate our approach we have developed a prototype that implements this algorithm and tested its effectiveness using concrete examples.

The rest of the paper is organized as follows. In section 2 we give a comparison of existing conversion methods from RDB to OWL. In section 3, we describe our conversion process by listing mapping rules for RDB tables, constraints and data. Section 4 provides our mapping algorithm based on the list of rules. In section 5, we present our tool that implements our algorithm. A conclusion of our work is the subject of the last section.

II. COMPARISON OF EXISTING MAPPING METHODS

In this section, we briefly introduce the methods targeted by our comparison for mapping relational databases to OWL.

A. RDB and OWL

OWL (Ontology Web Language) was proposed as an ontology language to fill in the gap that exists between the current Web contents and the Semantic Web. OWL provides semantic concepts and structures for the creation of schemas and related structured data document that can be processed on the semantic web. Such concepts and structures provided by OWL are indeed more adequate for learning semantic information from the huge amounts of data stored in relational databases which is critical for many web applications.

In a relational database data is stored in tables. A table is referred to as a relation and is a collection of records of the same type. Each row of the table represents a record and is a collection of a fixed number of values of the attributes of the record. The attributes (also called fields) of the records of a

table are also called the columns of the table. The relational schema of the database defines the structure and integrity constraints of its relational tables. Although this relational schema does not say much about the semantics of data, it contains implicit elements allowing to extract the inside implicit semantics. Because of all possibilities available in OWL one can say that associated OWL ontologies are semantically more expressive than relational schemas. The detailed conversion process we give in the following will give more insight into this fact.

B. Selected existing RDB to OWL mapping methods

Integration of relational databases is a key issue for the success of the semantic web since most of web data are stored in relational databases. To this end methods were proposed recently for migrating RDBs to OWL to learn ontology from such data.

For the comparison of existing mapping methods we consider the methods from the selected works of [1]-[3], [5], [8] and [10]. The authors in these papers developed mapping algorithms of RDB to semantic web based on rules related to the extraction of metadata from the data dictionary of the relational database. In the following section we give a detailed comparison of such rules with regards to the different conversion aspects. Other RDB to OWL mapping works were also done in [6], [7] and [9], but these could be considered as part of the selected papers cited above.

C. Comparison of selected works

In this section we investigate the approaches given in the papers [1]-[3], [5], [8] and [10] we selected for comparison purposes. We identify their similarities and their differences as well as their drawbacks. As in these papers we assume that relational database to be converted is at least in third normal form.

Relations Conversion

In [3], [1] and [8] all the relations (tables) of the relational schema are transformed into classes in web ontology. However, the works in [2], [10] and [5] have excluded the binary relations of this transformation.

Binary Relations Conversion

Identification of binary relations: It is said that R is a binary relation between two relations A and B if:

- A≠R and B≠R
- R contains exactly two attributes a and b
- a and b are primary keys in R
- a is a foreign key which references an attribute c in A
- b is a foreign key which references an attribute d in B

In [2], [10] and [5] every binary relation is converted into an object property "ObjectProperty" by associating with its domain and its range the two referenced relations. Let's note that in [5] another property declared as being the opposite of the first is generated for every object property. This statement is done thanks to the "inverseOf" property defined in OWL.

Atributes Conversion

All the methods targeted by our comparison convert the attributes which are not primary keys or foreign keys into data type properties "DataTypeProperty", and associate with its domain and range respectively, the class corresponding to the field table and the XSD type corresponding to the field type in RDB.

Primary Key Conversion

All the methods translate primary keys into data type properties "DataTypeProperty", by associating with its domain and its range respectively the corresponding field table class and the XSD type corresponding to field type in RDB, and add the "inverseFunctionalProperty" property to prevent null values insertion and duplicates in key fields.

Foreign Key Conversion

The foreign key constraint enables to maintain referential integrity between the different relations in RDB. This constraint engenders in all methods an object property "ObjectProperty" connecting the class representing the column table to the class representing the table referenced by the foreign key.

FK and PK in non binary relations

Madelle & al. [8] are the only ones who dealt with the case of a table that contains a PK and FK attributes and does not correspond to a binary relation by converting the table into subclass of the referenced table thanks to "subClassOf" property defined in OWL.

