Mapping Relational Database into RDF Graph Representation

Kyawt Kyawt San, Khin Nweni Tun
University of Computer Studies, Yangon
kyawtkyawts@gmail.com,knntun@gmail.com

Abstract

The Semantic Web has been developed in order to publish and access a lot of information for automatically processing of data and information by machines rather than for people. RDF (Resource Description Framework) data model is a proven technology for generating semantic data from existing relational data representation. The certain loss of semantics has occurred in the generation of relational data to RDF. This paper describes the development of the RDB to RDF mapping system, which transforms relational data into semantically structured data and publishes it on the web. The proposed mapping solution extracts relational schema and data from an existing relational database. Then, relation URI, attribute URI and tuple URI are generated. Finally, a series of triples are generated from those URI and validated against RDF validator. The implementation of the tool is made using the standard relational databases such as sakila, world and employee datasets.

1. Introduction

The goal of the Semantic Web is to meaningfully structure the content of the Web. The amount of data found on the web today is lack of semantics and becomes increasingly harder to retrieve a particular piece of information. Another problem arises when trying to combine the collected pieces of information in a meaningful way. For example, applications accessing information from several databases with different structure and content has to decide if a column 'A' from one database has the same meaning as some column 'A' from another database. This is very challenging due to the lack of semantics in the database schema.

Hence, RDF data model becomes a proven technology for generating semantic data from existing relational data representation. Those data should be represented in machine-readable format. Undoubtedly, the relational data model is the major sources of information for future development of Semantic Web. There are many formalized semantic web languages to represent web data such as Resource Description Framework (RDF), DARPA Agent Markup Language (DAML), and Web Ontology Language (OWL). To contribute to creating the Semantic Web data, generating RDF data from relational databases plays a major important role.

However, recreating RDF data manually is infeasible. Hence, RDF Metadata generation and processing are still topics of a research. Since the majority of information in the world still resides in relational databases, it should be investigated how to expose this information as RDF queryable with SPARQL Protocol and RDF Query Language (SPARQL).

One way to expose data in relational databases as RDFS representations is to download them to RDF repositories. However, this can be very costly when the relational database is large. The fact that all data in the relational database is duplicated as RDF introduces a lot of data redundancy. Also, when the rate of change in the relational database is high, a lot of time is spent on propagating the changes to the RDF repository.

This system implies the necessity to find automatic processes for converting relational data into RDF. In addition, the work described in this system aims to contribute to a complementary approach for extracting the necessary structure from existing information sources such as relational databases to fulfill the needs of the Semantic Web.

During the mapping process for relational DB to RDF triples, the solution proposed is to represent each "fact" or statement from the relational database as a subject–verb–object triple. The proposed system converts relational data into RDF and expose relational data so that it can be queried through SPARQL, the query language of the semantic web, which is often referred to as the "RDB-to-RDF" process.

The organization of the paper is as follows: Section 2 presents the challenges encountered in Semantic Web and state-of-the-art solutions. Section 3 describes the proposed RDF Data Creation system from Relational data. Section 4 presents the alternative way of displaying RDF graph and the processing steps for checking integrity constraints from RDF graph using SPARQL query language. Section 5 concludes the paper.

2. Motivation

The "deep web", as opposed to the "surface web", is a part of the web content that is hardly indexed by standard search engines. It refers to the data hidden in unstructured documents (images, scans), semi-

structured documents (csv, pdf...), or structured data sources (relational databases, xml databases, NoSQL databases, LDAP directories...) only accessible through query forms but which content cannot be browsed by standard tools. It is preferable that the data remains hosted in the legacy RDBs to expose in Semantic Web becomes a major challenge. Hence RDB-to-RDF techniques that can access relational data and convert it into RDF triples appears as a powerful and promising method to achieve in data integration.

2.1. State-of-the-Art Solutions

Using RDF as a format for representing relational data appears as a powerful and promising method to achieve such data integration, in which RDB-to-RDF methods will play a key role. The main purpose of the RDB2RDF Working Group is to be able to:

(i) Publish on the Web the vast amounts of information stored in Relational Databases (RDB)
(ii) Integrate data from different RDBs
(iii) Add semantics to the existing relational data

Many existing RDB-to-RDF mapping approaches have provided different mechanisms with which to tackle the RDB-to-RDF mapping process.

