International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 11, November-2014

RDB2OWL2: Schema and Data Conversion from RDB into OWL2

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*— In this paper we propose a process of automatic mapping of relational database (RDB) schema and data to OWL2. This process is an extension of our previous work on converting RDB to OWL by considering construction elements of OWL2 and other important RDB aspects such as those related to self relation relations, cyclic relations, check constraints and binary relations with attributes. Our process retrieves the metadata of the relational schema, extracts the semantics of its data and provides a model of ontology while covering the semantic of the source database, and then populates the ontology by individuals using the existing records in the various tables. In order to apply our approach in real environments, we have developed a tool RDB2OWL2 that implements our mapping algorithm for our conversion model and demonstrates the effectiveness and power of our strategy.**

***Keywords*— Ontologies; semantic web; relational database RDB; OWL 2**

I. INTRODUCTION

Applications based on ontologies are more and more numerous and evolving very fast particularly due to the development of semantic web technologies ([1]-[27]). However the large masses of data are always stored in relational databases (RDB). Therefore the need to find a migration solution which extracts the semantics of the data stored in RDB and uses them to construct views of dynamic data (ontologies) is a very active research area. As a result, this problem has been the subject of a large body of research work in recent years and various methods have been proposed to come up with solutions to it [1-6], [8-10] , [13-14] and [17-18].

In our previous work [3] we presented an investigation into approaches and techniques used for converting RDB into OWL. We analyzed existing conversion works and identified different gaps and problems within them. We developed a model that extracts all implicit and explicit information contained in RDB such those related to dependencies between relations (e.g., transitivity, binary relations) and different constraints (e.g., integrity constraints, unique, not Null).

In the present work we aim to extend our approach given in [3] to address other very important aspects that have not been touched yet in the world of conversion from RDB to OWL. These aspects are mainly related to circular relationships, self-referenced relationships, binary relations with additional attributes including many-to-many relations, and check constraints (Check values, Check in). We also use OWL2 as a target ontology language to achieve a significant improvement of our previous conversion model by adding the aforementioned aspects while keeping the semantics of the RDB data and respecting their consistency and integrity.

It is to be noticed that the use of OWL2 to build the resulting ontology allows us to benefit from a system of inference that is more powerful as well as from the possibility of a further check of consistency. OWL2 was adopted as a W3C recommendation in December 2009 [25]. It extends OWL 1 with new features based on real use in applications. It is indeed possible with OWL2 to define more constructions to express additional restrictions and get new characteristics on the properties of object modeled. Also the functional-style syntax of OWL2 (also called abstract syntax) which is a compact syntax makes it possible to easy understand the structure of ontologies expressed in OWL2 [16].

The remainder of this paper is organized as follows. Section2 discusses the methods for extracting semantics using ontology engineering from relational databases and gives the proposed mapping rules. To illustrate how to combine the rules together, Section 3 outlines the automatic mapping algorithm. The implementation based on the conversion approach is presented in section 4. Finally Section 5 summarizes our work with a conclusion.

II. RDB TO OWL2 MAPPING MODEL

In this section we detail our migration solution from a relational database into a web format OWL2 and give a complete list of rules for building the ontology from the RDB source. This solution covers both the migration of the relational schema and of the relational data instances.

Our approach begins with the extraction of the structure of the source database using the metadata. Then, by applying the rules of transformation from RDB to OWL2 we create the classes and the properties of the objects and types of data that make up the model of the ontology.

In the next two sections we give an algorithm for our mapping model and an implementation of it. The algorithm creates the complete structure and data of the resulting ontology obtained by the conversion model.

A. RDB schema

Relational databases are a well established technology that allows storing data into tables according to a predefined schema. The schema of a database reflects the way how data are structured in form of tables.

The definition of Relational database result in a table of metadata or formal descriptions of the relations (tables), attributes (columns) and constraints (Integrity constraints, unique constraint, Not null constraint …).
The notations we adopt in this paper related to the information stored in the metadata of a relational database are the following ones.

- For relationships
- BinRel(R, A, B): R is a binary relation between two relations A and B.
- PKAndFKRelation(R): the primary key of R also acts as a foreign key.

- For Primary Keys
- PK(x, R): x is the single or composite primary key of the relation R.
- IsPK(x, R): return true if x is a single or composite primary key in relation R.
- NonPK(x, R): x is an attribute in relation R that does not a primary key.

- For foreign keys
- FK(x, R, y, S): x is a single or composite foreign key in relation R that references y in relation S.
- IsFK(x, R) : return true if x is a single or composite foreign key in relation R
- NonFK(x, R): x is an attribute in relation R that is not a foreign key.
- FKAttributeReferencedSameTable: A foreign key that references another attribute in the same table.
- RefTable: Referenced table.

