International Journal of Physical Sciences Vol. 7(13), pp. 2085 - 2092, 23 March, 2012
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS12.072
ISSN 1992 - 1950 ©2012 Academic Journals

Full Length Research Paper

A novel method for improving the efficiency of automatic construction of ontology from a relational database

Saeed M. Sedighi[1]* and Reza Javidan[2]

[1]Department of Computer Engineering, Zanjan Branch, Islamic Azad University, Zanjan, Iran.
[2]Department of Computer Engineering, Islamic Azad University-Beyza Branch, Fars, Iran.

Accepted 15 March, 2012

With the development of the semantic web, ontology is playing an increasingly important role in many research areas such as semantic interoperability and knowledge base. However, constructing ontology manually is complicated and needs the supports of domain experts in knowledge acquisition as well, so it is time-consuming, error-prone and tedious-work. Learning ontology from existing resources is a good solution. We can use relational database for building ontology, because relational database is widely used for storing data. This paper proposes an approach of learning ontology web language (OWL) from data in relational database. Compared with existing methods, our approach can acquire ontology from relational database automatically. In addition, our proposed method, unlike other existing methods, all types of relationships between tables are considered. The proposed method is implemented using Jena and MySQL and is applied on a sample relational database (RDB). The resulting ontology was shown as an OWL file. The evaluation of the generated ontology will use FaCT + + and Pellet.

Key words: Ontology building, OWL, Relational database, Semantic web, mapping rules.

INTRODUCTION

Storage and dissemination of information on the web is done with ease, but this type of storage on the web, have many problems for later retrieval and use of information. Semantic web (Berners-Lee et al., 2001) is a key solution to solve this problem, which aims to share information on the web as more intelligent, so that it will be understandable to both humans and machines. Ontology is the core of the semantic web. Ontology (Chang-rui et al., 2006) is a modeling tool of conceptualization in semantics and knowledge level, which provides explicit description and modeling methods for information and knowledge. At present, most of the ontologies are made manually. Constructing ontology manually is complicated and needs the supports of domain experts in knowledge acquisition as well, so it is time-consuming, error-prone and tedious-work. Learning ontology from existing resources is a good solution. Existing resources may contain different types of data structures (Cullot et al., 2007): data may be structured as databases, semi-structured as XML documents and/or non-structured as web pages or other type of documents. In this paper, we focused only on the construction of local ontology from a relational database and propose an approach of learning ontology web language (OWL) from data in relational database. Compared with the existing methods, our approach can acquire ontology from relational database automatically. In addition, our proposed method, unlike other existing methods, all types of relationships between tables are considered.

RELATED WORK

At least, two issues exist in the field of relational databases (RDBs) and ontologies:

1. Mapping between RDB and ontology.

*Corresponding author. E-mail: Saeed.mohamadsedighi@gmail.com.

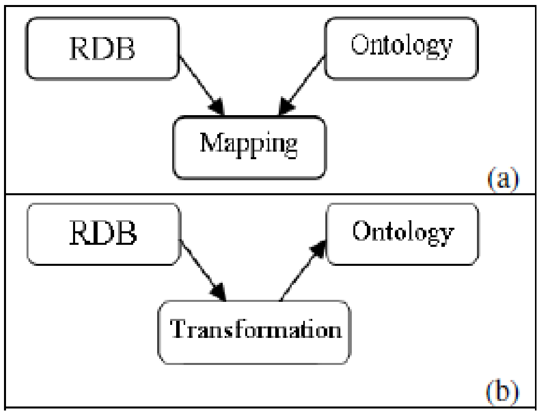

Figure 1.Mapping versus transformation.

2. Building ontology from RDB.

These issues are quite different. As shown in Figure 1a, in mapping, both the ontology and the corresponding RDB are assumed to exist. One should map RDB concepts to the corresponding ones in the ontology. But in building (Figure 1b), no constructed ontology exists. Some methods are applied to transform each concept in RDB to ontology.