Therefore, a comparative study retaining the aforementioned existing solutions of the state-of-the-art is described that summarizes their main characteristics.

❖ **OpenLink Virtuoso**

It enables the exposure of pre-existing accessible relational data as Virtual RDF graphs [14]. The data is accessible through SPARQL queries or SPASQL (SPARQL within SQL). RDF datasets are resulted from this process without physical regeneration of relational data.

The data materialization in this tool is possible but not that easy, and this is definitely not the intended goal as Virtuoso comes with its own SPARQL endpoint.

❖ **D2R Server**

D2RQ Platform is implemented as a Jena graph, used to access relational databases as virtual, read-only RDF graphs [4]. Using the virtual access to the relational content the replication of data is avoided. Using this platform is possible to query a non-RDF database using the query language SPARQL and access the content as Linked Data over the Web.

It also creates custom dumps of the relational contents in RDF format in order to be loaded into a triplestore.

❖ **Direct Mapping**

The translation process of Direct Mapping is simple and straightforward [15]. For each tuple of each table in the database, the same routine will be applied, the final goal being the obtaining of a set of triples (s,p,o).

The subject part (s) is gained by using the primary key of the tuple. It is aggregated with a base URI, the table name and the primary key column name to form a valid URI. The predicates are acquired by a similar mechanism; the column names are concatenated with the table name and the base URI. Objects are directly considered as literals.

Direct Mapping is an easy-to-learn and easy-to-use standard, designed for an easy implementation.

❖ **UltraWrap**

The triple representation of the relational data is implemented as a three-column SQL view (subject, predicate, object), that consists in the union of all the queries that define all the triples as defined by the local ontology [10]. Consequently, a SPARQL query can be naively rewritten into an SQL query on the SQL view, benefitting from the native query optimizer of the relational database.

Its main advantage is that SPARQL execution is as fast as SQL. There is no description, however, of the way the mapping description is derived into the SQL view.

In addition, the observed fact is that producing RDF data with sufficiently rich semantics is a critical concern of most approaches, in order to make the data usable, interoperable and linkable.

The main distinguishing feature of the proposed algorithm is the use of simple concatenation function to generate URI for RDF instances instead of using complex logics. In addition, implicit relational constraints are extracted and encoded during the URI generation process.

As a conclusion, it is observed that the quality of an RDF resulting of a mapping highly depends on the richness of the SQL schema with respect to its encoding of integrity semantics.

3. RDF Data Creation from Relational Database

In order to create an RDF dataset out of a regular relational dataset, it is necessary to map this data to RDF. The proposed system works as an intermediary between a Semantic Web client and a relational database. The system extracts relational data and schema from a relational database, transforms it into RDF and stores the generated triples in a text document.

The first step in converting relational data to RDF is to define a schema of the data. A collection of data does not say anything about the schema designed to work with the data. In order to show how this data relates, a schema needs to be designed. Relational schema with integrity constraints are extracted and encoded in terms of RDF format.

This paper focuses on extracting and checking integrity constraints from a single RDF document generated from RDB. Enforcing constraints in a single

RDF document represents the majority of ongoing research interests in the field of ICs in Semantic Web.

3.1. Translation Process

The translation process concerns converting existing RDB data to the format defined by the target schema. Data stored as tuples in an RDB are converted into series of RDF triples with the common subject.

RDB Data conversion is performed in three steps as shown in Figure 1. Firstly, the RDB relations' tuples are extracted. Secondly, these data are transformed (converted) to match the target format. Finally, the transformed data are loaded into text files suitable for bulk loading in order to populate the schema generated earlier during the Schema translation phase. RDF document is generated using a set of data instance conversion rules.