- For attributes
- Attr(x, R): x is an attribute in relation R.
- twoAttr(x, y, R): R contains exactly two attributes x and y

B. Ontology preparation

Classes, data types, object properties and data properties are entities, and they are all are uniquely identified by a URI. So, to avoid any ambiguity in 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.

C. Mapping Relations

Before introducing our relationships mapping rules, we briefly give a new categorization for all types of relations. The relations are divided into the four following distinct types.

Binary relation

A relation R is called a binary relation BinRel(R, A, B) between two relations A and B if there exist a, b, c, d such that

- A≠R and B≠R
- twoAttr(a, b, R)
- PK(a, R) and PK(b, R)
- FK(a, R, c, A) and FK(b, R, d, B)

PK and FK relation

A relation R is called a primary and foreign key relation PKAndFKRelation(R) if there exist x, y such that

- PK(x, R)
- FK(x, R, y, S)

Many-to-many relation with additional attributes

It is any binary relation BinRel(R, A, B) with additional attributes for the relation itself.

Normal relation

Every relation R which is not a binary relation, a PK and FK relation and a many-to-many relation is called a normal relation.

The different mapping rules for relations are summarized in Table 1.

TABLE I. RULES FOR MAPPING RELATIONS

Rule	Rule Definition	Equivalent into OWL 2
R1	Every normal relation is converted into simple class	*Declaration(Class(:TableName))*
R2	Every binary relation is transformed into two object properties (*ObjectProperty*) that are mutually inverse	*Declaration(ObjectProperty(:RefeTable1_RefTable2))* *ObjectPropertyDomain(:ReTable1_RefTable2 :RefTable1)* *ObjectPropertyRange(:ReTable1_RefTable2 :RefTable2)* *Declaration(ObjectProperty(:RefTable2_RefTable1))* *ObjectPropertyDomain(:RefTable2_RefTable1 :RefTable2)* *ObjectPropertyRange(:RefTable2_RefTable1 :RefTable1)* *InverseObjectProperty(:RefTable1_RefTable2 :RefTable2_RefTable1)*
R3	If the primary key of a table T1 is at the same time a foreign key that is referencing a field in another table T2, then the generated class from T1 must be a subclass of the generated class from T2	*Declaration(Class(:T1))* *SubClassOf(:T2 :T1)*
R4	For each Many-to-many relation R with additional attributes we create a new class with two pairs of inverse object properties, and we add a data property for every additional attribute.	*Declaration (Class(:R))* *Declaration (ObjectProperty(: R_A))* *ObjectPropertyDomain(: R_A : R)* *ObjectPropertyRange(: R_A : A)* *Declaration (ObjectProperty(: A_R))* *ObjectPropertyDomain(: A_R : A)* *ObjectPropertyRange(: A_R : R)* *InverseObjectProperty(: R_A : A_R)* *ObjectPropertyDomain(: R_B : R)* *ObjectPropertyRange(: R_B : B)* *Declaration (ObjectProperty(: B_R))* *ObjectPropertyDomain(: B_R : B)* *ObjectPropertyRange(: B_R : R)* *InverseObjectProperty(: R_B : B_R)* *Declaration(DataProperty(:AdditionalAttribute))* *DataPropertyDomain(: AdditionalAttribute : R)* *DataPropertyRange(: AdditionalAttribute xsd: AdditionalAttributeType)*

D. *Mapping Attributes*

In relational data base, an attribute x in relation R can be one of the following

- Primary Key: PK(x, R)
- Foreign Key: FK(x, R, y, S)
- Normal attribute: NonFK(x, R) and NonPK(x , R).

Table II gives all associated conversion rules for such attributes.

TABLE II. RULES FOR MAPPING ATTRIBUTES

Rule	Rule Definition	Equivalent into OWL 2
R5	For each normal attribute we create a data type property by respectively associating with its domain and range the URI of the class corresponding to the attribute and the XSD type corresponding to the type of the attribute in the RDB	*Declaration(DataProperty(:AttributeName))* *DataPropertyDomain(:AttributeName :TableName)* *DataPropertyRange(:AttributeName xsd:AttributeType)*
R6	A primary key attribute uniquely identifies the records in relational database. This implies that the values of the data type property that represent this attribute must be unique. Therefore, these properties must be declared with HasKey properties. Declaring a predicate as a HasKey property is similar to saying that it is InverseFunctionalObjectProperty. The difference between both is that: • HasKey is applicable only to individuals that are explicitly named by an IRI in ontology. • InverseFunctionalObjectProperty is applicable to any kind of individual (named individual, anonymous individual, and any individual whose existence is implied by existential quantification).	*Declaration(Data Property(:hasAttributeName))* *DataPropertyDomain(:hasAttributeName :TableName)* *DataPropertyRange(:hasAttributeName xsd:AttributeType)* *HasKey(:TableName :hasAttributeName)*
R7	For relations R and S, if an attribute x in R references another attribute y in S, then an object property is generated, and with its domain and range we respectively associate the URI of the class corresponding to R and the URI of the class that represents S. To ensure atomicity of the attribute we declare the object property as a "FunctionalObjectProperty".	*Declaration(ObjectProperty(: R_ S))* *ObjectPropertyDomain(: R_ S : R)* *ObjectPropertyRange(: R_ S : S)* *FunctionalObjectProperty(R_ S)*

E. Mapping Constraints:

In our transformation rules, other constraints, such as UNIQUE, NOT NULL and CHECK are also taken into account to make the mapping complete. We aim to preserve as many constraints as possible. The associated conversion rules are given in table III.

