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 as the following 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. Section 2 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 scheme

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.

1. 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)

2. 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)

3. Many-to-many relation with additional attributes
   It is any binary relation BinRel(R, A, B) with additional attributes for the relation itself.

4. 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.
D. Mapping Attributes

In relational database, 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>
<thead>
<tr>
<th>Rule</th>
<th>Rule Definition</th>
<th>Equivalent into OWL 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R5</td>
<td>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.</td>
<td>Declaration( DataProperty( :AttributeType ) ) DataPropertyDomain( :attributeName :TableName ) DataPropertyRange( :attributeName :TableName )</td>
</tr>
<tr>
<td>R6</td>
<td>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).</td>
<td>Declaration( DataProperty( :hasAttributeType ) ) DataPropertyDomain( :hasAttributeName :TableName ) DataPropertyRange( :hasAttributeName :TableName ) HasKey( :TableName :hasAttributeName )</td>
</tr>
<tr>
<td>R7</td>
<td>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 &quot;FunctionalObjectProperty&quot;.</td>
<td>Declaration( ObjectProperty( :R_S ) ) ObjectPropertyDomain( :R_S :R ) ObjectPropertyRange( :R_S :S ) FunctionalObjectProperty( :R_S )</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Rule</th>
<th>Rule Definition</th>
<th>Equivalent into OWL 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R8</td>
<td>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.</td>
<td>SubClassOf( :R DataMaxCardinality( 1 :A xsd:TypeOfA ) [Optional] )</td>
</tr>
<tr>
<td>R9</td>
<td>For each attribute A in a relation R with a NOT NULL constraint we set DataMinCardinality restriction to 1.</td>
<td>SubClassOf( :R DataMinCardinality( 1 :A xsd:TypeOfA ) [Optional] )</td>
</tr>
<tr>
<td>R10</td>
<td>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).</td>
<td>SubClassOf( :R DataExactCardinality( 1 :A xsd:TypeOfA ) [Optional] )</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Rule</th>
<th>Rule Definition</th>
<th>Equivalent into OWL 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R12</td>
<td>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.</td>
<td>Declaration( ObjectProperty( :T1_T3 ) ) ObjectPropertyDomain( :T1_T3 :T1 ) ObjectPropertyRange( :T1_T3 :T3 ) TransitiveObjectProperty( :T1_T3 )</td>
</tr>
</tbody>
</table>

G. Mapping Cyclic Relations

For a set of relations R1...Rn (n ≥ 1) such that Ri is referenced by R(i+1) (1 ≤ i ≤ 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 ≥ 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 (∃ x, y ∈ R / FK(x, R, y, R) and PK(y, R)).

**Definition2.** A circular relation is defined as a set of relations R1 ... Rn (n ≥ 2), where Ri is referenced by Ri+1 (1 ≤ i ≤ 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

<table>
<thead>
<tr>
<th>Rule</th>
<th>Rule Definition</th>
<th>Equivalent into OWL 2</th>
</tr>
</thead>
</table>
| 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

<table>
<thead>
<tr>
<th>Rule</th>
<th>Rule Definition</th>
<th>Equivalent into OWL 2</th>
</tr>
</thead>
</table>
| 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.
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)
  Else if (isManyToManyRelation(Ti) = true) then
    MappingManyToManyRelation(Ti)
  Else
    MappingNormalRelation(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 (isFK(Ai)=true) then
    T" = getReferencedTable(Aj, Ti)
    If ((T' != T") and (isBinaryRelation(Ti)=false) then
      CheckTransitiveChain(Ti, T)
    End if
  Else if (isPK(Ai)=true) then
    MappingPK(Ai)
  Else if (isNotNullAttribute(Ai)=true) then
    MappingNotNullAttribute(Ai)
  End if
End loop
End

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)
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, T)
      If ((T' != T") and (isBinaryRelation(Ti)=false) then
        CheckTransitiveChain(Ti, T)
      End if
    End if
  End loop
End

Procedure CheckTransitiveChain(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 (isBinaryRelation(Ti)=false) then
      CreateTransitiveChain(Ti, T")
    End if
  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.
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 ResultList
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 )
    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 (FindInRefTableList( RelatedTable ) == false) then
            ResultList = FindCircularRelation( MainTable, RelatedTable, ResultList )
            If (ResultList != 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, Table T )
End
```

The MappingData() procedure is the following one.

```
Procedure MappingData( S, Table T )
Input: Schema S,
Begin
    For each Ti in S loop
        If (isBinaryRelation( Ti ) = false) then
            For each RS in T loop
                individualName = Concatenate( Ti, " ", getPk( Ti ) )
                individualType = Ti
                For each P in RS loop
                    PName = P.AttributeName
                    PType = P.AttributeType
                    If (isFK( PName ) = true) then
                        Ref = getReferencedTable( PName, Ti )
                        OPName = Concatenate( Ti, " ", Ref )
                        OPADestination = Concatenate( Ref, " ", PValue )
                        CreateObjectAssertion( OPName, individualName, OPADestination )
                    Else
                        CreateDataAssertion( PName, individualName, PValue, individualType )
                    End if
                End for
            End for
        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 for
        End if
    End for
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

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

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