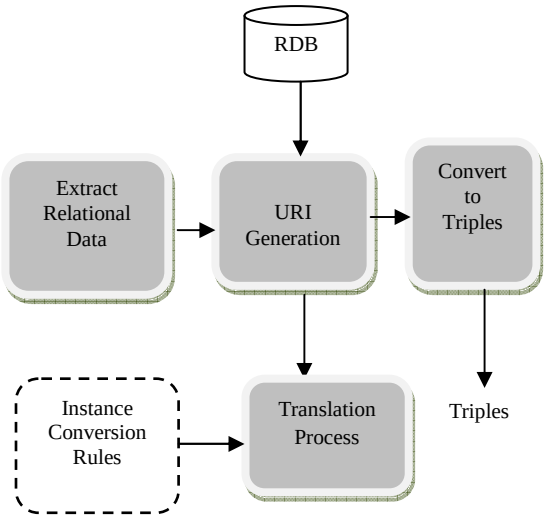

Figure1. RDF Data Generation Process

3.1.1. Creating URI

URI is generated based on URI generation principles and the proposed algorithm shown in Figure2. URI can be any combination of numbers and letters, provided it is unique. RDF requires the subject and predicate of a triple to be URIs, while the object can be a URI or a literal string such as "Ode to himself". When the object of a triple is a URI, it is possible to match it to the subject of another triple, allowing the triples to be chained into "linked data".

The subject of a triple must be a URI. Therefore it must be globally unique, and not used to identify any other resource. According to the standard example database, the column name "***Code***" may be unique to the database providing the record, but it cannot be guaranteed to be unique outside of that local environment.

In this translation of relational data to RDF approaches, relations, attributes and tuple URI are generated. URI generation in this section is done according to the rules described in Figure 2. All URIs are generated by appending to a base URI.

Relation URI is an URI formed from the concatenation of the base URI and table name.

Then, **tuple URI** is generated for each relation having a primary key.

URI for each attribute is generated before generating Reference triples. It is formed from the concatenation of the base URI, table name and the column name.

For example, a description of a standard relational database "***world***" includes information as shown in Table 1, Table 2 and Table 3. This information is laid out as a set of pairs of attributes and their values.

The attribute/value pairs are reformatted into subject-predicate-object statements by using the record identifier as the subject of each statement, as shown in Table 4.

For each statement, the attribute will form the predicate and the value will be the object. Each statement can now be represented as an RDF data triple.

3.1.2. Replacing Relational Data with URI

Assume given a base URI "***base"*** for the relational database to be translated (e.g., "http://www.ucsy.edu.mm/db/"), then the proposed algorithm produces relation URI according to running example:
<http://www.ucsy.edu.mm/worlddatabase/city/>
<http://www.ucsy.edu.mm/worlddatabase/country>,
http://www.ucsy.edu.mm/worlddatabase/countrylanguage/>.

Then, **tuple URI** is generated for each relation having a primary key. Thus, given that the facts PK_1 *(ID, Name)* and *VALUE("1", "Kabul")*, then the URI ***<http://www.ucsy.edu.mm/worlddatabase/city/ID=1>*** is the identifier for the tuple in table ***City*** with value *1* in the primary key.

Then, according to the algorithm, attribute URI is generated. **URI for each attribute** is generated before generating Reference triples. It is formed from the concatenation of the base URI, table name and the column name.
<http://www.ucsy.edu.mm/worlddatabase/city/ID>
<http://www.ucsy.edu.mm/worlddatabase/city/Name>
<http://www.ucsy.edu.mm/worlddatabase/city/CountryCode>
<http://www.ucsy.edu.mm/worlddatabase/city/District>
<http://www.ucsy.edu.mm/worlddatabase/city/Population>
are generated for each attribute in relation ***City*.**

Through out this section, a standard relational database ***"world"*** is used as a running example.

The mapping process generates three types of triples when translating a relational instance: Table triples, reference triples and literal triples.

3.1.3. Publish RDF Triples

The final step is to publish the set of RDF triples derived from the example record. Triples can be stored and displayed in a number of formats, called serializations. One of these uses NTriple language [6], which makes it easier to see the three-part structure of each triple.

- **Table triples**: For each relation, store the tuples that belong to it.
- **Literal triples**: For each tuple, store the values in each of its attributes.
- **Reference triples**:Store the references generated by foreign keys.
- For each attribute, the following literal triples are generated from relation ***City***:

<http://www.ucsy.edu.mm/worlddatabase/city/ID=1>
<http://www.ucsy.edu.mm/worlddatabase/city/ID> "1"

<http://www.ucsy.edu.mm/worlddatabase/city/ID=1>
<http://www.ucsy.edu.mm/worlddatabase/city/Name>
"Kabul" . etc,.