TABLE III. RULES FOR MAPPING CONSTRAINTS

Rule	Rule Definition	Equivalent into OWL 2
R8	For each attribute A in a relation R with a UNIQUE constraint we set maxCardinality restriction to 1 in order to prevent the creation of individuals having the same value	*SubClassOf(:R DataMaxCardinality(1 :A xsd:TypeOfA [Optional])*
R9	For each attribute A in a relation R with a NOT NULL constraint we set DataMinCardinality restriction to 1	*SubClassOf(:R DataMinCardinality(1 :A xsd:TypeOfA [Optional])*
R10	If the attribute A is declared as UNIQUE and NOT NULL at the same time then we set DataExactCardinality to 1 (DataExactCardinality is equivalent to DataMinCardinality=1 and DataMaxCardinality=1)	*SubClassOf(:R DataExactCardinality(1 :A xsd:TypeOfA [Optional])*
R11	For attributes with the special constraints CHECK VALUES or CHECK IN Key, we treat them as follows: • CHECK number x: denotes all values that x can take. In this case we use the facets xsd:minInclusive, xsd:maxInclusive, xsd:minExclusive or xsd:maxExclusive. • CHECK IN constraint on a column allows only certain values for this column: In this case we use data range DataOneOf which is property defines a datatype with a fixed predefined value space.	The following expression contains all individuals that are connected by a data property hasX to an integer that is strictly less than 100: *DataSomeValuesFrom(a:hasX DatatypeRestriction(xsd:integer xsd:maxExclusive "100"^^xsd:integer))* The following expression shows that the weekend data property can take one of values "Sunday" or "Saturday" *DatatypeDefinition(* *:Weekend* *DataOneOf(" Sunday "^^xsd:String " Saturday "^^xsd:String))*

F. Mapping Transitive Chain

Let R1, R2 and R3 be three different relations. If there is a relationship between R1 and R2, and if there is another relation between R2 and R3, then there is a transitivity chain between R1 and R3. The associated transformation rule is given in table IV.

TABLE IV. RULES FOR MAPPING TRANSITIVE RELATIONS

Rule	Rule Definition	Equivalent into OWL 2
R12	For relations 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. We use *TransitiveObjectProperty* axiom to express it.	*Declaration(ObjectProperty(: T1_T3))* *ObjectPropertyDomain(: T1_T3 : T1)* *ObjectPropertyRange(: T1_T3 : T3)* *TransitiveObjectProperty(:T1_T3)*

G. Mapping Cyclic Relations

For a set of relations R1…Rn ($n \geq 1$) such that Ri is referenced by R(i+1) ($1 \leq i \leq n$) and Rn is referenced by R1, we say that a cyclic relationship exists between these relations. Note that if n= 1 then we get a self-referenced relation, and if $n \geq 2$ then we get a circular relationship between the relations. In this case we have the following two definitions.

Definition1. A self-referenced relation is defined as a relation which has a foreign key column referencing its own primary key ($\exists$ x, y $\in$ R / FK(x, R, y, R) and PK(y, R)).

Definition2. A circular relation is defined as a set of relations R1 ... Rn ($n \geq 2$), where Ri is referenced by Ri+1 ($1 \leq i \leq n$) and Rn is referenced by R1.

The mapping rules for self-referenced relations and circular relations are given in table V.

TABLE V. RULES FOR MAPPING CYCLIC RELATIONS

Rule	Rule Definition	Equivalent into OWL 2
R13	Each self-referenced relation is transformed to: • Object property by associating with both its domain and its range the name of the generated Class. • To ensure that the Object Property relates only 2 instances of the same class we add the self restriction objectHasSelf	*Declaration(ObjectProperty(:hasFKAttributeReferencedSameTale))* *ObjectPropertyDomain(:hasFKAttributeReferencedSameTale :TableName)* *ObjectPropertyRange(:hasFKAttributeReferencedSameTale :TableName)* *ObjectHasSelf(: hasFKAttributeReferencedSameTale)*
R14	A circular relation composed of a set of different relations can be transformed using a chain axiom property and self restriction objectHasSelf.	*SubObjectPropertyOf(ObjectPropertyChain(:R1_R2* *:R2_R3* --- *:Rn_R1) :Z)* *SubClassOf(ObjectHasSelf(:Z) : R1_R1)*

H. Mapping Records

The step of mapping records focuses on the conversion of records of the different RDB tables to OWL instances.

