TRANSFORMATION RULES FOR BUILDING OWL ONTOLOGIES FROM RELATIONAL DATABASES

Mohammed Reda Chbihi Louhdi[1], Hicham Behja[2] and Said Ouatik El Alaoui[3]

[1]Laboratoire Informatique & Modélisation,
Faculty of sciences Dhar El Mehraz, Fez, Morocco
chbihi@gmail.com
[2]Ecole Nationale Supérieure d'Electricité et de Mécanique, Casablanca, Morocco
h_behja@yahoo.com
[3]Laboratoire Informatique & Modélisation,
Faculty of sciences Dhar El Mehraz, Fez, Morocco
s_ouatik@yahoo.com

ABSTRACT

Relational Databases (RDB) are used as the backend database by most of information systems. RDB encapsulate conceptual model and metadata needed in the ontology construction. Schema mapping is a technique that is used by all existing approaches for ontology building from RDB. However, most of those methods use poor transformation rules that prevent advanced database mining for building rich ontologies. In this paper, we propose transformation rules for building owl ontologies from RDBs. It allows transforming all possible cases in RDBs into ontological constructs. The proposed rules are enriched by analyzing stored data to detect disjointness and totalness constraints in hierarchies, and calculating the participation level of tables in n-ary relations. In addition, our technique is generic; hence it can be applied to any RDB. The proposed rules were evaluated using a normalized and open RDB. The obtained ontology is richer in terms of non- taxonomic relationships.

KEYWORDS

Semantic Web, Ontology Building, Relational Databases, Schema Mapping, Data Analysis.

1. INTRODUCTION

The Semantic Web [1], [2], [3] provides a common framework that allows data to be shared and reused across applications, enterprises, and community boundaries. The current web is dominated by unstructured and semi-structured documents. One of the objectives of the Semantic Web is to convert the current web into a "web of data", by encouraging the inclusion of semantic content in web pages and documents. Besides, the Semantic Web aims at making information on the Web machine processable and understandable, and therefore, facilitates interoperability between applications.

Sundarapandian et al. (Eds) : ICAITA, SAI, SEAS, CDKP, CMCA-2013
pp. 271–283, 2013. © CS & IT-CSCP 2013 DOI : 10.5121/csit.2013.3822

Tom Gruber[1] defined ontology as "*a formal and explicit specification of a shared conceptualization that refers to an abstract model of some phenomenon in the world that identifies the relevant concepts of that phenomenon*" [2]. In the context of database systems, ontology could be defined as a process of abstraction of data models that are similar to relational and hierarchical models, but which is supposed to model individuals' knowledge, attributes and relationships.

Ontologies are particularly specified in languages that make possible the abstraction of data structures and allow strategies implementation. Semantic Web is then expected to provide languages that can both express data and rules for reasoning about the data, and also to export rules from any existing knowledge-representation system onto the web.

Building ontologies from scratch is a very expensive and laborious task. A manual approach is a complex process and needs the supports of domain experts in knowledge acquisition as well, so it is time-consuming and error-prone. However, the automatic building of ontologies from existing information sources (text, databases, etc.) is relevant and unavoidable.

Generally, Relational Databases are used as the backend database by most of information systems. These databases are well-designed, encapsulate conceptual data models and hide a strong semantics that could be exploited in the process of ontology extraction.

All The existing methods for ontology engineering from relational databases use the schema mapping to transform the components of the conceptual data model or the physical model into ontology's concepts and relations [4].

In this work, we propose a technique which consists in enhancing transformation rules for building owl ontologies from relational databases. It combines schema mapping and data analysis techniques to detect disjointness and totalness in simple inheritance cases, and to compute the table's participation level in the n-ary relations. Our proposal covers all possible cases in databases and allows having richer ontologies.

The rest of this paper is organized as follows. Section 2 discusses related works in ontology engineering from relational databases. Section 3 describes the proposed transformation rules. Implementation and experimentations are presented in Section 4. Finally, Section 5 concludes this paper, and discusses the perspectives of this work.

2. RELATED WORKS

The realization of semantic web requires structuring of web data using domain ontologies. Extracting domain knowledge from database schemata can profitably support ontology development. There are many relational databases on the web that store important and useful information, which is a valuable source for ontology learning.