There are several approaches for construction of ontology from a relational database. Zhou and Meng (2010) presents a prototype tool for translating relational databases schema into ontology. The key feature of the tool is that it can directly and automatically translate relational database schema into ontology. But in this tool, database structure is assumed to be simple and transformed only the structure and not the data. Zhang and Li. (2011) proposed a tool named ontology automatic generation system based on relational database (OGSRD), for automatic ontology building using the relational database resources to improve the efficiency. This tool Firstly, mapping analysis of ontology, and database is done. Secondly, construction rules of ontology elements based on relational database, which are used to generate ontology concepts, properties, axioms and instances are put forward. Thirdly, OGSRD is designed and implemented. Finally, the practical experiments prove the method and system feasibility. Hu et al. (2008) proposed three mapping rules from relational database schema to ontology class and property. Based on these rules, an initial ontology of material science that can be modified later is built and ontology instances can be generated. He-ping et al. (2008) proposed technology of ontology automatic construction based on relational databases. Then, an ontology generator named OWLFROMDB was implemented, which can automatically convert a relational database to OWL ontology. The technology strategy includes four steps: (1) extract entity relationship (ER) model from relational database by reverse engineering tool or querying for database system tables; (2) analyze the ER model from step1, transfer ER model to OWL ontology model by schema conversion mechanism; (3) transfer database data to ontology instances in batch by data conversion mechanism; (4) evaluate and verify the integral OWL file by existing ontology engineering tool and output the object OWL ontology. Astrova et al. (2008), proposed a novel approach to extract ontologies from relational schemata. A relational schema is represented in SQL, while ontology is represented in OWL. This approach maps all constructs of the relational schema, with the exception of stored procedures, triggers, default constraints and check constraints that do not use enumeration. In addition, a number of other existing approaches are of these studies (Zhou ., 2011; Yang et al., 2010; Ge et al., 2010; Alalwan et al., 2009; Brank et al., 2005; Fortuna et al., 2006; Jiang et al., 2010; Xu et al., 2011; Secer et al., 2011; Ochoa et al., 2011; Secer et al., 2011). However, all approaches suffer from at least one of the following problems:

1) They are not implemented.
2) They transform only the structure and not the data.
3) They are semi-automatic and require much user interaction.
4) Transformed structures are so simple: e.g. primary keys assumed to be single-column, foreign keys assumed to be single-column and relationship assumed only to be 1:1.

To solve the aforementioned problems, a novel approach is proposed. In this paper, we designed and implemented tool to be more efficient than existing methods. Our proposed method is fully automatic. It means no user interaction is needed. Also, in addition to structure conversion mechanism, it has data conversion mechanism. So, all types of relationships between tables is considered (As 1:1, 1: N and M:N).

PROPOSED CONSTRUCTION OF ONTOLOGY APPROACH

Databases include so full conceptual models and information resources that can be taken as the conceptualization repository of ontology. Through analysis, the formal corresponding relationships between relational databases and OWL ontologies are as follows: a relational database contains several tables, a table contains several fields and records are the collection of fields value; on the other hand, an OWL ontology contains several classes, a class contains several properties and instances are the collection of property value. The formal corresponding relationships between tables, fields and records in relational databases and classes, properties and instances in OWL ontologies make it possible to convert one schema to another. The

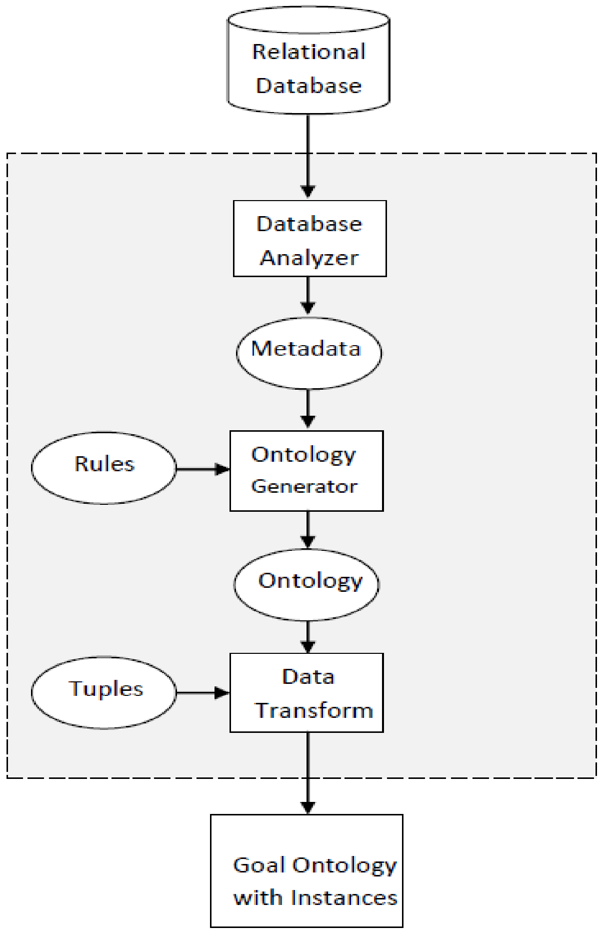

Figure 2. Proposed method architecture.

main purpose of the proposed approach is to try to make use of the existing relational databases to generate ontology automatically, hence reducing the manual tedious work, saving developing time and improving efficiency of ontology. The proposed method architecture (Figure 2) includes the following four steps:

1) Extract metadata from relational database by Java Database Connectivity (JDBC) component.
2) Analyze the metadata from step1, transfer conceptual model to OWL ontology model by rules library.
3) Transfer database data to ontology instances by data conversion mechanism.
4) Evaluate and verify the obtained ontology by FaCT++ and Pellet components.

Defining the mapping rules

Definition 1

A relational database schema is a tuple D = (N, col, datatype, pk, fk, assoc, subof), where,

- N is a finite name set partitioned into: (1) a subset ET of entity table names; each entity table contains rows of instance data describing entities in the real world, (2) a subset RT of relationship table names; each relationship table contains rows of instance data describing the relationships between entities, and (3) a subset DT of datatype names; each datatype is a predefined relational database management system (RDBMS) datatype, specifying a value range of the relevant instance data. Furthermore, for each t ∈ ET∪RT, there is a finite nonempty set col(t) of column names each c ∈ col(t) has an associated datatype, denoted datatype(c) ∈DT.
- For each t ∈ ET ∪ RT there is exactly one primary key pk(t) whose values uniquely determine each row of the instance data in t, where either pk(t) ∈ col(t) (in this case pk(t) is a single-attribute key and t is an entity table) or pk(t) ∈col(t) (in this case pk(t) is a composite key with more than one attribute, and t is a relationship tables).
- For each t є ET∪RT, there are n (n>0) foreign keys fk (t,r) where r ∊ET;each fk(t,r) ∊ col(t) references the values of the single-attribute primary key of entity table r , and it holds that value(fk(t, r)) ⊆value(pk(r)) ∪{null} where value(*) denotes the value range of '*' and pk(r) ∊col(r) .
- assoc ⊆ RT × ET × ET is a ternary relation over RT and ET that models an association relation between one relationship table and two entity tables (we assume without loss of generality that n-nary (n ≥ 3) relationships do not exist in the ER schema). For some t∈RT and r, s∈ET , assoc(t, r, s) is satisfied if ∃fk(t, r), fk(t, s)∈col(t) such that pk(t) = {fk(t, r), fk(t, s)}⊆ col(t) .
- subof ⊆ ET × ET is a binary relation over ET that models an inheritance relation between two entity tables. For some t, r∈ET , subof(t, r) is satisfied iif ∃fk(t, r)∈col(t) such that either fk(t, r) = pk(t) (single inheritance) or fk(t, r)∈pk(t) (multiple inheritance). Here t is a subentity table, r is a superentity table, and all the related tables form a generalization hierarchy of entity tables.

Definition 2

An OWL ontology is a tuple O = (ID, Axiom), where

- ID is a finite OWL identifier set partitioned into: (1) a subset CID of class identifiers including userdefined identifiers plus two predefined classes (OWL:Thing and OWL:Nothing); classes are either entity classes describing entities or relationship classes describing the relationships between entities; (2) a subset DRID of data range identifiers; data range identifiers are predefined XML schema datatypes, such as xsd:integer; (3) a subset OPID of object property identifiers, object properties link individuals (that is, entities) to individuals; and (4) a subset DPID of datatype property identifiers, datatype

Table 1. Data type transformation.

MYSQL data type	Jena data type
Numeric type	
Tinyint	xByte
Smallint	Xint
Mediumint	Xint
Int	Xint
Integer	integer
Bigint	Xlong
Float(x)	Xfloat
Float	Xfloat
Double	Xdouble
Double precision	Xdouble
Real	Xdouble
Decimal(m,d)	Decimal
Numeric(m,d)	
Data and time type	
Date	Date
Datetime	Datetime
Timestamp	
Time	Time
Year	Gyear
String type	
Char(m)	xstring
Varchar(m)	Xstring
Tinyblob, tinytext	Xstring
Blob, text	Xstring
Mediumblob, mediumtext	Xstring
Longblob, longtext	Xstring
Enum('v1', 'v2', …)	Xstring
Set('v1', 'v2', …)	Xstring

properties link individuals to data values.
- Axiom is a finite OWL axiom set partitioned into a subset of class axioms and a subset of property axioms; each axiom is formed by applying OWL constructs to the identifiers or descriptions that are the basic building blocks of a class axiom and describe the class either by a class identifier or by specifying the extension of an unnamed anonymous class via the construct restriction.

Mapping rules

A relational database consists of tables, columns, rows, datatypes, primary keys, foreign keys, etc. Similarity, ontology consists of classes, object, properties, datatype properties, individuals, etc. An approach is designed that transform RBD structure and data to the equivalent from in ontology. Each part is transformed from RDB to the corresponding part in the ontology. Here, the transformation process is explained.

Tables to classes

Each table is transformed into a class (concept) in the ontology. Tables in RDB can have relationships to other tables through freeing keys that are discussed in subsequently in the relationship. But, we can divide tables into the following groups according to their relationships:

1. Tables that have no relationship
2. Tables that have 1: relationship
3. Tables that have 1: N relationship
4. Tables that have M: N relationship

The first group is the simplest one and requires no additional transformation, but the others need some additional attention. Of course, we cannot have M:N relationship in RDBs directly, but it can be implemented through adding a third table that breaks M:N into two 1:N relationship. It can be recognized through foreign keys relations to other tables. If primary key of a table is multi-column and at least two subset of it are foreign keys to two other tables, then this table is the third table and the two other tables had M:N relationship that has been broken to two 1:N relationships.

Columns to properties

Each table in RDB consists of columns that could be primary key, foreign key or a simple column. So, the columns are classified to three groups: simple columns, primary keys and foreign keys. Each column, in each of the aforementioned groups, could have the restriction that should not be null. To transform this restriction, we should create min cardinality restriction with the value of 1 in the ontology.

The other constraint on columns is the uniqueness that forces one column not to have two rows with the same value in this column. Inverse functional property in ontology can be a good transformation for this constraint. Simple columns in RDBs are columns that contain a data item with a determined datatype. They are transformed to datatype properties in ontology. Each datatype property should have a domain, in which the class it belongs, and a range, from which class it takes the value. The domain of each datatype property is the class (table) it belongs to. For the range, OWL has XSD datatypes. Each datatype in RDB is transformed to XSD, as shown in Table 1.

A primary key is a column (or columns) that are unique and not null. Both inverse functional property and min cardinality restriction should be added. All previous

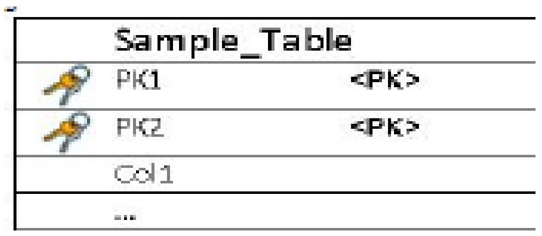

Figure 3. An example of multi-column primary key.

```
(1)  <owl:Class rdf:about=NS + "#sample_table"/>
(2)  <owl:Class rdf:about=NS +"#sample_table_pk_class"/>
(3)  <owl: InverseFunctionalProperty rdf:about=NS+ "#sample_table_pkOP">
(4)  <rdfs:range rdf:resource=NS + "#sample_table_pkClass"/>
(5)  <rdfs:domain rdf:resource=NS + "#sample_table"/>
(6)  </owl: InverseFunctionalProperty>
(7)  <owl:DatatypeProperty rdf:about=NS + "#sample_table_PK1">
(8)  <rdfs:domain rdf:resource=NS + "#sample_table_pkClass"/>
(9)  <rdfs:range    rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
(10)  </owl:DatatypeProperty>
(11)  <owl:DatatypeProperty rdf:about=NS + "#sample_table_PK2">
(12) <rdfs:domain rdf:resource=NS + "#sample_table_pkClass"/>
(13) <rdfs:range     rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
(14) </owl:DatatypeProperty>
(15) <owl:Restriction rdf:about=NS + "#sample_table_pkMinRes">
(16) <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1
(17) </owl:minCardinality>
(18) <owl:onProperty rdf:resource=NS + "#sample_table_pkOP"/>
 (19) </owl:Restriction>
```

Figure 4. The transferred ontology for multi-column primary key.

researches concentrated on single-column primary keys. If primary key becomes multi-column, then not null is applied to each column of the key, but uniqueness is not mandatory for each column and should be applied to the set of columns. To overcome this, a class for primary key (e.g. pk_class) is considered. Each column of primary key has pk_class for its domain. To relate pk_class to the original class, an object property is defined that relates two classes in ontology. This object property has the original class in its domain and pk_class in its range. To apply uniqueness on primary key, the created object property should become an inverse functional property. Figure 3 shows an example for two column primary key. Figure 4 illustrates the transferred ontology. NS in Figure 4 is the name of the namespace for the ontology. As displayed in Figure 4, sample_table is transformed to a class (line 1) and a class named sample_table_pk_class is created for its primary key (line 2). To relate these two classes, an object property named sample_table_pkOP is added that its domain is sample_table and its range is sample_table_pk_class. To apply uniqueness on the primary key, the object property is determined as an inverse functional property (lines 3 to 6). Lines 7 to 14 show how to add PK1 and PK2 as datatype properties. Both of them have sample_table_pk_class as their domain and according to their datatypes in the related table, they have corresponding datatype in their range through Table 1. At last in lines 15 to 19, min cardinality restriction is applied to guarantee not-nullable for the primary key.

A foreign key is a column (or columns) that relate two tables. As described in defining the mapping rules, RDBs can contain three kinds of relationships: 1:1, 1:N and M:N (tacitly). Here, an approach is presented to recognize type of relationship and convert it into ontology.

Relationships

1:1 Relationships: If two (or more) tables are related to each other through their primary keys, it could be 1:1 relationship. If both the primary keys of the related tables have the same number of columns, certainly it is 1:1 relationship. It means that the primary key of a table is also its foreign key. So, we follow the rules of the primary key, but each column is an object property that has the source table as its range and destination table as its domain. Figure 5 shows a sample of 1:1 relationship.

1:N Relationships: When two tables are related to each other and not matched to 1:1 relationship, then it is 1:N relationship. It contains the following cases:

1. When foreign key is a simple column (not a primary key). Figure 6 shows a sample example. So, an object property is added to the ontology that its domain is the destination table and its range is the source table.
2. When foreign key is a part (not the whole) of primary key, but other parts of primary key are not foreign keys (as shown in Figure 7). Like the previous case, an object property is added with the same range, but its domain is the pk_class that is inserted in the ontology for the primary key of the destination table, because this column(s) is also a part of a primary key that has some features.

It should be mentioned that 1:N relationships and 1:N relationships that are the result of M:N relationships divisions are discriminated here.

M:N Relationships: If tables are connected as follows, then M:N relationship occurred: if the primary key of a table consists of more than one foreign key to other

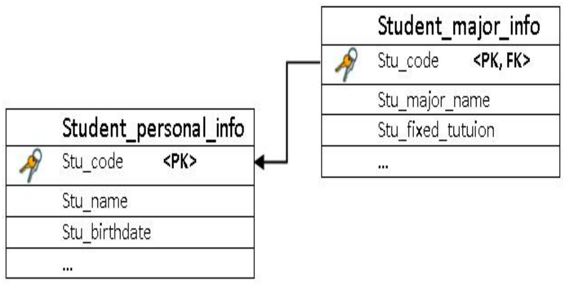

Figure 5. A sample of 1:1 relationship.

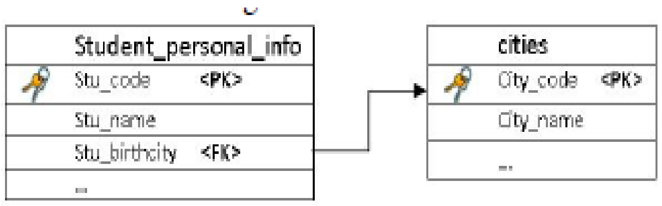

Figure 6. A sample of 1:N relationship (the first case)

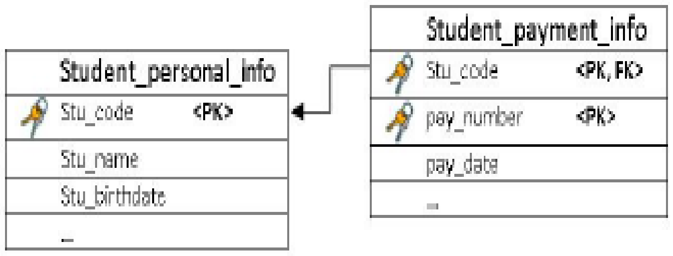

Figure 7. A sample of 1:N relationship (the second case).

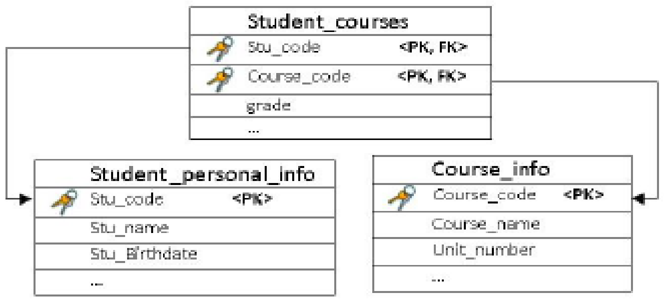

Figure 8. A sample of M:N relationship that is divided into two 1:N relationships.

tables, then this table is the one which must be added since M:N relationships is not supported explicitly in RDBs (as described in tables to classes). Figure 8 illustrates an example of this case. Regardless of M:N relationship, 1:N relationships are transformed according to the rules of 1:N relationships.

Rows to individuals

Each row is converted into an individual in the ontology. For each row, if the table has primary key, an individual for its pk_class is created, first. Then, relation between these individuals is created (that is, the object property). For each simple column, a datatype property is added to the individual. After all classes were created, foreign keys (object properties) are added. For each value of a foreign key, the primary key individual should be found and related to it.

IMPLEMENTATION

The proposed method is implemented using Jena 2.5.7 (http://jena.hpl.hp.com/). To examine it, the new method should be applied to an RDB. A sample RDB named, BIRT sample database, from Eclipse (http://www.eclipse.org/birt/) is used. It is implemented using MySQL 6.0. Other RDMSs could be used, but BIRT is in MySQL. It consists of 8 tables as follows and their relations as shown in Figure 9. It also includes some test data in the package.

1. Offices: sales offices;
2. Employees: all employees, including sales reps who work with customers;
3. Customers: customers info;
4. Orders: orders placed by customers;
5. Order details: line items within an order;
6. Payments: payments made by customers against their account;
7. Products: the list of scale model cars;
8. Product lines: the list of product line classification.

Table 2 illustrates the comparison between existing methods and the proposed method. The result ontology is too long to be displayed here. Therefore, a part of offices table and some of its individuals in the ontology are as shown in Figure 10. As stated earlier, NS is replaced by the corresponding namespace, for example http://www.um.ac.ir/. As illustrated in Figure 10, a class for offices table, named offices, is created and another is created for its primary key, that is, offices_pk_class (lines 1 to 2). Primary key for this table is named officeCode and is of varchar type; therefore a datatype property, named offices-officeCode, is created that has offices_pk_class in its domain and string in its range (lines 12 to 15). To relate offices_pk_class to offices class, an object property named offices-pkOP, that is an inverse functional property, is inserted with offices_pk_class in its range and offices in its domain

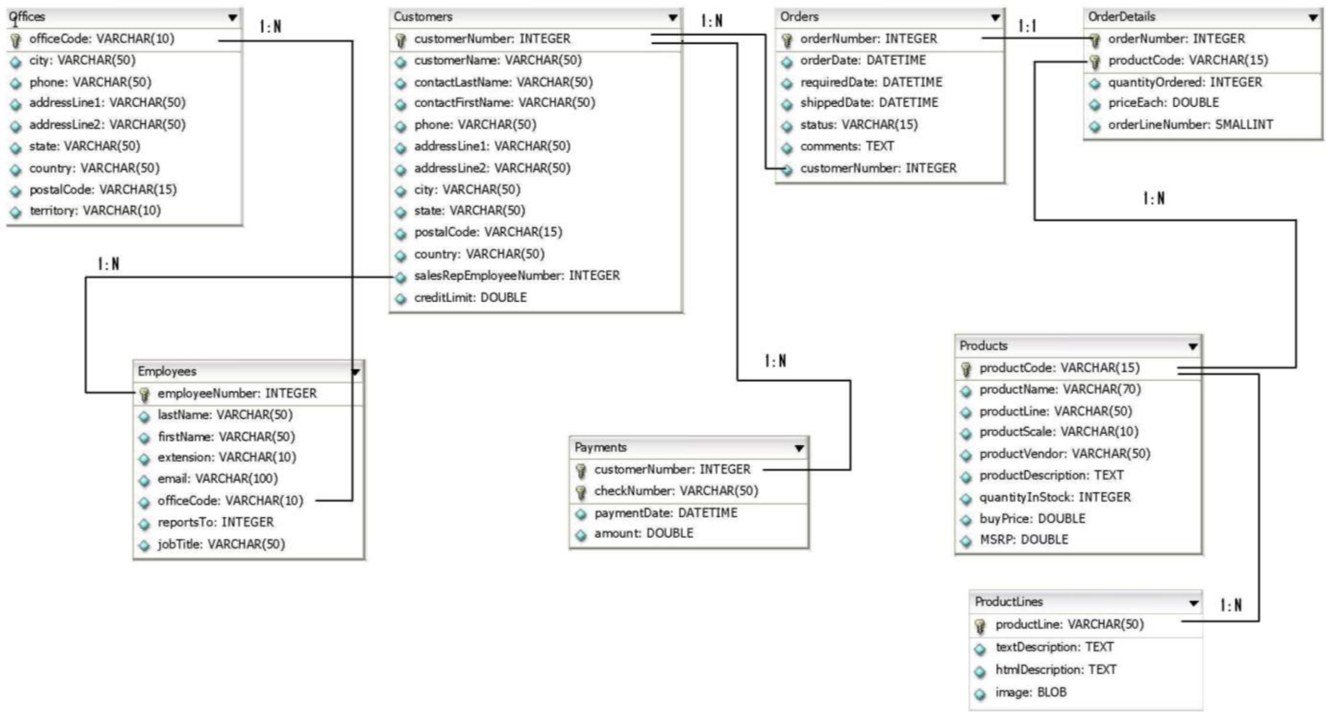

Figure 9. ER diagram for BIRT sample database.

Table 2. Comparison of existing methods with proposed method.

Method	Mode	Relationship	Data transform	Implementation
Zhuo (2007)	Auto	1:1	NO	YES
Hu et al. (2008)	Semi - Auto	1:1	NO	NO
OGSRD	Semi - Auto	1:1	YES	YES
OWLFROMDB	Auto	1:1	YES	YES
Proposed method	Auto	1:1,1:N,1:M	YES	YES

(lines 3 to 6). To apply uniqueness on the primary key, offices-pkOP is determined to be an inverse functional property. And for the not null, a min cardinality restriction named offices-pkMinRes is inserted (lines 7 to 11). For each simple column a datatype property is created. Here, two columns territory and city are displayed (lines 16 to 23). Column names in the ontology are accompanied with their table names. The reason is that, database might have same column names in different tables. If only column names are transformed to the ontology, then it is assumed to be the same objects and the result is the union of both definitions. So, their table names are included in naming objects in the ontology, such as offices_territory. A sample row is transformed to an individual as shown in lines 24 to 47. In lines 24 to 30, primary key transformation is displayed and in the rest (lines 31 to 47) simple columns are transferred. The other tables are transferred like this sample. Each row is transformed to an individual like the example.

As stated earlier, no user interaction is needed. He/she is needed to enter the name of the database. The written application in Jena is automatic and read the structure of the database, that is, tables, columns and all their information, primary keys, foreign keys, etc. It is possible as the database schema is available in RDBMSs (as meta-data). For example in MySQL, it is available in MySQL information_schema database. The transferred ontology is validated through Jena features, first. Any conflict is reported to the user. If validation is OK, the reasoning is started. Although, several reasoning methods are supported in Jena, for simplicity, Protégé software was used instead. FaCT++ (http://owl.man.ac.uk/factplusplus/) and Pellet (Sirin, 2007) are supported in Protégé 4.0. Both of them were used to evaluate the new ontology. No conflict was reported. Now, one can write queries which could be applied on OWL files to extract knowledge from the ontology. Protégé has DL Query tab, and also, SPARQL

```
<owl:Class rdf:about= NS + "#offices"/>
<owl:Class rdf:about= NS + "#offices_pk_class"/>
<owl:InverseFunctionalProperty rdf:about=NS+"#offices-pkOP">
<rdfs:range rdf:resource=NS+ "#offices-pk_class"/>
<rdfs:domain rdf:resource=NS+ "#offices"/>
</owl: InverseFunctionalProperty>
<owl:Restriction rdf:about=NS+ "#offices-pkMinRes">
<owl:minCardinality
rdf:datatype=http://www.w3.org/2001/XMLSchema#string>
</owl:minCardinality>
<owl:onProperty rdf:resource=NS+ "# offices-pkOP"/>
</owl:Restriction>
<owl:DatatypeProperty rdf:about=NS+ "#offices-officeCode">
<rdfs:domain rdf:resource=NS+ "# offices–pk_Class"/>
<rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about= NS +"#offices_territory">
<rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
<rdfs:domain rdf:resource= NS +"#offices"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about= NS +"# offices_city">
<rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
<rdfs:domain rdf:resource= NS +"#offices"/>
</owl:DatatypeProperty>
<NS:offices>
<NS:offices-pkOP>
<NS:offices-pk_Class rdf:about=NS+ "#pk_offices1">
<NS:offices-officeCode rdf:datatype =
"http://www.w3.org/2001/XMLSchema#string">36A90
</NS: offices-officeCode>
</NS: offices-pk_Class>
</NS:offices-pkOP>
<NS: offices_city> San Francisco
</NS: offices_city>
<NS: offices_phone>+1 650 219 4782
</NS:offices_phone>
</NS:offices_addressLine1>Suite 300
</NS:offices_addressLine1>
<NS: offices_addressLine2>100 Market Street
</NS: offices_addressLine2>
<NS:offices_state>CA
</NS:offices_state>
<NS:offices_postalCode>94080
</NS: offices_postalCode>
<NS:offices_country>USA
</NS:offices_country>
<NS:offices_territory> NA
</NS: offices_territory>
</NS:offices>
```

Figure 10. A part of the result ontology for offices table.

could be used.

CONCLUSION AND FUTURE WORK

In this paper, a novel approach to transfer RDB to ontology is proposed. Different parts of a relational database, such as tables, columns, primary keys, foreign keys, etc., are transferred into the corresponding map in the ontology. In previous works, primary keys and foreign keys were supported to be single-column , but in the proposed method all cases are considered. Data is transformed into individuals, too. The proposed approach was implemented using Jena and MySQL (but not limited to MySQL). No conflicts were reported either in Jena or in Protégé. For future work, we decide to apply our proposed method to extract new information from ontology or even proposed new design for RDBs.

REFERENCES

Alalwan N, Zedan H, Siewe F (2009). Generating OWL Ontology for Database Integration. 3rd Int. Conf. Adv. Semantic Process., pp. 22-31.

Astrova I, Korda N, Kalja A (2006). Toward the Semantic Web: Extracting OWL Ontologies from SQL Relational Schemata. IADIS Int. Conf. WWW/Internet, pp. 62-66.

Berners-Lee T, Hendler J, Lassila O (2001). The semantic web. Sci. Am., 284: 34-43.

Brank J, Grobelnik M, Mladenić D (2005) A survey of ontology evaluation techniques. Proc. Conf. Data Min. Data Warehouses SiKDD, 1: 166-170.

Chang-rui YU, Hong-wei W, Fu J (2006). Development Method of Domain Ontology Based on Reverse Engineering. Appl. Res. Comput., pp. 1-5.

Cullot N, Ghawi R, Yétongnon K (2007). DB2OWL: A Tool for Automatic Database-to- Ontology Mapping. In Proc. 15th Italian Symp. Adv. Database, pp. 491-494.

Fortuna B, Mladeni D, Grobelnik M (2006). Semi-Automatic construction of topic ontologies. In: Semantics, Web Min. Lect. Notes Comput. Sci., 4289: 121-131.

Ge JK, Chen ZQ (2010). Constructing Ontology-based Petroleum Exploration Database for Knowledge Discovery. Inf. Technol. Manuf. Syst., pp. 975-980.

He-ping C, Lu H, Bin C (2008). Research and Implementation of Ontology Automatic Construction Based on Relational Database. Int. Conf. Comput. Sci. Softw. Eng., pp. 1078-1081.

Hu C, Li H, Zhang X, Zhao C (2008). Research and Implementation of Domain-Specific Ontology Building from Relational Database. 3rd IEEE Int. Conf. ChinaGrid Annu., pp. 289-293.

Jiang X, Tan AH (2010). CRCTOL: A semantic-based domain ontology learning system. J. Am. Soc. Inf. Sci. Technol., 61: 150-168.

Ochoa J, HernÙndez-Alcaraz M, Valencia-Garcيa R, Martيnez-Béjar R (2011). A semantic role-based methodology for knowledge acquisition from Spanish documents. Int. J. Phys. Sci., 6(7): 1755-1765.

Secer A, Sonmez C, Aydin H (2011). Ontology mapping using bipartite graph. Int. J. Phys. Sci., 6(17): 4224-4244,

Xu G, Liu L (2011). An Approach for Ontology Construction Based on Relational Database. Int. J. Res. Rev. Artif. Intell., pp. 102-108.

Yang S, Zheng Y, Yang X (2010). Semi-automatically Building Ontologies from Relational Databases. ICCSIT, pp. 1024-1029.

Zhang L, Li J (2011). Automatic Generation of Ontology Based on Database. J. Comput. Inf. Syst., pp. 1148-1154.

Zhou L (2007). Ontology learning: State-of-the-art and open issues. Inf. Technol. Manage., 8: 241-252.

Zhou S, Meng G (2010). Tool for Translating Relational Databases Schema into Ontology for Semantic Web. 2nd Int. Workshop Educ. Technol. Comput. Sci., pp. 101-108.