Each record is a set of pairs (attribute, value) indicating the value for an attribute of the record.
Table VI gives the conversion rule we adopt for such a conversion.

TABLE VI. RULES FOR MAPPING RECORDS

Rule	Rule Definition	Equivalent into OWL 2
R15	Each record of relational database (isNotBinRel) 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. Each record of a relation with a foreign key value which connects it to another record in another relation is converted into an individual containing an object property linking the classes corresponding to the two relations. For binary relations, we parse records from the table, and for each record we use SQL Queries to locate individuals that represent referenced records in order to link them to each other.	*ClassAssertion (:TableName :TableName_idTuple)* *DataPropertyAssertion(:Attribute1 TableName_idTuple "Value" ^^xsd:TypeAttribute1)* *DataPropertyAssertion(:Attribute2 TableName_idTuple "Value" ^^xsd:TypeAttribute2)* --- *ObjectPropertyAssertion(:TableName_RefTable :TableName_idTuple :RefTable_FK)* ***(if there is a relationship with other tables)***

III. MAPPING ALGORITHM

In this section, we present our algorithm for the automatic construction of OWL Ontology from a relational database. This algorithm takes into consideration all the aforementioned conversion rules.

A. Algorithm for mapping the RDB schema

The schema mapping procedure is divided into three steps. The first step converts every relation in our database schema and creates the equivalent ontology in owl2.

The second step finds all transitive relations in the relational database and translates them to object property by adding the TransitiveProperty axiom. The last step detects and extracts all circular relations in the database schema and converts them into OWL2 applying the mapping circular relations rule.

```
Procedure MappingShema(S)
Input: Schema S
Begin
    MappingRelations(S)
    MappingTransitiveChain(S)
    MappingCircularRelation(S)
End
```

Applying the mapping relation rules, the procedure MappingRelations() distinguishes between four types of relationships.

```
Procedure MappingRelations(S)
Input: Schema S, Table T
Begin
   For each Ti in S loop
      If (isBinaryRelation(Ti )=true) then
         MappingBinaryRelation(Ti )
      Else if (isPKandFKRelation(Ti )=true) then
         MappingPKandFKRelation(Ti )
         MappingAttributes(Ti )
      Else if (isManyToManyRelation(Ti )=true) then
         MappingManyToManyRelation(Ti )
         MappingAttributes(Ti )
      Else
         MappingNormalRelation(Ti )
         MappingAttributes(Ti )
      End if
   End loop
End
```

MappingAttributes() procedure uses the metadata from the data dictionary to define the field types.

We get a referenced table T' and for each foreign key attribute x in T, if:

- T= T' (FK(x, T, y, T), then we apply the self-referenced mapping rule
- If T≠ T', we apply the foreign key mapping rule.

```
Procedure MappingAttributes(T)
Input: Table T, Attribute A
Begin
   For each Ai in T loop
      If (isPK(Ai)=true) then
         MappingPK(Ai)
      Else if (isFK(Ai)=true) then
         T'=getReferencedTable(Ai, T)
         If (T=T') then
            MappingSelfReferencedRelation(Ai, Ti)
         Else
            MappingFK(Ai)
         End if
      Else
         MappingNormalAttribute(Ai)
         MappingConstraints(Ai)
      End if
   End loop
End
```

MappingConstraints() procedure is applied to each normal attribute, and cardinalities are learned from the metadata in data dictionary:

- if an attribute is NOT NULL, then the minimum cardinality is 1.
- if an attribute is UNIQUE, then the maximum cardinality is 1.
- if an attribute is UNIQUE and NOT NULL at the same time, then the exact cardinality is 1.

```
Procedure MappingConstraints(A)
Input: Attribute A
Begin
   If ((isUniqueAttribute(A)=true )
         and (isNotNullAttribute(A)=true)) then
            MappingUniqueAndNotNullAttribute(A)
   Else if (isUniqueAttribute(A)=true) then
            MappingUniqueAttribute(A)
   Else if (isNotNullAttribute(A)=true) then
            MappingNotNullAttribute(A)=true)
End
```

The following *MappingTransitiveChain()* procedure finds all transitive relations in the relational database and convert them to object property by adding the TransitiveProperty axiom.

```
Procedure MappingTransitiveChain(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
               CheckTransitiveChain(Ti, T)
            End if
         End if
      End loop
   End loop
End
```

The procedure *CheckTransiveChain()* used by *MappingTransitiveChain()* is given as follows.

```
Procedure CheckTransiveChain(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
            CreateTranstiveChain(T, T'')
         End if
      End if
   End loop
End
```