Table1. Sample Data for Country Table

Code	Name	Continent	Region	Indep Year	Population	Capital
AFG	Afghanistan	Asia	Southern and Central Asia	1919	22720000	1
ALB	Albania	Europe	Southern Europe	1912	3401200	13
ANT	Netherlands Antilles	North America	Caribbean		217000	16
DZA	Algeria	Africa	Northern Africa	1962	31471000	20
NLD	Netherlands	Europe	Western Europe	1581	15864000	5

Table2. Sample Data for City Table

ID	Name	Countrycode	District	Population	CapitalCity ID
1	Kabul	AFG	Kabol	1780000	2
5	Amsterdam	NLD	Noord-Holland	731200	2
13	Apeldoorn	ALB	Gelderland	153491	2
16	Haarlem	ANT	Noord-Holland	148772	2
20	´s-Hertogenbosch	DZA	Noord-Brabant	129170	18

Table3. Sample Data for CountryLanguage Table

CountryCode	Language	IsOfficial	Percentage
'AFG'	'Balochi'	'F'	0.9
'ALB'	'Greek'	'F'	1.8
'ANT'	'Papiamento'	'T'	86.2
'DZA'	'Berberi'	'F'	14.0
'NLD'	'Fries'	'F'	3.7

Table 4: Description Formatted as a Set of Statements

ID	Attribute	Value
1	Name	'South Hill'
1	Countrycode	'AIA'
1	District	'–'
1	Population	961
1	CapitalCityID	2

3.1.3. Publish RDF Triples

The final step is to publish the set of RDF triples derived from the example record. Triples can be stored and displayed in a number of formats, called serializations. One of these uses NTriple language [6], which makes it easier to see the three-part structure of each triple.

- **Table triples**: For each relation, store the tuples that belong to it.
- **Literal triples**: For each tuple, store the values in each of its attributes.
- **Reference triples**:Store the references generated by foreign keys.

For each attribute, the following literal triples are generated from relation ***City***:
<http://www.ucsy.edu.mm/worlddatabase/city/ID=1>
<http://www.ucsy.edu.mm/worlddatabase/city/ID> "1"

<http://www.ucsy.edu.mm/worlddatabase/city/ID=1>
<http://www.ucsy.edu.mm/worlddatabase/city/Name>
"Kabul" . etc,.

- **Reference Triples**

Recall that attribute ***CountryCode*** is a foreign key in relation ***City*** that references the attribute ***Code*** in relation ***Country***. Then from
the facts Value (1, ***ID***, t1, ***City***) and Value (AFG, ***Code***, t3, ***Country***), the following triple is generated:

Triple:
<http://www.ucsy.edu.mm/worlddatabase/city/ID=1>
<http://www.ucsy.edu.mm/worlddatabase/city/Country Code>
<http://www.ucsy.edu.mm/worlddatabase/country/Code =AFG>.

Each row is turned into a series of triples with a common subject. The final RDF triples is obtained by one processing step. Once information is in RDF form, it becomes easy to process it.

The validation of resulted RDF is done by using RDF validator and converter. The validated RDF code in shown in Listing 1by using RDF/XML format.

Another way of displaying triples is as a graph, in which nodes represent the subject and object of a triple, with a connecting line representing the predicate. A round node is used for a subject or object URI, while a rectangular node displays an object literal. The connecting line uses an arrow to indicate the direction from subject to object. The RDF graph for the reconstituted record is shown in Figure 3 and it shows the RDF graph representation of the relational data with its corresponding relational schema. During the mapping process, a corresponding RDF instances are automatically generated. RDF code for entire example database is too long and some of the triples are shown as an RDF graph.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:city="http://www.ucsy.edu.mm/standardworld/city/"
xmlns:country="http://www.ucsy.edu.mm/standardworld/country/"
xmlns:
countrylanguage="http://www.ucsy.edu.mm/standardworld/countrylanguage/">
<rdf:Description
rdf:about="http://www.ucsy.edu.mm/standardworld/city/ID=1">
<city:ID>1</city:ID>
<city:Name>Kabul</city:Name>
<city:CountryCode>
<rdf:Description
rdf:about="http://www.ucsy.edu.mm/standardworld/country/Code=AFG">
<country:Code>AFG</country:Code>
<country:Name>Afghanistan</country:Name>
<country:Continent>Asia</country:Continent>
<country:Region>Southern and Central Asia
</country:Region>
<country:SurfaceArea>652090.00
</country:SurfaceArea>
<country:IndepYear>1919</country:IndepYear>
<country:Population>22720000</country:Population>
<country:GNP>5976.00</country:GNP>
<country:LifeExpectancy>45.9</country:LifeExpectancy>
<country:GNPOld> </country:GNPOld>
<country:LocalName>Afganistan/Afqanestan</country:LocalName>
country:GovernmentForm>Islamic Emirate
</country:GovernmentForm>
<country:HeadOfState>Mohammad Omar
</country:HeadOfState>
<country:CapitalCityID
rdf:resource="http://www.ucsy.edu.mm/standardworld/city/ID=1" />
<country:Code2>AF</country:Code2>
</rdf:Description>
</city:CountryCode>
```