However, some existing methods [5], [7], [9], [10], [11] use the conceptual data model as a source of ontology learning, because it is semantically richer than the relational model. Unfortunately, in the most cases, the databases are available in a physical format (the corresponding conceptual data model is not available). In addition, the mapping operation (from the conceptual data model to the relational model) may create new tables and new attributes, that makes difficult the ontology populating task (since there are many differences between the components of the conceptual data model and those of the relational model).

---

[1] http://tomgruber.org/bio/short-bio.htm

Schema mapping technique is used by ontology building methods. It converts the relational database schema (or the ER Model) to an ontology by using a set of predefined transformation rules. Some works [12], [13], [14], [15], [16] map ontologies to relational databases schemata in order to maintain interoperability between them. The schema mapping is performed using mapping rules that update the ontology when the database is modified and vice-versa.

Note that most methods based on the schema mapping technique cannot handle some complex cases like multiple inheritance, many-to-many relations with attributes, and the n-ary relations. The multiple inheritance case was treated by [11] where authors reproduce the hierarchy found in the conceptual data model in the taxonomy of the ontology. Concerning the many-to-many relations with attributes, they were supported by the transformation rules of some methods [9], [10], [11], [17]. The n-ary relation is a difficult case, because only binary relations between classes can be represented through object properties in the ontology. However, some works [9], [10], [17], [18], [19] propose solutions to represent n-ary relations in OWL ontologies. In [10], [17], [18], [19], authors create a class for the bridge table related by two object properties mutually inverse. The method [9] uses AllValuesFrom restrictions to link the class corresponding to the bridge table, with the classes that correspond to the participating tables to the n-ary relation. This solution is more representative than the first one, because the existence of a record in the bridge table is conditioned by the existence of records in tables that participate to the n-ary relation.

The foreign key columns (or one-to-many relations) and the simple inheritance cases are processed by the most of existing methods. Furthermore, the most existing methods transform the simple attributes into Data Type Properties. Some of them [10], [11], [17], [18], [19] suggest to add restrictions to the attributes that have a constraint (Primary Key, NOT NULL or UNIQUE). In the Table I, we present the main methods and the different cases treated from the relational model.

Table 1. The main methods and the different cases captured from the relational model.

Methods	**One-to-many relation**	**Simple inheritance**	**Multiple inheritance**	**Many-to-many relation**	**Many-to-many relation with attributes**	**n-ary relations**
[5], [6]	X	X				
[7], [8], [20]	X			X		
[9], [17]	X	X		X	X	X
[10]	X			X	X	X
[11]	X	X	X	X	X	
[18], [19]	X	X		X		X
[21], [22], [23], [24], [25]	X	X		X		
Our method	X	X	X	X	X	X

In this work, we propose to use the relational model as a source for ontology learning. Unlike the methods mentioned previously, we propose exhaustive transformation rules that deal with most of existing cases in databases (table1). Moreover, we analyze the database records to recover some disappeared aspects during the mapping from the conceptual data model to the relational model (like disjointness and totalness in simple inheritance cases and the participating level of tables in n-ary relations).

3. TRANSFORMATION

Before applying transformation rules, we classify tables of the database schema into six categories according to their attributes. After that, we transform the tables of each category into ontological components by applying both mapping rules and data analysis. The latter finds disjointness and totalness in inheritance cases, and spots the participation level of tables in n-ary relations.

3.1. Classifying database tables

This step consists of classifying database tables into six categories according to the different cases. The table 2 shows the proposed classification. In this work, we suppose that the databases are at least in the third normal form.

Table 2. The different categories adopted for classifying the database tables.

Entity type	**Category**	**Features**
Strong entities	1	Tables containing only simple attributes without foreign keys constraint (Example : Tables *PERSON* and *PROGRAM* in Figure 1)
	2	Tables containing at least one foreign key (Example: Table *ACTIVITY* in Figure 1).
Weak entities	3	Tables whose entire primary key is also a foreign key referencing a single table. (Example: Tables *STUDENT* and *TEACHER* in Figure 1).
	4	Tables containing a composite primary key (two or more fields) which is also a foreign key whose fields are referencing exactly two tables (Example: Table *SUPERVISION* in Figure 1).
	5	Tables containing a composite primary key (two or more fields) which is also a foreign key whose fields are referencing more than two tables. Simple attributes are not duplicated in any of the referenced tables (Example: Table *OFFERED_COURSE* in Figure 1).
	6	Tables containing a composite primary key (two or more fields) which is also a foreign key whose fields are referencing more than two tables. Some simple attributes are duplicated in the referenced tables (Example: Table *PEDAGOGICAL_PROJECT* in Figure 1).

3.2. Rules

After classifying the database tables, we apply the appropriate transformation rules for each table's category (see table 2). In the rest of this paragraph, we will present the proposed transformation rules. All these rules are illustrated by examples using the database which is presented in Figure 1.

<u>Rule 1:</u> The tables that contain only simple columns (without foreign key constraint) are transformed into simple classes into the ontology (category 1). Example:

```
<owl:Class rdf:ID="PERSON"/>
<owl:Class rdf:ID="PROGRAM"/>
```

<u>Rule 2:</u> Tables of the second category are transformed into simple classes in the ontology. Each foreign key is mapped into two Object-Properties (mutually inverse). The first one has the class corresponding to current table as domain, and its range is the referenced table by the foreign key. The second one (inverse of the first Object-Property) is declared as inverse functional. Example:

```
<owl:Class rdf:ID="ACTIVITY"/>
<owl:ObjectProperty rdf:ID="activityHasProject">
      <rdfs:domain rdf:resource="#ACTIVITY" />
      <rdfs:range  rdf:resource="#PROJECT" />
</owl:ObjectProperty>
<owl:InverseFunctionalProperty rdf:ID="project'sActivity">
      <rdfs:domain rdf:resource="#PROJECT" />
      <rdfs:range  rdf:resource="#ACTIVITY"/>
      <owl:inverseOf rdf:resource="#activityHasProject" />
</owl:InverseFunctionalProperty>
```

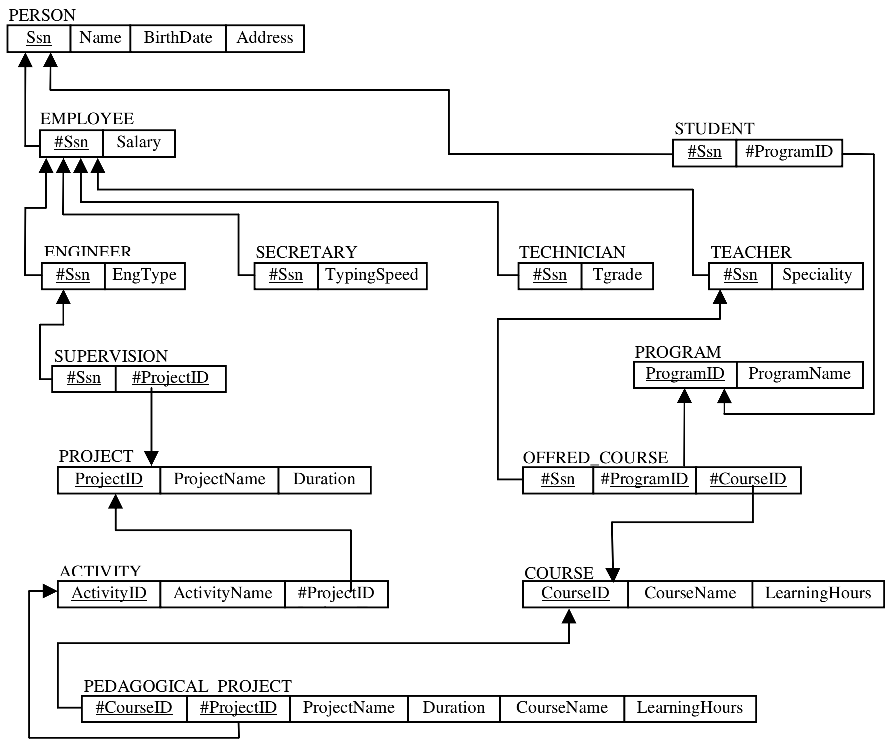

Figure 1. Example of a normalized database

<u>Rule 3:</u> we can identify simple inheritance relationships from tables of the third category. All tables in this category are sub-tables in hierarchies. Each sub-table is transformed into a class in the ontology and is declared as a subclass of the table referenced by the foreign key (which is also the primary key of each sub-table). Example:

```
<owl:Class rdf:ID="PERSON" />
<owl:Class rdf:ID="STUDENT">
      <rdfs:subClassOf rdf:resource="#PERSON" />
</owl:Class>
<owl:Class rdf:ID="TEACHER"/>
      <rdfs:subClassOf rdf:resource="#PERSON" />
</owl:Class>
```

After reproducing simple inheritance relations into the taxonomy of the ontology, we will identify disjointness and totalness constraints in those relations.

In a simple inheritance relation, disjointness means that an entity can be a member of at most one of the subclasses (that have the same level) of a hierarchy [26]. To identify the existence of disjointness between tables having the same level in a hierarchy, we propose the algorithm presented in Figure 2. For example, the following OWL code illustrates the disjointness constraint in a simple inheritance relation:

```
<owl:Class rdf:about="#TECHNICIAN">
      <rdfs:subClassOf rdf:resource="#EMPLOYEE"/>
      <owl:disjointWith rdf:resource="#SECRETARY"/>
      <owl:disjointWith rdf:resource="#TEACHER"/>
</owl:Class>
<owl:Class rdf:about="#SECRETARY">
      <rdfs:subClassOf rdf:resource="#EMPLOYEE"/>
      <owl:disjointWith rdf:resource="#TEACHER"/>
</owl:Class>
<owl:Class rdf:about="#TEACHER">
      <rdfs:subClassOf rdf:resource="#EMPLOYEE"/>
</owl:Class>
```

Disjointness

Input:
SC : List of the sub-tables of a simple inheritance relation

Output:
DL : two-dimensional array that will contain disjoint tables

Let **pk(T)** a function retrieving the primary key of a table **T**
Let **val(attr)** a function that retrieves the value of the attribute "**attr"** for the current record

```
Let N the size of the list SC
FOR i = 0 to N-1
  FOR j = i+1 to N-1
    FOR each record of the table SC[i]
      FOR each record of the table SC[j]
        IF val(pk(SC[i])) = val(pk(SC[j])) THEN
          BREAK
        ENDIF
      ENDFOR
    ENDFOR
    Add [SC[i], SC[j]] to the array DL
  ENDFOR
ENDFOR
```

Figure 2. Disjointness detection algorithm

Concerning the totalness, it specifies that every entity in the superclass must be a member of at least one subclass in the hierarchy [26]. To identify the existence of totalness in a hierarchy, we propose the algorithm presented in Figure 3. For example, the following OWL code illustrates the totalness constraint in a simple inheritance relation:

```
<owl:class rdf:ID="EMPLOYEE">
      <owl:unionOf rdf:parseType="Collection">
            <owl:class rdf:about="#SECRETARY" />
            <owl:class rdf:about="#TEACHER" />
            <owl:class rdf:about="#TECHNICIAN" />
      </owl:unionOf>
</owl:class>
```

In the case of existence of both a disjointness and totalness in a simple inheritance relationship, we combine the two previous proposals. The following example illustrates this situation:

```
<owl:Class rdf:about="#TECHNICIAN">
       <rdfs:subClassOf rdf:resource="#EMPLOYEE"/>
       <owl:disjointWith rdf:resource="#SECRETARY"/>
       <owl:disjointWith rdf:resource="#TEACHER"/>
</owl:Class>
<owl:Class rdf:about="#SECRETARY">
       <rdfs:subClassOf rdf:resource="#EMPLOYEE"/>
       <owl:disjointWith rdf:resource="#TEACHER"/>
</owl:Class>
<owl:Class rdf:about="#TEACHER">
       <rdfs:subClassOf rdf:resource="#EMPLOYEE"/>
</owl:Class>
<owl:class rdf:ID="EMPLOYEE">
       <owl:unionOf rdf :parseType="Collection">
              <owl:class rdf:about="#SECRETARY" />
              <owl:class rdf:about="#TEACHER" />
              <owl:class rdf:about="#TECHNICIAN" />
       </owl:unionOf>
</owl:class>
```

Totalness

Input:
- **ST :** the super-table
- **ssT :** a list of the sub-tables of **ST**

Output:
F : Boolean to flag the existence of totalness

Let **pk(T)** a function retrieving the primary key of a table **T**
Let **val(attr)** a function that retrieves the value of the attribute "**attr**" for the current record

```
Let N the size of the list ssT
F ← FALSE
FOR each table T of ssT
   FOR each record of ST
      FOR each record of T
         IF val(pk(T)) = val(pk(ST))THEN
            N ← N – 1
            BREAK
         ENDIF
      ENDFOR
   ENDFOR
ENDFOR
IF N = 0 THEN
   F ← TRUE
ENDIF
```

Figure 3. Totalness detection algorithm

Rule 4: The tables containing a composite primary key (two or more columns) which is also a foreign key whose fields are referencing exactly two tables (category 4), are mapped into two Object-Properties mutually inverse. Example:

```
<owl:ObjectProperty rdf:ID="hasProject">
       <rdfs:domain rdf:resource="#ENGINEER" />
       <rdfs:range  rdf:resource="#PROJECT" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasEngineer">
       <rdfs:domain rdf:resource="#PROJECT" />
```

```
      <rdfs:range  rdf:resource="#ENGINEER" />
      <owl:inverseOf rdf:resource="#hasProject"/>
</owl:ObjectProperty>
```

If a table of category 4 contains simple columns (table resulted from a many-to-many relation with attributes), we apply rules 4 and 5 to this table.

Rule 5: the tables of the category 5 are resulting from n-ary relations. Their primary keys are composed by several foreign keys (more than two) referencing the participating tables to the relation.

OWL does not support n-ary relations. To represent this type of relations, W3C proposed two solutions [27]. The first one is to create an individual that represents the relation instance itself, with links from the subject of the relation to this instance and with links from this instance to all participants that represent additional information (see Figure 4).

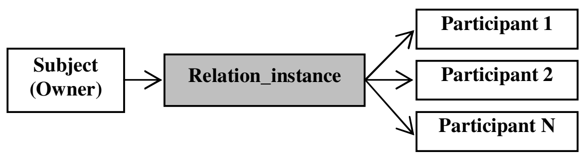

Figure 4. First solution of the W3C to represent n-ary relations

The second solution proposed by [27] is used when the n-ary relationship links individuals that play different roles in a structure without any single individual standing out as the subject or the "owner" of the relation. In this case, we create an individual to represent the relation instance with links to all participants (Figure 5).

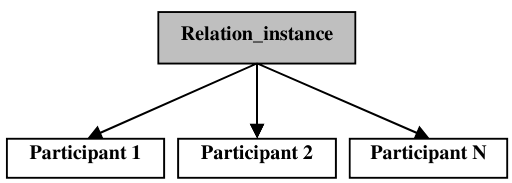

Figure 5. Second solution of the W3C to represent n-ary relations

In the case of ontology automatic generation, the first solution cannot be applied, because we should choose a subject for the relationship. Therefore, we will adopt the second solution in our method.

To represent n-ary relations from the relational model, we create a class corresponding to the bridge table related to the classes that correspond to the participating tables to the n-ary relation by OWL restrictions (*allValuesFrom* or *someValuesFrom*). These restrictions are depending on the participation level of tables in the relation. In Figure 6, we present an example of transforming the n-ary relation *OFFRED_COURSE* in the example given in the Figure 1.

To define the participation level of each table to the n-ary relation, we check if all records of the participating tables are referenced in the bridge table. If so, we use an *allValuesFrom* restriction, else a *someValuesFrom* restriction is used.

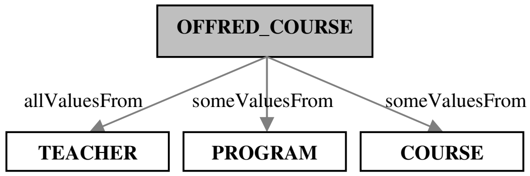

Figure 6. Example of transforming a n-ary relation

To illustrate this solution, we present below the OWL code corresponding to the n-ary relation *OFFRED_COURSE*:

```
<owl:Class rdf:ID="OFFRED_COURSE">
      <rdfs:subClassOf>
            <owl:Restriction>
                  <owl:someValuesFrom>
                        <owl:Class rdf:about="#COURSE"/>
                  </owl:someValuesFrom>
                  <owl:onProperty>
                        <owl:ObjectProperty rdf:about="hasCourse"/>
                  </owl:onProperty>
            </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
            <owl:Restriction>
                  <owl:someValuesFrom>
                        <owl:Class rdf:about="#PROGRAM"/>
                  </owl:someValuesFrom>
                  <owl:onProperty>
                        <owl:ObjectProperty rdf:about="hasProgram"/>
                  </owl:onProperty>
            </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
            <owl:Restriction>
                  <owl:allValuesFrom>
                        <owl:Class rdf:about="#TEACHER"/>
                  </owl:allValuesFrom>
                  <owl:onProperty>
                        <owl:ObjectProperty rdf:about="hasTeacher"/>
                  </owl:onProperty>
            </owl:Restriction>
      </rdfs:subClassOf>
</owl:Class>
```

Rule 6: in the conceptual data model, the tables of the category 6 are subclasses in more than one class/subclass relationship (multiple inheritance). In the relational model, these tables can be confused with the bridge tables of n-ary relations (category 5), since they are weak entities whose primary key consists of two or more foreign keys referencing two or more tables. To distinguish between both categories (5 and 6), we supposed that tables of category 6 must contain in addition to the primary key, inherited attributes (during the mapping process) belonging to super-tables.

To map the multiple inheritance case, we reproduce the same hierarchy in the taxonomy of the ontology. Each sub-table (of category 6) is transformed into a subclass of the classes corresponding to the tables referenced by the foreign keys of the sub-table. For example, the OWL code for the *PEDAGOGICAL_PROJECT* (Figure 1) is as follows:

```
<owl:Class rdf:ID="COURSE" />
<owl:Class rdf:ID="PROJECT"/>
<owl:Class rdf:ID="PEDAGOGICAL_PROJECT">
	<rdfs:subClassOf rdf:resource="#PROJECT" />
	<rdfs:subClassOf rdf:resource="#COURSE" />
</owl:Class>
```

Rule 7: Concerning the transformation of the columns (without foreign key constraint), we create for each attribute a dataType property for which the domain is the class corresponding to the table containing this column and the range is the type in XML schema. Example:

```
<owl:DatatypeProperty rdf:ID="Name">
	<rdfs:domain rdf:resource="#PERSON"/>
	<rdf:range rdf:resource="&xsd;String"/>
</owl:DatatypeProperty>
```

For attributes with special constraints such as NOT NULL, UNIQUE and Primary Key, we propose to treat them as follows:

NOT NULL: add the MinCardinality restriction to the Datatype Property with the value 1,

```
<owl:Class rdf:ID="PERSON">
	<rdfs:subClassOf>
		<owl:Restriction>
			<owl:onProperty rdf:resource="#Name" />
			<owl:minCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
		</owl:Restriction>
	</rdfs:subClassOf>
</owl:Class>
```

UNIQUE: declare the Datatype property as inverse functional,

```
<owl:InverseFunctionalProperty rdf:ID="Program_name">
	<rdfs:domain rdf:resource="#PROGRAM"/>
	<rdf:range rdf:resource="&xsd;String"/>
</owl:InverseFunctionalProperty>
```

Primary Key: add the MinCardinality restriction to the Datatype Property with the value 1, and declare it as inverse functional,

```
<owl:InverseFunctionalProperty rdf:ID="Ssn">
	<rdfs:domain rdf:resource="#PERSON"/>
	<rdf:range rdf:resource="&xsd;String"/>
</owl:InverseFunctionalProperty>
<owl:Class rdf:ID="PERSON">
	<rdfs:subClassOf>
		<owl:Restriction>
			<owl:onProperty rdf:resource="#Ssn" />
			<owl:minCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
		</owl:Restriction>
	</rdfs:subClassOf>
</owl:Class>
```

The table IV summarizes the proposed transformation rules.

Table 3. Summary of the proposed transformation rules.

Rule	Case	OWL Component
1	Strong entity	Class
2	Foreign Key	Two Object Properties mutually inverse 1. The First one : - Domain: Class corresponding to the table containing the column, - Range : Class corresponding to the referenced table, 2. The Second one is the inverse of the first, and it is declared as inverse functional
3 & 6	Simple and Multiple inheritance	Reproducing inheritance relations into the taxonomy of the ontology
4	Many-to-Many relation	Two Object Properties mutually inverse that rely the classes corresponding to the participant tables of the relation
5	N-ary Relation	A class corresponding to bridge table, related to the classes that correspond to the participating tables with "is-a" relation restricted with the OWL restriction "AllValuesFrom" or "SomeValuesFrom" according to the participating level of each table of the relation
4 & 5	Many-to-Many relation with attributes	Combination of the above two cases
7	Simple column	Data Type Property - Domain : Class corresponding to the table containing the column, - Range : The column type expressed with XML Schema,
	Column with UNIQUE constraint	Inverse Functional Property - Domain : Class corresponding to the table containing the column, - Range: The column type expressed with XML Schema,
	Column with NOT NULL constraint	Data Type Property - Domain : Class corresponding to the table containing the column, - Range : The column type expressed with XML Schema, - Minimal cardinality = 1
	Primary Key	Inverse Functional Property - Domain : Class corresponding to the table containing the column, - Range : The column type expressed with XML Schema, - Minimal cardinality = 1

4. EXPERIMENTAL RESULTS

To evaluate the efficiency of the proposed transformation rules, we implemented the proposal with Java and the Jena API (Java framework for building Semantic Web applications) for automatically OWL ontology building from relational database. Having a friendly-user interface, our system allows building OWL file that contains the definition of the extracted ontology from MySql database.

We conducted several experiments using a normalized relational database (SAKILA available on the official MySql website) containing different cases discussed previously. The metadata of this database is presented in the table 4.

Table 4. Metadata of SAKILA database.

Metadata		Count
Entities	Strong	14
	Weak	2
Columns	Primary Key (PK) constraint	14
	Foreign Key (FK) constraint	16
	PK and FK constraints	4
	Others	53

Using the proposed transformation rules to map SAKILA database into ontology, the obtained numbers of concepts, Object Properties and Data Type Properties are respectively 16, 42 and 67. All the existing cases in the SAKILA database were successfully transformed into ontological components.

5. CONCLUSION AND FURTHER WORKS

In this paper, we have proposed a technique which consists in a set of transformation rules for building OWL ontologies from relational databases. The schema mapping uses the transformation rules to transform the components of the physical model into ontology's components. The data analysis is used to recover some disappeared aspects during mapping conceptual data model to the relational model (like disjointness and totalness in simple inheritance cases and the participating level of tables in n-ary relations).

We have tested our proposal using the SAKILA database. The obtained results are satisfactory compared to other methods in terms of the number of the treated concepts. Moreover, our method covers all possible cases in databases. The generated ontologies have richer non-taxonomic relations.

A major direction for improvement could be to add a reverse engineering phase before applying the transformation rules in order to detect generalization and specialization inheritance cases. This process will allow us to recover disappeared tables during mapping conceptual data model to the relational model. Furthermore, integrating a reverse engineering phase will make the built ontologies richer in terms of taxonomic relations. Finally, we also suggest testing our technique on a second database to show its robustness and its genericity.

REFERENCES

[1] W3C, the semantic web activity, Available at : http://www.w3.org/2001/sw/

[2] T. Gruber, Ontology, In Ling Liu & M. Tamer Özsu (Ed.), the Encyclopedia of Database Systems (Springer-Verlag, 2009, 1963-1965).

[3] Tim Berners-Lee, James Hendler & Ora Lassila, (2001) “The Semantic Web”, *Scientific American*, Vol. 284, pp. 34-43.

[4] Alexander Maedche & Steffen Staab, (2005) “Ontology Learning for The Semantic Web”, *IEEE Intelligent Systems*, Vol. 16, No. 2, pp. 72-79.

[5] Chen He-ping, He Lu & Chen Bin, (2008) “Research and implementation of ontology automatic construction based on relational database”, *Proceedings of the International Conference on Computer Science and Software Engineering,* pp. 1078-1081.

[6] Heru Agus, Su-Cheng Haw & Ziyad.T. Abdul-Mehdi, (2011) “Ontology extraction from relational database: Concept hierarchy as background knowledge”, *Knowledge-Based Systems*, Vol. 24, No 3, pp. 457-464,

[7] Justas Trinkunas & Olegas Vasilecas, (2007) “Building ontologies from relational databases using reverse engineering methods”, *Proceedings of the international conference on Computer systems and technologies.*

[8] Farid Cerbah, (2008) "Mining the Content of Relational Databases to Learn Ontologies with Deeper Taxonomies", *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology,* pp. 553- 557.

[9] Zhuoming Xu, Xiao Cao, Yisheng Dong & Wenping Su, (2004) "Formal Approach and Automated Tool for Translating ER Schemata into OWL Ontologies", *Advances in Knowledge Discovery and Data Mining*, pp. 464-475.

[10] Igor Myroshnichenko & Marguerite C. Murphy, (2009) "Mapping ER Schemas to OWL Ontologies", *Proceedings of the IEEE International Conference on Semantic Computing,* pp. 324-329.

[11] Sujatha R. Upadhyaya & P. Sreenivasa Kumar, (2005) "ERONTO: A Tool for Extracting Ontologies from Extended ER Diagrams", *Proceedings of the 20th ACM Symposium on Applied Computing,* pp. 666-670.

[12] Shihan Yang & Jinzhao Wu, (2010) "Mapping Relational Databases into Ontologies through a Graph-based Formal Model", *Proceedings of the Sixth International Conference on Semantics Knowledge and Grid*, pp. 219-226.

[13] Christian Bizer, (2003) "D2R MAP - a database to RDF mapping language", *Proceedings of the 12th International World Wide Web Conference*.

[14] Nikolaos Konstantinou , Dimitrios-emmanuel Spanos , Michael Chalas , Emmanuel Solidakis & Nikolas Mitrou, (2006) "VisAVis: An Approach to an Intermediate Layer between Ontologies and Relational Database Contents", *Proceedings of Workshops and Doctoral Consortium, The 18th International Conference on Advanced Information Systems Engineering - Trusted Information Systems.*

[15] Shihan Yang, Ying Zheng & Xuehui Yang, (2010) "Semi-automatically building ontologies from relational databases", *Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology*, pp. 150 - 154.

[16] Jesús Barrasa , Óscar Corcho & Asunción Gómez-pérez, (2004) "R2O, an Extensible and Semantically based Database-to-Ontology Mapping Language", *Proceedings of the 2nd Workshop on Semantic Web and Databases*, pp. 1069-1070.

[17] Irina Astrova , Nahum Korda & Ahto Kalja, (2007) "Rule-Based Transformation of SQL Relational Databases to OWL Ontologies", *Proceedings of the 2nd International Conference on Metadata & Semantics Research*, pp. 415-424.

[18] Man Li, Xiao-Yong Du & Shan Wang, (2005) "Learning ontology from relational database", *Proceedings of the International Conference on Machine Learning and Cybernetics*, pp. 3410-3415.

[19] Zdenka Telnarova, (2010) "Relational database as a source of ontology creation", *Proceedings of the International Multiconference on Computer Science and Information Technology*, pp. 135-139.

[20] Kobra etminani , Mohsen Kahani & Yanehsari N.R, (2009) "Building ontologies from relational databases", *Proceedings of the First International Conference on Networked Digital Technologies*, pp. 555-557.

[21] Changjun Hu, Huayu Li, Xiaoming Zhang & Chongchong Zhao, (2008) "Research and Implementation of Domain-Specific Ontology Building from Relational Database", *Proceedings of the Third ChinaGrid Annual Conference,* pp. 289 - 293.

[22] Ahmed Waqas, Aslam Muhammad Ahtisham, Shen Jun & Yong Jianming, (2011) "A light weight approach for ontology generation and change synchronization between ontologies and source relational databases", *Proceedings of the 15th International Conference on Computer Supported Cooperative Work in Design*, pp. 208-214.

[23] Nadine Cullot, Raji Ghawi & Kokou Yétongnon, (2007) "DB2OWL: A Tool for Automatic Database-to-Ontology Mapping", *Proceedings of the 15th Italian Symposium on Advanced Database Systems*, pp. 491-494.

[24] Irina Astrova & Ahto Kalja, (2006) "Towards the Semantic Web: Extracting OWL Ontologies from SQL Relational Schemata", *Proceedings of the 2nd International Conference on Metadata & Semantics Research*, pp. 62-66.

[25] Guohua Shen, Zhiqiu Huang, Xiaodong Zhu & Xiaofei Zhao, (2006) "Research on the Rules of Mapping from Relational Model to OWL", *Proceedings of the OWLED*06 Workshop on OWL: Experiences and Directions*.

[26] Ramez Elmasri & Shamkant Navathe, (2011) Fundamentals of Database Systems sixth Edition Pearson Education.

[27] W3C Working Group, Defining n-ary Relations on the Semantic Web, 12 April 2006. Available at : http://www.w3.org/TR/swbp-n-aryRelations/