Cardinality Constraints Conversion

These constraints are considered only in [8] and [5]

- The unique constraint : In [8] Wondu & al. have used the "inverseFunctionalProperty" property to prevent creation of individuals having same values for each data type property that represents an attribute declared as "unique". In [5] Ling & al. have given value 1 to the "maxCardinality" of data type property which represents the attribute declared "unique" to not have different instances for this property.
- The Not Null constraint: to force insertion of a value for a "Not Null" attribute of each record, Wondu & al. and Ling & al. have given the value 1 to the "minCardinality" of the data type property that represents this attribute.

Transitive Relations

Let R1, R2 and R3 be three different relations. If there is a relationship between R1, R2 and another relation between R2 and R3, then there is a transitivity chain between R1 and R3.

The conversion of these relations is only considered in [5] thanks to the "TransitiveProperty" property defined in OWL.

Record Conversion

Except [8] which is only interested in relational database schema conversion, all the methods have converted the records of the database into OWL individuals.
For each record in the relational table, we generate an individual and fill it with values of all the record fields, including the primary keys and foreign keys, these values are used for assertion of data type property corresponding to this field.

III. A COMPLETE LIST OF MAPPING RULES

In this section we give a complete list of rules of ontology construction from relational database schema. The proposed conversion rules are based on considering all possible cases in RDB constructs. An algorithm summarizing our approach in these rules will be given in the following section. The algorithm and the associated tool can be applied to any relational database.

Our migration technique is divided into three distinct phases. The first phase deals with the structure of the relational database and its significance. In the second phase the metadata of relational schema is extracted with the record set of database. In the last phase we describe the process of migration to generate the structure and data of the OWL document.

To avoid any ambiguity of interpretation of the different identifiers of our ontology, we create a model parameterized by a namespace as follows:

- For classes, the namespace receives OntologyURI/DatabaseName#tableName
- For properties, the namespaces receives OntologyURI/DatabaseName#TableName-fieldName.

The different rules are described as follows.

A. mapRelation

Rule 1 "MapNormalRelation()"
Every Normal Relation who is not a binary relation will be mapped to an OWL Class.

```
<owl:Class  rdf:ID = "TableName"/>
```

Rule 2 "MapBinaryRelation()"
Every binary relation is converted into two mutually inverse Object-Properties.

```
<owl:ObjectProperty rdf :ID="RefTable1_RefTable2">
  <rdfs:domain  rdf:resource = "#RefTable1"/>
  <rdfs:range rdf :resource =  "#RefTable2"/>
</owl: ObjectProperty >
<owl:ObjectProperty rdf :ID=" RefTable2_RefTable1">
  <rdfs:domain  rdf:resource = "#RefTable2"/>
  <rdfs:range rdf :resource =  "#RefTable1"/>
    <owl:inverseOf rdf:resource=”#RefTable1_RefTable2”/>
</owl: ObjectProperty >
```

Rule 3 "MapPKandFKNonBinaryRelation()"
For two different relationships T1 and T2, if the primary key of T1 is at the same time a foreign key that is referencing a field in T2, then the generated class from T1 must be a subclass of the T2 mapping generated class.
The case of empty binary relation does not belong to this conversion rule since it was not translated to an OWL class.

```
<owl :class rdf :ID="T1">
  <rdfs :subClassOf rdf :resource="#T2"/>
</owl :class>
```

Rule 4 "MapTransitiveChain()"
For any relation T1, T2 and T3, if there is a foreign key relationship between T1 and T2 and if there is also a foreign key relationship between T2 and T3, then there is a transitive chain between T1 and T3.

```
<owl:ObjectProperty rdf :ID="TableName1_TableName3">
   <rdfs:domain  rdf:resource = "#TableName1 "/>
   <rdfs:range rdf :resource =  "#TableName3 "/>
    <rdf :type rdf :resource="&owl;TransitiveProperty"/>
</owl: ObjectProperty >
```

B. mapAttribute

Rule 5 "MapNormalAttribute**()**"
Each normal attribute is converted to a data type property, by associating with its domain and range respectively the URI of the class corresponding to the table field and the XSD type corresponding to the type of the field in the RDB

```
<owl:DataTypeProperty rdf :ID="AttributeName">
    <rdfs:domain  rdf:resource = "#TableName"/>
    <rdfs:range rdf :resource = "&xsd ;AttributeType"/>
</owl:DataTypeProperty>
```