```
<city:District>Kabol</city:District>
<city:Population>1780000</city:Population>
<city:CapitalCityID
rdf:resource="http://www.ucsy.edu.mm/standardw
orld/city/ID=1" />
</rdf:Description>
</rdf:RDF>
```

Listing1.Validated RDF/XML Code for Sample World Database.

4. Querying Generated RDF Data

SPARQL queries are run on the generated triples by using Twinkle query engine.

In SPARQL, the example query for extracting primary key constraints from generated ***world*** RDF data would look like this:

```
select  DISTINCT ?keyname
where {
?keyname
<http://www.ucsy.edu.mm/Constraints/KeyType>
"Primary Key".
}
```

Thus, for the previous query, the result returned by Twinkle is shown in Figure 4.

In this step, it is interrogated that whether primary key constraints violation occurs or not in the generated RDF document. SPARQL queries for checking primary key constraints would look like this:

```
ASK {
 ?x
<http://www.ucsy.edu.mm/standardworld/country/Code
> "AFG" .
 ?y
<http://www.ucsy.edu.mm/standardworld/country/Code
> "ALB".
  FILTER (? x=?y)  }
```

If the result returned by Twinkle is “yes”, the violation occurs. In this example, the result is “no”. Hence, primary key constraint is followed as shown in Figure 5.

The sample queries for checking foreign key constraint and the returned result is shown in Figure 6. The proposed system gives an easy and automatic production of RDF document once the classic relational database is loaded into the system.

Some tests have been done on an experimental set of standard relational databases. A prototype has been developed to show the proposed transformation system of RDF triples and the results shows the promising results. The mapping process is implemented using java and mysql.

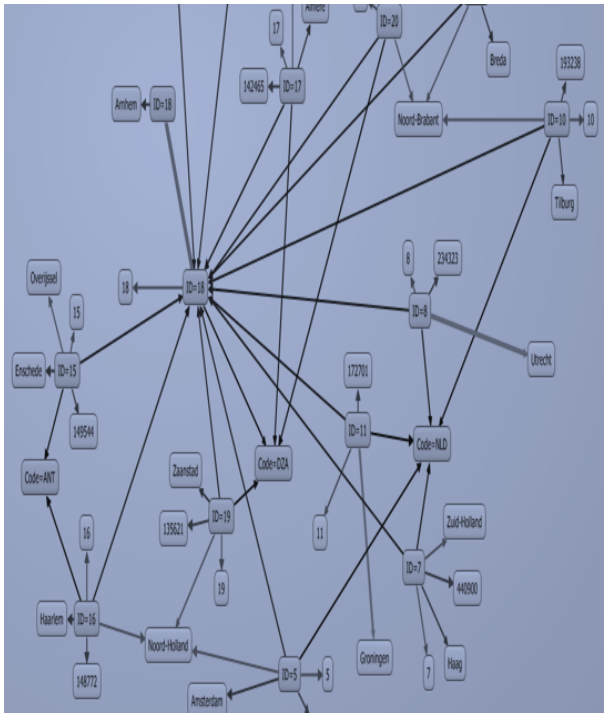

Figure3. Publishing RDF Triples as an RDF Graph

keyname
http://www.ucsy.edu.mm/standardworld/countrylanguage/Language
http://www.ucsy.edu.mm/standardworld/countrylanguage/CountryCode
http://www.ucsy.edu.mm/standardworld/country/Code
http://www.ucsy.edu.mm/standardworld/city/ID

Figure4. Result of the Extracting Primary Key Name

```
ASK {
 ?x <http://www.ucsy.edu.mm/standardworld/country/Code> "AFG" .
 ?y <http://www.ucsy.edu.mm/standardworld/country/Code> "ALB" .
   FILTER (?x=?y)
}
```

```
no
```

Figure5. Result of the Checking Primary Key Constraint

```
ASK {
   <http://www.ucsy.edu.mm/standardworld/country/Code=AFG>
   <http://www.ucsy.edu.mm/standardworld/country/Capital> ?x

OPTIONAL {  ?y <http://www.ucsy.edu.mm/standardworld/city/ID> "1" .  }
   FILTER (!bound(?y))
}
```

```
no
```

Figure6. Result of the Checking Foreign Key Constraint

5. Conclusion

Mapping between RDB and RDF is very important and a lot of research investigates this point. However, so far, there is none full-automatic mapping tool of SQL to RDF, most of which is defined manually. Further, with RDF parsers being available easily today, data expressed in RDF can be parsed and processed easily by a machine for any desired purpose. The mapping described in this system can be enhanced further so as to represent higher feature of integrity constraints.

5.1. Advantages and Limitations

The main advantages of the proposed system are that RDF instances are generated automatically from RDB. In addition, some of the desirable features are described below:

1. User does not need to have knowledge of the relational schema.
2. No need for extra information from a user to translate his database.
3. No additional mapping is required to achieve the process.
4. Increases the value of the existing data and enables new applications on that data.
5. No need for using a new mapping language.

Despite of its benefits, it has its own some drawbacks. Here are some of the limitations of the proposed system. The tool does not support officially other database systems except the MySQL relational database. In addition, it is lack of ability to infer knowledge with ontologies.

References

[1] A. A. Kharusi and M.H. Selamat,et al., "Visual Mapping of Relational Database for Semantic Web", In: Proceedings of the 3rd International Conference on Machine Learning and Computing, ICMLC 2011, ©2011 IEEE, pp.189-192.

[2]C. Bizer, R. Cyganiak, "D2R server - Publishing relational databases on the semantic web", In: Proceedings of the 5th International Semantic Web Conference, 2006: p. 26.

[3]C. Bizer, R. Cyganiak. "D2RQ — Lessons Learned". In: Proceedings of the W3C Workshop on RDF Access to Relational Databases, Cambridge, MA, USA, October 2007. W3C. Position paper.

[4]C.Bizer, A.Seaborne. "D2RQ-treating non-RDF Databases as Virtual RDF Graphs", In: Proceedings of the 3rd international semantic web conference (ISWC2004), http://www.wiwiss.fu-berlin.de/suhl/bizer/pub/Bizer-D2RQ-ISWC2004.pdf. volume 2004, 2004.

[5] C.Lei, Y.Nansheng, "Publishing Linked Data from Relational Databases Using Traditional Views", In: Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2010, pp.9-12.

[6] D.Beckett, "N-Triples A line-based syntax for an RDF graph", W3C Working Group Note 09 April 2013, http://www.w3.org/TR/2013/NOTE-n-triples-20130409/.

[7] E. P.hommeaux and A.Seaborn, "SPARQL query language for RDF",Technical report, World Wide Web Consortium, 2013, http://www.w3.org/TR/rdf-sparql-query/.

[8]G. Lausen, M. Meier and M. Schmidt, "SPARQLing Constraints for RDF", In: Proceeding of the 11th international conference on Extending database technology: Advances in database technology, EDBT '08', New York, USA, 2008, pp. 499-509.

[9] J.Bakkas and M.Bahaj, "Generating of RDF graph from a relational database using Jena API", International Journal of Engineering and Technology, Vol 5 No 2, Apr-May 2013.

[10]J.F. Sequeda, R. Depena, D.P. Miranker, " Ultrawrap: Using sql views for rdb2rdf" , In: Proceedings of the 8th International Semantic Web Conference ISWC2009, 2009.

[11]Kyawt Kyawt San, