Mapping DB to RDF with Additional Discovered Relations

MAMDOUH FAROUK, MITSURU ISHIZUKA
Creative Informatics Department
Graduate School of Information Science & Technology, the University of Tokyo
Tokyo, Japan
mamdouh@mi.ci.i.u-tokyo.ac.jp, ishizuka@i.u-tokyo.ac.jp

Abstract—Converting web data to semantic format is an important task for the next generation of the web. A huge amount of web data is stored in databases. Moreover, many approaches are proposed to map between DB and web ontologies. This paper proposes an approach to convert DB to RDF with extra user-defined rules. These extra rules, which extend the original data, enable semantic query engines to answer more queries. The proposed approach generates RDF data, which represents not only the original relational database but also extra discovered relations based on user defined rules. A prototype for the proposed approach is applied on a large database to show the effectiveness of the proposed idea.

Key-Word:- Mapping DB to RDF; Semantic web; Discover relations; Data representation

1 Introduction

Semantic web is a vision in which a web agent can understand web data [1]. Web data should be represented into a machine-readable format to enable web agents to understand this data. There are many semantic web languages are formalized to represent web data such as RDF, DAML, OWL. The Resource Description Framework (RDF) is a W3C recommendation that represents current web into machine understandable format.

Moreover, a huge amount of web data are stored in databases [2]. Therefore, many researchers pay much care to convert relational DB to RDF triples. Moreover, many researchers try to represent dynamic web pages into semantic format [3]. In the other hand, the process of converting DB to RDF should be simple [4] to encourage the DB owner to convert his data.

There are different approaches to convert DB to RDF [2][5][8]. A common step in these approaches is finding a mapping between DB schema and ontology structure. Based on this mapping, the DB can be accessed semantically either by generating RDF corresponding to original data or by keeping the data in the DB, where it can be managed better, and generating DRF on demand. There are different approaches for the latter way. One approach is converting SQL query result to RDF on the fly when the DB is queried [6]. This approach is suitable in case of dynamic web pages that retrieve content from underlying DB. Another approach is developing a semantic access layer as an intermediate layer between web agents and normal DB [7].

One approach that maps and converts DB to RDF is D2R [7]. D2R tool auto-generates the mapping file and the user modifies this generated file to fit the appropriate meaning. Moreover, D2R server enables the user to query the generated RDF using SPARQL queries. Dumping RDF data that represents DB is also supported by D2R.

On the other hand, the main objective of converting relational DB to RDF is to enable web agents to understand this data. However, there are some difficulties facing web agent to understand this data. One important issue that should be faced is finding implicit data. In other words, how the web agent can infer the implicit data like a human who read the normal web pages. Showing this implicit data will enables web agent to deeply understand web data.

This paper proposes an approach to convert relational DB to RDF with additional relations discovered based on user-defined rules. Unlike other approaches, our approach provides not only mapping and generating RDF but also adding extra knowledge, which is very useful in query answering process. In other words, this work proposes adding extra knowledge (user-defined rules) to the mapping schema level to improve the query-answering process. The generated RDF semantic representation together with the added knowledge can be used by intelligent search engines to infer more data and obtain accurate search results.

Although, DB is an excellent tool to store and manage data, it needs simple inference to improve its performance of querying data [5]. This work is an extension to DB2RDF approach that converts DB to

RDF data. This paper does not focus on converting DB to RDF. However, it focuses on adding extra knowledge (user-defined rules) during mapping process. These rules are useful to discover extra relations. Using these rules, web agents can understand web data easily. In the proposed approach, the generated RDF data contains not only original DB data but also inferred data that supports query answering process.

A related approach, which tries to express rules and infer additional RDF data, is SPIN [9]. SPIN is a group of RDF properties that can be used to express rules. These rules attached to a specific ontology class and can be applied to infer data, or modify the current data. *spin:rule* property can be used to defined an inference rule using SPARQL construct or insert/delete.

Moreover, SPIN adds rules to ontology level. However, our approach separates between rules level and ontology level. Separation between ontology and rules levels gives the user flexibility to add rules. In other words, it is difficult for the user to update the standard shared ontology to add his own rules. Moreover, there are many users may add rules to infer the same property depending on their own data. The user wants to extend his data depending on the semantics of the data and the expected queries to be asked. Therefore, the users have different data want to make many rules even for the same ontology. Attaching rules to dataset gives flexibility to the users and avoids rules conflicts on ontology level.

The remainder of this paper is organized as follows. Section 2 describes the overall system architecture as well as system details. The experiments and results are discussed in section 3. Finally, section 4 provides the conclusion of this research.

2 System Architecture

Although, there are many approaches convert relational database to RDF, our contribution is adding extra knowledge (user-defined rules) during the mapping process. This knowledge extends the original data and enables search engines to answer more queries easily.

The proposed system is divided generally into three main tasks as shown in Figure. 1. The first task is mapping between DB schema and web ontologies. This step is semi-automatic in which the user uses the developed mapping tool, Figure. 2, to map DB to ontology. The second task is adding extra knowledge to the mapping file. This knowledge can be used as an extension to the DB. The last task is RDF generation,

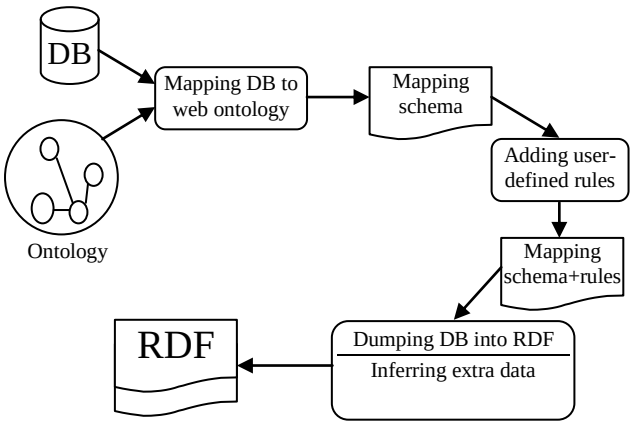

Fig.1System architecture

which generates RDF from both mapping schema and extra-added user-defined rules.

2.1 Mapping between DB and RDF

In our approach, the mapping between relational DB and RDF is a semi automatic process in which the user maps between schema of DB and ontology structure. The user uses the developed tool, Figure. 2, to map between DB tables and classes from different ontologies. The user also maps between DB fields and ontology properties. The relation between DB objects should be represented in the mapping by map foreign key fields to appropriate ontology property that represents corresponding relation between ontology classes. The generated mapping is expressed into XML intermediate format.

There are three steps for this mapping process. The first step is mapping DB tables to ontology classes. In this step, the user selects ontology class corresponding to each table. The user can select classes belong to different ontologies. The second step is mapping between DB fields and ontology properties. The user

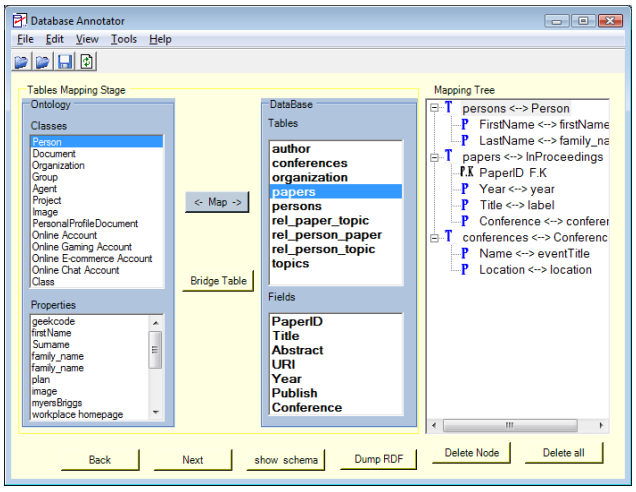

Fig.2 The mapping tool

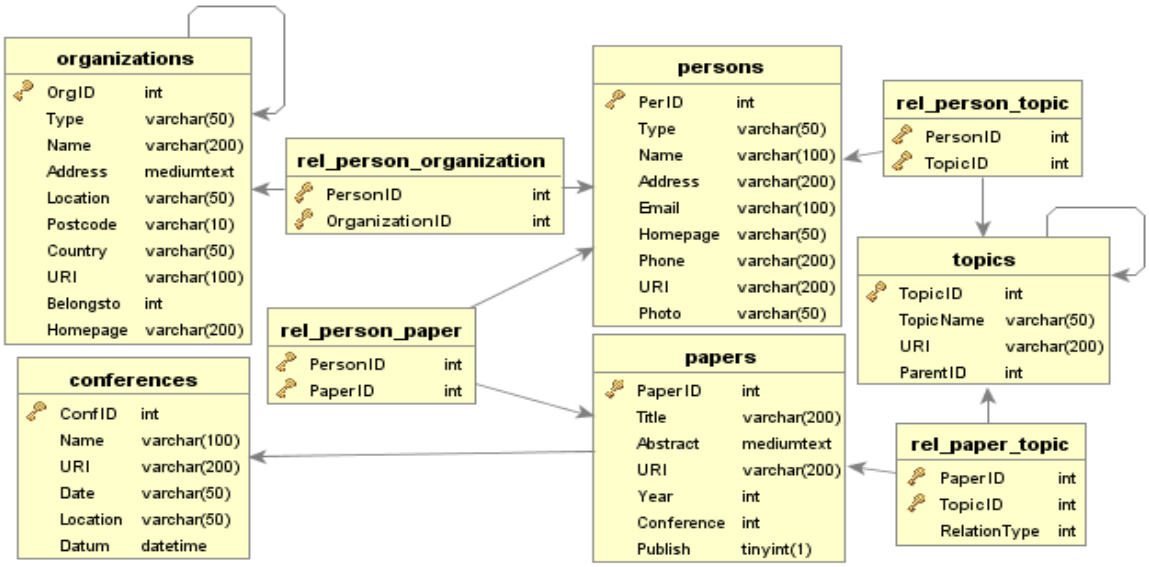

Fig.3 ISWC DB schema

selects a suitable property for each field. The mapping tool helps users to do this mapping easily and correctly. The last step is mapping relations of the DB. In this step, the user represents M-M and 1-M relation in terms of ontology relations. M-M relation is considered as two relations each one is 1-M relation. The user maps the foreign key field to the appropriate ontology property that represents same relation between ontology classes. For example, consider a DB for university researchers contains two tables: *researchers*, and *departments*. The field *deptID* in researchers table is a foreign key refers to *departments* table. In such case, the user may map *deptID* field to *hasaffiliation* property in *person* class. The domain of *hasaffiliation* property is *organization* class. *hasaffiliation* property represents the relation between *researchers* table and *department*s table.

2.2 Adding Rules

After finishing mapping between DB schema and ontology, our approach adds extra knowledge to the mapping file. This extra knowledge is considered as an extension for the original data stored in the DB. Moreover, this knowledge is used to infer more data from the DB and to support query answering process.

To clarify the idea of adding user-defined rules to the process of mapping DB to RDF consider this scenario. The database of international semantic web conferences (ISWC) contains information about some conferences in semantic web field and other data related to these conferences such as published papers, authors and so on. Figure 3 shows the schema of ISWC DB. Normally, this database is queried about authors and their interest points or their publications. For example, who is interested in semantic representation?. Who knows Prof. John? The DB or the traditionally generated RDF data cannot properly answer these questions based on the available data. Moreover, *knows* relation in foaf ontology does not exist neither in DB nor in the generated RDF data. However, a human can suggest an answer based on a simple inference. Consequently, adding some inference rules helps web agents to understand the data and answer such queries. For example, the following rules can be added:

- If a person A is an author to a paper Y, and a person B is an author to the same paper Y → A knows B.
- If a person A is an author to a paper Y, and the main topic of Y is T then → A is interested in T.

Using these rules is considered as a DB extension that adds more relations to the original relational database. As a result, these rules enable search engines to answer more queries. Adding user-defined rules depends on the meaning of the DB schema and the queries that used to be asked. These rules are added during the mapping process, which occurs only once. Using these rules solves the problem of finding the implicit information and enables web agents to go one more step to understand web data.

Production rule format, "condition → action", is used to represent the user defined rules. The syntax of production rule is carefully designed to be easy for implementation of generating extra data and to be easy for reasoning. The condition part syntax is the same as SPARQL query condition syntax. The action part is also represented into SPARQL syntax to be easy for execution. Figure. 4 shows an example for

ISBN: 978-1-61804-068-8

the added rules. The first part of the rule is xml namespaces for the used vocabularies. The second part is the conditions of the rule represented into SPARQL syntax. The last part is the action part, which is true if the conditions are true.

Moreover, there are two approaches to use these rules. The first one is to expand the original data with adding new inferred information. For example, adding the inferred relations between DB objects to RDF data. In this way, the generated RDF data contains more relations than relational DB. One advantage of this approach is that there is no need for new web agents that can use the new added rules. In other words, a normal semantic agent can make the use of these rules and consume the new added data in the same way as the original data without change its behavior. However, adding new data to the original one increases data.

The other approach to use the defined rules is to use these rules during the processing of original data to infer more data on the fly without storing the new data. This approach keeps the size of the original data. However, there is overhead processing of using inference rules during searching or processing the original data. the proposed system supports the first approach because the generated data can be consumed using normal SPARQL query engines.

2.3 Dumping RDF Data

The process of dumping or generating RDF data corresponding to DB contains two steps. The first step is automatic generation for RDF triples that represent the relational DB. This step is based on the mapping between DB schema and ontology. The second step is applying the user-defined rules on the generated DRF and adding the inferred data to the original RDF.

```
<rule id="2" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:foaf="http://xmlns.com/foaf/0.1/"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:iswc="http://annotation.semanticweb.org/iswc/iswc.daml#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
   <text>
    if two people are authors for the same paper, then they know each others.
   </text>
<condition>
<con>
{
 ?ppr rdf:type iswc:InProceedings.
 ?person rdf:type foaf:Person.
 ?person2 rdf:type foaf:Person.
 ?ppr dc:Creator ?person.
 ?ppr dc:Creator ?person2.
 FILTER (?person != ?person2)
}
</con>
</condition>
<action>
{   ?person foaf:knows ?person2.  }
</action>
 </rule>
```

Fig.4 Example of user-defined rules

2.3.1 Dumping Relational DB into RDF

This process auto-generates RDF data corresponding to the data stored in the DB. Moreover, the proposed system dumps RDF data based on the mapping file generated by the developed mapping tool. The following steps should be executed to generate RDF data.

1- From the mapping file, get all tables mapped to ontology classes.
2- For each table
   a. Create an SQL select query to retrieve all data in the table
   b. For each retrieved record, create an instance of the corresponding ontology class of the current table. // *uri of the created instance is constructed from the following pattern (table name/ auto-increment number). i.e papers/23.*
   c. For each mapped field belongs to this table in mapping file, create an instance of the corresponding property inside the created class instance
   d. Assign a value to the created property from the retrieved data.
   e. If the field represents a foreign key, the value of the created property will be a reference to another class instance

This algorithm is implemented using Java. It generates the corresponding RDF of a DB including the relations between DB objects depending on mapping schema file.

2.3.2 Adding Extra Inferred RDF Data

Using the user-defined rules, our approach inferred additional RDF triples. These triples are added to RDF data that represents DB. Rule syntax that facilitates the process of inferring and adding extra RDF is adopted. The decided format quoted from SPARQL syntax. As a result, it is easy to use SPARQL engine in inference process.

Furthermore, the proposed algorithm for inferring extra RDF data uses forward chaining to fire the rules. This means that if the condition part of a rule is true based on the available RDF data then the action part should be inserted as a new RDF triple into the RDF data. The algorithm of adding RDF triples based on the user-defined rules is as follows.

ISBN: 978-1-61804-068-8

Inputs: RDF data, user-defined rules
Output : new RDF data
For each user-defined rule

1- Get condition part of the rule
2- Construct a SPARQL select query
3- Execute the SPARQL query on RDF data
4- Replace variables in the action part of the rule with the values from the query result
5- Construct a SPARQL update query using the action part
6- Execute the update query to insert the new information to the RDF data.

This algorithm takes RDF data that represents DB and the extra rules as inputs and adds inferred RDF triples to RDF data based on rule execution. The second step in the above algorithm constructs a SPARQL query from the condition part of the current rule. The query construction is simple, the common variables in the condition part and action part are extracted and a select query for these variables is constructed with the same conditions stated in condition part of the rule. The variables in the action part are replaced with the resulted values. In addition, a new SPARQL query (insert query) is constructed from action part after replacing the variable. The new query adds inferred data to the RDF data.

Moreover, the process of adding discovered relations in the proposed approach is simple and powerful. Actually, this process implemented as execution of two SPARQL queries: a select query to check rule conditions, and an insert query to execute the action part of the rule. These two queries are constructed directly based on adopted SPAQRL syntax rule format. Consequently, adding discovered relations to RDF data is easy to implement and can be executed in a high performance way.

3 Experiments

A prototype for the proposed approach is implemented using C#. The developed tool can deal with different ontologies (RDF, DAML, OWL) and different DBMS (MySQL, SQL, MSAccess). This experiment applied on ISWC DB, which contains information about papers and authors involved in some conferences related to semantic web field. Figure 3 shows the schema of this DB. The total numbers of records in this DB is 9800 records. It contains information about 2427 authors and more than 1000 published papers. The first step to convert ISWC DB to RDF is to map between DB schema and ontology. In this mapping, we used eight different ontologies:

rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
foaf=http://xmlns.com/foaf/0.1/
iswc=http://annotation.semanticweb.org/iswc/iswc.daml#
rdfs=http://www.w3.org/2000/01/rdf-schema#
dc=http://purl.org/dc/elements/1.1/
swrc= http://swrc.ontoware.org/ontology#
swc=http://data.semanticweb.org/ns/swc/ontology#
owl= http://www.w3.org/2002/07/owl#

A part of the mapping result is shown in Figure. 5. In this mapping, DB table *papers* is mapped to *Inproceedings* class in *ISWC* ontology. The fields of the *papers* table are mapped to ontology properties as shown in Figure. 5. For example, the field *title* in the table papers mapped to *Title* property in Dublin Core ontology.

In addition, we add user-defined rules to the mapping file to be used as an extension to the original data. According to the meaning of ISWC DB and the queries to be asked, we added some rules to the mapping file. The following rules are added during the experiment.

1. If a person A is an author to a paper Y, and a person B is an author to the paper Y then → A knows B.
2. If a person A is an author to a paper Y, and the main subject of Y is T → A is interested in T.

The first rule adds *knows* relation information to the DB. *Knows* is a relation in foaf standard ontology that relates two people. ISWC DB does not contain relations between people. The second rule adds author's *interest_points* relation, which relates between person and topic.

The second step after maing DB to ontology and adding rules is auto-generation for RDF triples represent the DB. The developed tool is used in this experiment. The number of generated RDF triples is 26719. This conversion process takes 6.1408 seconds. By applying the algorithm of adding inferred data to RDF, more relations are added to the original data. The number of inferred RDF triples is 9954 using the previous two rules.

A large number of inferred RDF triples are added to the original data. By applying the algorithm of adding inferred data to RDF, more relations are added to the original data. This extra data improves query answering process and enables web agents to get implicit information. For example, the following SPARQL query asks about people who know Prof. *Edward Benson*.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT distinct ?x
WHERE
 {
  ?per foaf:name 'Edward Benson'.
  ?x foaf:knows ?per .
 }
```

By running this query on the original data, no result will be returned. However, after applying our

ISBN: 978-1-61804-068-8

approach the query returns 3 results: "Samuel Madden", "Adam Marcus" , "David Karger". This means that these three people knows Prof. *Edward Benson*

Our approach provides conversion from relational DB to RDF. In addition, it uses extra user-defined rules to generate more RDF data. Finally, our approach generates more information represented into RDF that helps semantic search engines to answer more queries.

4 Conclusion

All web data should be available into semantic languages even implicit data because this datasets should be machine understandable. Mapping between DB to RDF is very important and many researches investigate this point. However, this paper proposes a new approach that adds extra user-defined rules to the mapping between DB and web ontologies. These rules that add the implicit data are considered as an extension to the DB. A prototype is implemented to show the visibility and effectiveness of the proposed idea. Moreover, the experiment shows that the proposed approach converts a large DB to RDF in a few seconds and adds more relations, which are useful to facilitate query answering.

```
<DB>
  <bridge_table name="rel_person_paper">
    <foreignkey field="PersonID"
belongToClass="InProceedings" mapToProp="Creator"
refToClass="persons" corespondFK="PaperID"
ontoIndex="dc" />
  </bridge_table>
  <table name="papers" RTClass="InProceedings"
ontoIndex="iswc">
    <primarykey>
      <field name="PaperID" />
    </primarykey>
    <foreignkey name="Conference" RTProperty="conference"
RTTable="conferences" ontoIndex="iswc" />
    <field name="Title" RTProperty="Title" ontoIndex="dc" />
    <field name="Abstract" RTProperty="Abstract"
ontoIndex="dc" />
    <field name="Year" RTProperty="Date" ontoIndex="dc" />
  </table>
....
```

Fig.5 Mapping between DB and ontology

References:

[1] T. Berners-Lee, J. Hendler, O. Lassila, "The Semantic Web," Scientific American, Vol. 284, No. 5, 2001, pp. 34-43.

[2] Siegfried Handschuh, Raphael Volz, Steffen Staab, Annotation for the Deep Web, IEEE Intelligent Systems, v.18 n.5, September 2003, pp.42-48.

[3] Zhuoming Xu, Shichao Zhang, and Yisheng Dong, Mapping between Relational Database Schema and Owl Ontology for Deep Annotation, WI'06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, IEEE Computer Society, 2006, pp. 548-552.

[4] Svihla, M., Jelinek, I.: The Database to RDF Mapping Model for an Easy Semantic Extending of Dynamic Web Sites. Proceedings of IADIS International Conference WWW/Internet, Lisbon, Portugal, 2005, pp.27-34

[5] Pan, Z. and Heflin, J.: DLDB: Extending Relational Databases to Support Semantic Web Queries, In Workshop on Practical and Scaleable Semantic Web Systems, The 2nd International Semantic Web Conference (ISWC2003) (2003).

[6] Mamdouh Farouk, Samhaa R. El-Beltagy, Mahmoud Rafea, "On-the Fly Annotation of Dynamic Web ," Proceedings of the First International Conference on Web Information Systems and Technologies (WEBIST 2005)," Miami (USA), may 2005, pp 327-332.

[7] Chris Bizer, and Richard Cyganiak :D2R server Publishing Relational Databases on the Semantic Web , www4.wiwiss.fu-berlin.de/bizer/d2r-server/ , 2010

[8] Ismael Navas Delgado, Nathalie Moreno Vergara, Antonio C. Gomez Lora, María del Mar Roldán García, Iván Ruiz Mostazo, José Francisco Aldana Montes: "Embedding Semantic Annotations into Dynamic Web Contents". Proceeding of 15th international workshop on database and Expert Systems Applications, 2004, pp. 231-235

[9] Holger Knublauch, James A. Hendler, Kingsley Idehen "SPIN - Overview and Motivation", http://www.w3.org/Submission/2011/SUBM-spin-overview-20110222/ , February 2011.