Rule 6 "MapPK()"
Primary keys attributes are converted to data type properties by adding the property "InverseFunctionalProperty" to ensure the uniqueness of their values.

```
<owl:DataTypeProperty rdf :ID="AttributeName ">
  <rdfs:domain  rdf:resource = "#TableName "/>
  <rdfs:range rdf :resource = "&xsd : AttributeType "/>
  <rdf :type rdf :resource="&owl;InverseFuctionalProperty"/>
</owl:DataTypeProperty>
```

Rule 7 "MapFK()"
Each foreign key attribute is converted to object property by associating with its domain and its range respectively, the URI of the class corresponding to the table of field and the URI of the class that represents the referenced table. To ensure atomicity of the attribute we declare the property as a"functionnalProperty".

```
<owl:ObjectProperty rdf :ID=TableName_RefTable">
  <rdfs:domain  rdf:resource = "#TableName "/>
  <rdfs:range rdf :resource =  "#RefTable "/>
  <rdf :type rdf :resource="&owl ;FuctionalProperty"/>
</owl: ObjectProperty >
```

C. mapConstraint

Rule 8 "MapUniqueAttribute()"
If the attribute is declared as UNIQUE, we set maxCardinality to 1, to prevent the creation of individuals having the same value.

```
<owl:Restriction>
    <owl :onProperty rdf:resource="#AttributeName"/>
   <owl:maxCardinality> 1 </owl:maxCardinality>
</owl : Restriction >
```

Rule 9 "MapNotNullAttribute()"

If the attribute is declared as NOT NULL, we set minCardinality to 1.

```
<owl:Restriction>
    <owl :onProperty rdf:resource="#AttributeName"/>
   <owl:minCardinality> 1 </owl:minCardinality>
</owl : Restriction >
```

Rule 10 "MapUniqueAndNotNullAttribute()"
For a relation and an attribute, the maximal and minimal cardinality of the property corresponding to the attribute is set to 1, if the attribute is declared as UNIQUE and NOT NULL at the same time.

```
</owl : Restriction >
    <owl :onProperty rdf:resource="#AttributeName"/>
    <owl:minCardinality> 1</owl:minCardinality>
     <owl:maxCardinality> 1</owl:maxCardinality>
 </owl : Restriction >
```

D. mapRecordSet

Rule 11 "MapRecords()"
Each record of RDB is converted to an individual of ontology (or assertion) whose type is the class that represents the record table. And to guarantee the uniqueness of these individuals, we propose to give for each of them a name obtained by concatenating the name of the table and the primary key value corresponding to the converted record.

```
<owl :NamedIndividual rdf :ID=" TableName _idTuple">
    <rdf :type rdf :resource="#TableName ">
    <Attribute1 rdf :dataType="&xsd :TypeAttribute1">
     Value </ Attribute1 >
    <Attribute2 rdf :dataType="&xsd :TypeAttribute2">
     Value </ Attribute2 >
     ----------------------------------------------------------------------
    <TableName_RefTable rdf:resource=" idTuple"/>
    (if there is a relationship with other tables)
</ owl :NamedIndividual>
```

The following table gives summarizes all mentioned rules and the approaches that have considered them.

TABLE I. RDB TO OWL MAPPING COMPARISON METHODS

Constraints	[3]	[1]	[2]	[10]	[8]	[5]	Our approach
MapNormalRelation	✓	✓	✓	✓	✓	✓	✓
MapBinaryRelation	×	×	✓	✓	×	✓	✓
MapNormalAttribute	✓	✓	✓	✓	✓	✓	✓
MapPK	✓	✓	✓	✓	✓	✓	✓
MapPKAndFKNonBinaryRelation	×	×	×	×	✓	×	✓
MapFK	✓	✓	✓	✓	✓	✓	✓
MapUniqueAttribute	×	×	×	×	✓	✓	✓
MapNotNullAttribute	×	×	×	×	✓	✓	✓
MapUniqueAndNotNullAttribute	×	×	×	×	×	×	✓
MapTransitiveChaine	×	×	×	×	✓	×	✓
MapRecordSet	✓	✓	✓	✓	×	✓	✓

IV. MAPPING ALGORITHM

Considering all results and discussions we gave above we now want to give our mapping algorithm that takes into consideration all the aforementioned rules. This algorithm converted all relations (tables) of our relational schema including attributes, constraints and transitivity relations.

```
MapShema(S)
 Input: Schema S
 Begin
     MapRelations(S)
     MapTransitiveChaine(S)
 End
```

Before converting the Relations MapRelations() distinguishes between three types of relationships:

- A binary relation is translated into ObjectProperty without converting its attributes;
- A relation that contains a PK and FK attributes is converted into class, and this class is declared as subclass of generated class from referenced relation;
- Any other normal relation is converted in OWL class.

```
MapRelations(S)
Input: Schema S, Table T
Begin
  For each Ti  in S loop
    If (isBinaryRelation(Ti )=true) then
          MapBinaryRelation(Ti )
    Else if (isPKandFKNonBinaryRelation(Ti )=true)
         then
          MapPKandFKNonBinaryRelation(Ti )
          MapAttributes(Ti )
     Else
          MapNormalRelation(Ti )
          MapAttributes(Ti )
     End if
   End loop
End
```

Before converting attributes of non-binary relations MapAttributes uses the meta-data from the data dictionary to define the associated field types:

- If a field is a primary key we converted it into a Data Type Property with MapPK() function.
- If a field is a foreign key, two inverse object property are generated thanks to MapFK() function.
- For any other attribute we use MapNormalAttribute() to convert it.

```
MapAttributes(T)
Input: Table T, Attribute A
Begin
      For each Ai in T loop
          If (isPK(Ai)=true) then
              MapPK(Ai)
          Else if (isFK(Ai)=true) then
              MapFK(Ai)
          Else
              MapNormalAttribute(Ai)
              MapConstraints(Ai)
          End if
      End loop
End
```

mapConstraint() algorithm maps relational database constraints into OWL as follows:

- If the attribute has a unique constraint, the maximum cardinality of the property is set to one.
- On the other hand if the attribute has a NOT NULL constraint, the minimum cardinality of the property is set to 1.
- If the attribute has the both constraint previously cited, the minimum and the maximum cardinality of the property are set to 1.

```
MapConstraints(A)
Input: Attribute A
Begin
      If ((isUniqueAttribute(A)=true )and
       (isNotNullAttribute(A)=true)) then
          MapUniqueAndNotNullAttribute(A)
      Else if (isUniqueAttribute(A)=true) then
          MapUniqueAttribute(A)
      Else if (isNotNullAttribute(A)=true) then
          MapNotNullAttribute(A)=true)
      End if
End
```

MapTransitiveProperty() finds all transitive relations in the relational data base and convert them to object property by adding the "TransitiveProperty".

```
MapTransitiveProperty(S)
Input: Schema S, Table T, Attribute A
Begin
   For each Ti  in S loop
     For each Aj in Ti loop
       If (isFK(Aj)=true) then
         T' = getReferencedTable(Aj, Ti)
         If ((Ti != T') and (isBinaryRelation(Ti)=false)
           then
             CheckTransitiveRelations(Ti, T)
         End if
       End if
     End loop
   End loop
End
```

```
CheckTransiveRelation(T, T')
Input: Table T, Table T', Attribute A
Begin
    For each Ai in T loop
      if (isFK(Ai)=true) then
        T'' = getReferencedTable(Aj, T')
            If ((T' != T'') and (T != T'')) then
                CreateTranstiveRelation(T, T'')
            End if
       End if
    End loop
End
```

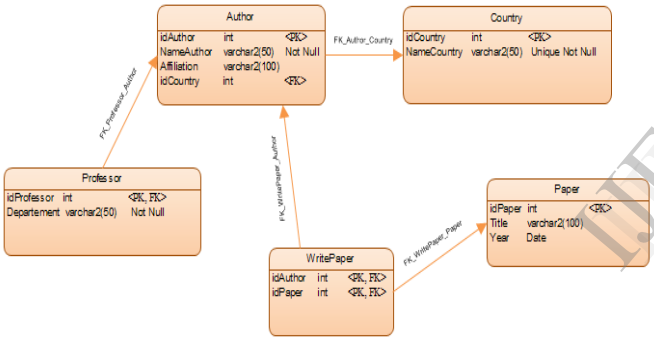
V. IMPLEMENTATION AND VALIDATION

In this section we devote our study to a simple example of a relational database schema, shown in Figure 1

To demonstrate the effectiveness and validity of our method, we developed a prototype that reads the relational database directly and generates the resulting OWL schema and instances documents.

To develop our prototype, we used Java as a programming language because Java is an object-oriented language, is compatible with all operating systems and can encode algorithms effectively.
To store the data and metadata we used Mysql DBMS which contains system tables that define the structure of the database (including names of tables, columns, constraints, ...). Our implementation can however also work with any other relational database system.
We used the JDBC-API to establish a connection with the migrated database. This API allows full access to relational database metadata and quickly retrieves a description of the tables and constraints of the database from data dictionaries.

Fig. 1. Relational Database Schema Overview



The mapping test between our example of relational database and owl ontology is shown by the sample screenshots in Figure 2 and Figure 3.

Fig. 2. Resulting mapping of RDB schema

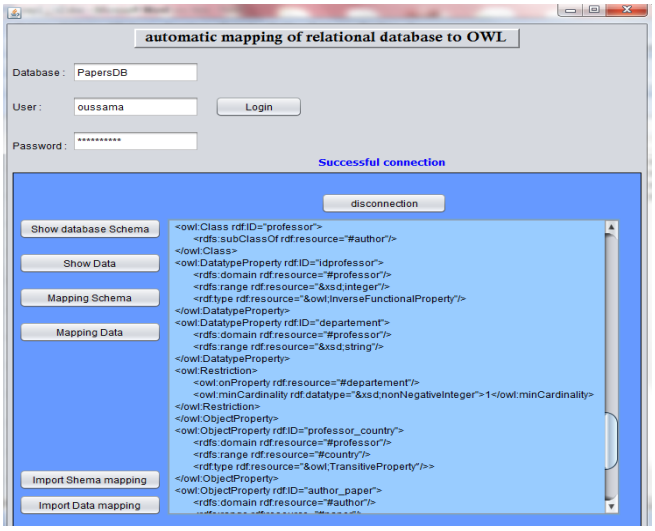


Fig. 3. Resulting mapping of RDB data

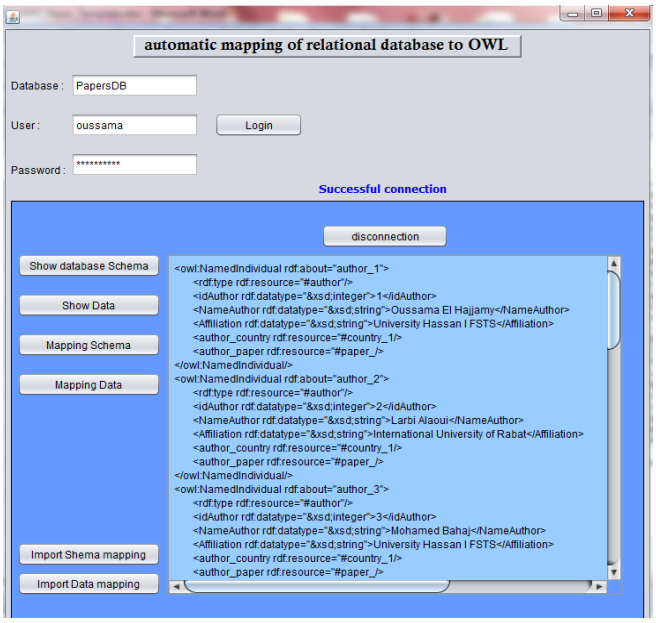
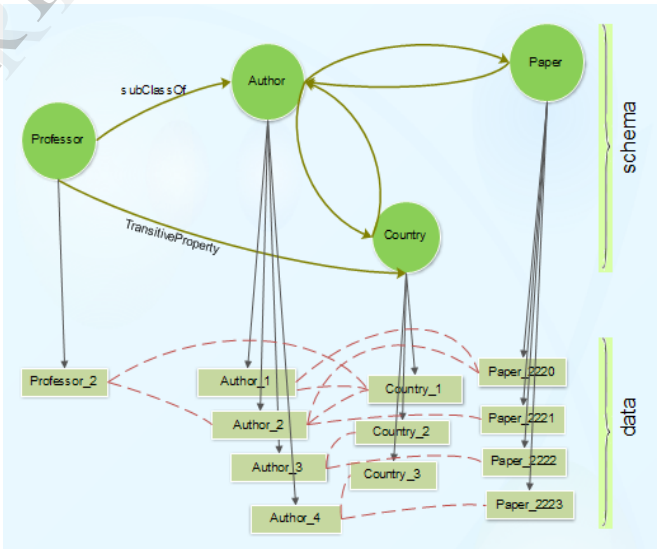


Figure 4 shows the resulting RDF graph of ontology with two levels, the schema level and the assertions level.

Fig. 4. the OntoGraph schema of resulting mapping



VI. CONCLUSION

In this paper we addressed the problem of automatic conversion of relational databases into OWL ontologies which has become one of the most attracting research area to enrich the success of semantic web. We especially gave a thorough analysis and comparison of existing direct automatic mapping methods from RDB to OWL, extracted weaknesses and limitations of theses methods. As a result we gave a complete list of elements that are crucial for the conversion and a

complete list of associated mapping rules. This has allowed us to build an associated general and complete mapping algorithm that covers different aspects of the relational model which are relevant for the mapping process. The algorithm deals among others with various multiplicities for relationships, relation transitivity and constraints such as primary key, foreign key, UNIQUE and NOT NULL constraints. The algorithm first converts the RDB schema into an ontology model and converts in a second step the RDB data into instances of the result ontological schema. The results obtained from our prototype prove the accuracy and performance of our mapping strategy.

REFERENCES

[1] J. Bakkas, M. Bahaj, "Direct Migration Method of RDB to Ontology while Keeping Semantics," International Journal of Computer Science and Information Security, vol. 65, No. 3, March 2013.

[2] J. Bakkas, M. Bahaj, "Generating of RDF graph from a relational database using Jena API," International Journal of Engineering and Technology, vol. 5, No. 2, Apr-May 2013.

[3] N. Gherabi, K. Addakiri, and M. Bahaj, "Mapping relational database into OWL Structure with data semantic preservation," International Journal of Computer Science and Information Security, vol. 10, No. 1, January 2012.

[4] G. Klyne and J. Carroll, "Resource Description Framework (RDF) Concepts and abstract syntax. W3C Recommendation 10 February 2004," World Wide Web Consortium. http://www.w3.org/TR/rdf-concepts/.

[5] H. Ling, S. Zhou "Mapping Relational Databases into OWL Ontology," International Journal of Engineering and Technology , Vol . 5, No. 6 , Dec 2013-Jan.

[6] M. Li, X. Y. Du and S. Wang , "Learning Ontology From Relational Database," Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005.

[7] M. R. Louhdi, H. Behja and S. O. EL Alaoui, "A novel Method for Generating an e-learning ontology," International Journal of Data Mining & Knowledge Management Process (IJDKP), Vol.3, No.6, November 2013.

[8] W. Y. Mallede, F. Marir, and V. T. Vassilev, "Algorithms for Mapping RDB Schema to RDF for Facilitating Access to Deep Web ," WEB 2013 : The First International Conference on Building and Exploring Web Based Environments, IARIA, 2013.

[9] C.Ramathilagam, M. L. Valarmathi, "A Framework for OWL DL based Ontology construction fro Relational Database using Mapping and Semantic Rules," International Journal of Computer Applications (0975– 8887), Volume 76– No.17, August 2013.

[10] J. F. Sequeda, M. Arenas, D. P. Miranker "On Directly Mapping Relational Databases to RDF and OWL," International World Wide Web Conference committee (IW3C2), WWW 2012, April 16–20, 2012, Lyon, France.

[11] OWL, "Web Ontology Language (OWL)," http://www.w3.org/2004/OWL, 2004.

[12] W3C, OWL Working Group, "OWL 2 Web ontology language document overview. W3C Recommendation 27 October 2009," http://www.w3.org/TR/owl2-overview/.