MappingCircularRelation() procedure uses a recursive function (FindCircularRelation()) to detect if there are any circular relations in relational schema.

```
Procedure MappingCircularRelation(S)
Input: Schema S, List ListOfTable, List ReultList
Begin
    ListOfTable=getAllListTable(S)
    While (ListOfTable != null) do
        tbl= ListOfTable.nextElement
        ResultList = FindCircularRelation(tbl, tbl, ResultList)
        If (ResultList != null) then
            ResultList.LastElement = tbl
        End if
        CreateCircularRelation( ResultList)
        Empty(ResultList)
    End while
End
```

The used *FindCircularRelation()* finds all circular relations in the considered relational database schema.

```
List FindCircularRelation(MainTable, Table, ResultList)
Input: String MainTable, String RelatedTable, List RefTableList
Output: List ResultList
Begin
    RefTableList=getReferencedTables(Table)
    While (RefTableList != null) do
        RelatedTable= RefTableList.nextElement
        If (MainTable=RelatedTable) then
            ResultList.FirstElement=RelatedTable
        Else if (FinInRefTableList(RelatedTable) = false) then
            ResultList=FindCircularRelation(MainTable,
                         RelatedTable, ResultList)
            If (ResultLis t!=null) then
                ResultList.nextElement=RelatedTable
            End if
        End if
    End while
End
```

B. *Algorithm for mapping records*

The migration process of data stored as tuples in RDB takes place in three stages. First, the relationships are divided into two types: normal relations and binary relations. Secondly, the RDB tuples are extracted using SQL queries. Finally, these extracted tuples are transformed into OWL2 format.

```
Procedure MappingRecords(S)
Input: Schema S
Begin
    MappingData(S)
    MappingDataOfBinaryRelations(S)
End
```

The *MappingData()* procedure is the following one.

```
Procedure MappingData(S)
Input: Schema S, Table T
Begin
  For each Ti in S loop
    If (isBinaryRelation(Ti)=false) then
      For each RS in T loop
        individualName = Cocatenate(Ti, "_", getPk(Ti))
        individualType = Ti
        For each P in RS loop
          PName = P.AttributeName
          PName = P.Value
          PType = P.AttributeType
          If (isFK(PName) = true) then
            Ref = getReferencedTable(PName, Ti)
            If(PValue != null) then
              OPAName = Cocatenate(Ti, "_", Ref)
              OPADestination = Cocatenate(Ref, "_", PValue)
              CreateObjectAssertion(OPAName, individualName,
                                    OPADestination)

            End if
          Else
            CreateDataAssertion(PName, idividualName, PValue,
                                individualType)
          End if
        End loop
      End loop
    Else
      For each RS in Ti loop
        P1 = RS. FirstElement
        P2 = RS.LastElement
        Ref1 = getReferencedTable(P1.AttributeName, Ti)
        Ref2 = getReferencedTable(P2.AttributeName, Ti)
        OPSource = Concatenate(Ref1, "_", P1.Value)
        OpDestination = Concatenate(Ref2, "_", P2.Value)
        CreateObjectAssertion(Ti, OPSource, OPDestination)
    End if
  End loop
End
```


IV. IMPLEMENTATION AND VALIDATION

The tool RDB2OWL2 we developed to implement our mapping algorithm from RDB to OWL2

- extracts schema and data of the to be converted relational database,
- extracts all circularly relations,
- maps the database schema to an OWL 2 model
- and maps the database data and generates a populated OWL2 ontology.

This tool demonstrates the effectiveness and validity of our method. For portability and interoperability purposes, we made it based on the Java programming language. This eliminates the need to rebuild (recompile and relink) the code when running the prototype in different platforms. The user interface of RDB2OWL2 was designed using Java Swing. RDB2OWL2. It is therefore to be considered as a MVC (model-view-controller) application.

To extract the data and schema information, we used MySQL. It is a multi user, multithreaded database management system and available on most important OS platforms. However any other relational database system can be used (Oracle, PostgreSQL, …). It is sufficient in this case to the appropriate JDBC driver for the database connectivity.

To illustrate the functioning of our tool RDB2OWL2 we consider the database below (Figure 1) which includes various characteristics and types of relationships between tables namely, primary keys, foreign keys, binary relations, and circular relations.

Figure 1 shows the records in the example tables and figure 2 shows an extraction of the associated schema.

Fig. 1. Example of a RDB with different types of relationships between tables

Paper

idPaper	titlePaper	Year
2220	automatic mapping of relational database to owl 2	2014
2221	The Value of Useless Information	2012
2223	Generating of RDF graph from a relational database...	2013
2224	Mapping Relational Databases into OWL Ontology	2014

WritePaper

idauthor	idpaper
2	2220
1	2221
1	2223
3	2224

Region

idRegion	NameRegion	idAuthor
1	Amerique	1
2	Afrique	2
3	Europe	3

Author

idAuthor	NameAuthor	idCity
1	Mallede	3
2	Oussama	2
3	Schneider	1

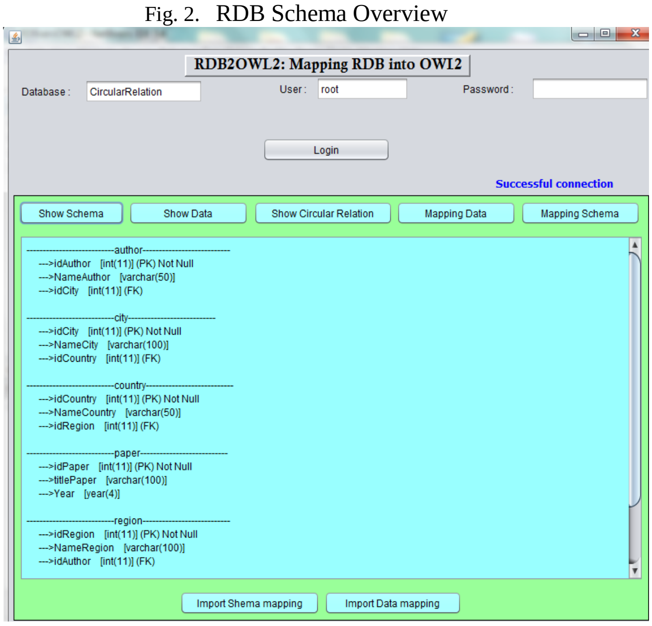
City

idCity	NameCity	idCountry
1	Beatty	2
2	Casablanca	1
3	Lincoln	3

Country

idCountry	NameCountry	idRegion
1	Morroco	2
2	England	3
3	USA	1

Fig. 2. RDB Schema Overview



The mapping results obtained by the RDB2OWL2 tool for this sample database (Figure 2) are shown by the sample screenshots of Figure 3.

Fig. 3. Resulting mapping of RDB schema

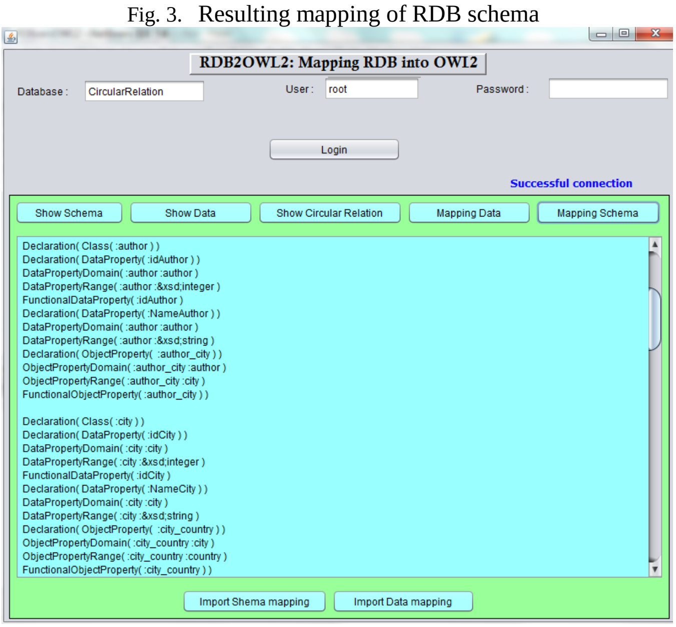
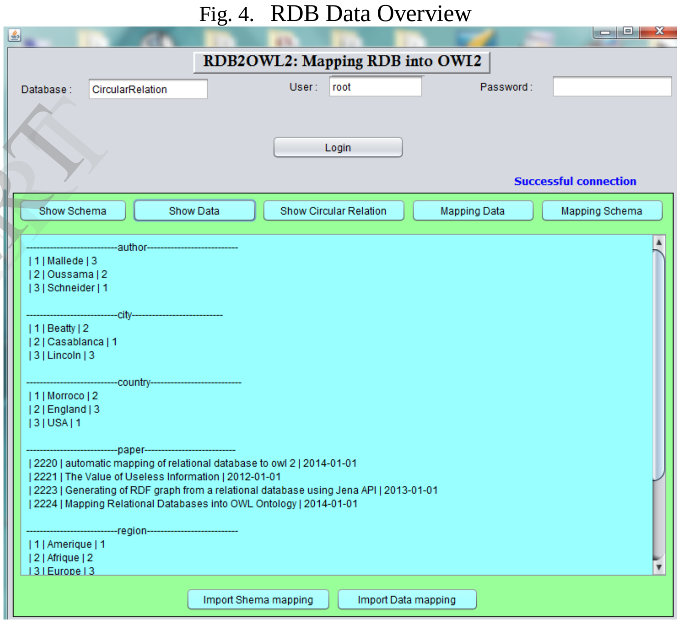
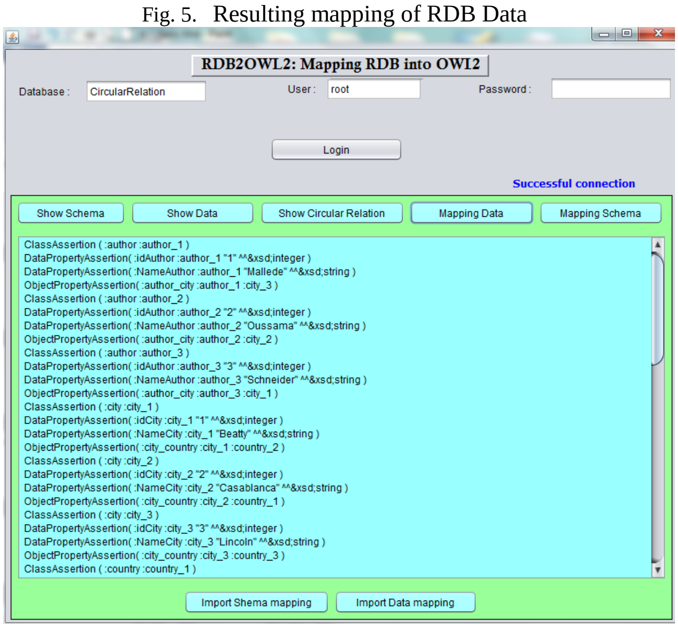


Fig. 4. RDB Data Overview



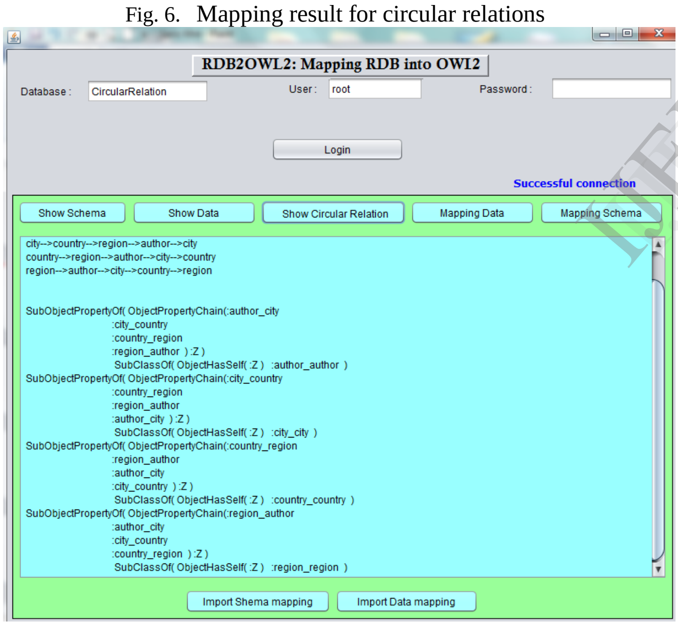
RDB2OWL2 also gives the possibility to extract and display the data (Figure 4) from the relational tables. The conversion of the data into OWL instances is done by applying the mappingRecords() algorithm as a screenshot of the associated display is shown in Figure 5.

Fig. 5. Resulting mapping of RDB Data



The sample screenshot in Figure 6 shows both the extracted circular relationships and their converted OWL parts.

Fig. 6. Mapping result for circular relations



V. CONCLUSION

The increasing use of ontologies in applications and the domination of relational databases with their over many decades developed technologies and tools have made the problem of migration of RDB to the web ontology a fertile area for researchers. In [3] we analyzed different existing works related to this topic and gave a model that generalizes existing works with a conversion approach that automatically extracts all the relevant structural and semantic information of the relations stored in databases.

In this paper, we have established a global solution that extends our previous mapping approach for a complete automatic transformation a relational data base into OWL2. Besides the structural constructs, our new model detects all restrictions and hierarchies between relations out from the tables of the database. As in our previous mapping solution, our new one provides necessary mapping rules for various multiplicities of relationships, different constraints, relation transitivity using OWL2 functional syntax. Besides all these considerations it also add mapping of circular relationship, self-referenced relationship, binary relations with additional attributes including many-to-many relations, and check constraints. To our knowledge the conversion related to these points has not been touched before.

Compared to our previous work, our new solution optimizes constraints extraction, and thanks to OWL 2 the rules are also refined to be more expressive and less complicated using more expressive constructs (e.g., hasKey, ObjectHasSelf, exactcardinality) and its powerful and easy to understand functional syntax. Indeed, OWL2 language provides a large variety of powerful constructs for building and reasoning over ontologies. OWL2 also simplifies many programmatic tasks associated with ontologies, including ontology querying and processing. In addition OWL2 can be used to construct full applications that have dependencies on complex ontologies.

REFERENCES

[1] H. Agus Santoso, S. C. Haw, Z. T. Abdul-Mehdi, "Ontology extraction from relational database: Concept hierarchy as background knowledge", Knowledge-Based Systems, vol.24, no.3, pp.457-464, 2011.

[2] N. Alalwan, H. Zedan, and F. Siewe, "Generating OWL Ontology for Database Integration," Third International Conference on Advances in Semantic Processing, Volume 76– No.17, August 2013.

[3] L. Alaoui, O. EL Hajjamy, and M. Bahaj, "Automatic Mapping of Relational Databases to OWL Ontology," International Journal of Engineering Research & Technology (IJERT), vol. 3, Issue. 4, April 2014.

[4] 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.

[5] 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.

[6] J. Barrasa, Ó. Corcho, A. Gómez-Pérez, "R2O: an Extensible and Semantically based Database-to-Ontology Mapping Language", In Proceedings of Second Workshop on Semantic Web and Database (SWDB2004), pp.1-17, 2004.

[7] C. Bizer, "D2R MAP - A Database to RDF Mapping Language", In Proceedings of the 12th International World Wide Web Conference (WWW2003), 2003.

[8] K. Čerāns, G. Būmans, "RDB2OWL: A RDB-to-RDF/OWL Mapping Specification Language", In Proceedings of the 2011 conference on Databases and Information Systems VI: Selected Papers from the Ninth International Baltic Conference (DB&IS 2010), pp.139-152, IOS Press, 2011.

[9] N. Cullot, R. Ghawi, K. Yétongnon, "DB2OWL: A Tool for Automatic Database-to-Ontology Mapping", In Proceedings of 15th Italian Symposium on Advanced Database System (SEBD 2007), pp.491-494, 2007.

[10] 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.

[11] M. Hert, G. Reif, H. C. Gall, "A Comparison of RDB-to-RDF Mapping Languages", In Proceedings of 7th International Conference on Semantic System (I-Semantics 2011), ACM, 2011.

[12] M. Li, X. Du, S. Wang, A Semi-automatic Ontology Acquisition Method for the Semantic Web. Advances in Web-Age Information Management, vol. 3739, pp.209-220. Springer Berlin Heidelberg 2005.
[13] M. Li, X. Du and S. Wang, "Learning ontology from Relational Database", in Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, August 2005,18-21.
[14] H. Ling, S. Zhou "Mapping Relational Databases into OWL Ontology," International Journal of Engineering and Technology, Vol. 5, No. 6, Dec 2013-Jan.
[15] 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.
[16] 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.
[17] K. Munir, M. Odeh, R. McClatchey, "Ontology-driven relational query formulation using the semantic and assertional capabilities of OWL-DL", Knowledge-Based Systems, vol.35, no.0, pp.144-159, 2012.
[18] C.Ramathilagam, M. L. Valarmathi, "A Framework for OWL DL based Ontology construction from Relational Database using Mapping and Semantic Rules," International Journal of Computer Applications (0975– 8887), Volume 76– No.17, August 2013.X
[19] M. Schneider, S. Rudolph2, G. Rudolph, "Modeling in OWL 2 without RestrictionsarXiv: 1212.2902 v3 [cs.AI] 28 Apr 2013.
[20] 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.
[21] M. K. Smith, C. Welty, D. L. McGuinness, OWL Web Ontology Language Guide (W3C Recommendation 10 February 2004) [EB/OL]. http://www.w3.org/TR/owl-features/, (last modified on 10 February 2004).
[22] L. Stojanovic, N. Stojanovic, R. Volz, "Migrating data-intensive web sites into the Semantic Web", In Proceedings of the 2002 ACM symposium on Applied computing (SAC '02), pp.1100-1107, ACM, 2002
[23] K. N. Vavliakis, T. K. Grollios, P. A. Mitkas, "RDOTE - Publishing Relational Databases into the Semantic Web", Journal of Systems and Software, vol.86, no.1, pp.89-99, 2013.
[24] W3C, OWL Working Group,, "Web Ontology Language (OWL)," http://www.w3.org/2004/OWL, 2004.
[25] W3C, OWL Working Group, "OWL 2 Web ontology language document overview. W3C Recommendation 27 October 2009," http://www.w3.org/TR/owl2-overview/.
[26] W3C, OWL Working Group, "OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. W3C Recommendation 11 December 2012," http://www.w3.org/TR/owl2-syntax/
[27] R. Zhou, C. Liu, and J. Li "On Holistic Constraint-Preserving Transformation from Relational Schema into XML Schema," 13th International Conference, DASFAA 2008, New Delhi, India, Volume 4947, March 2008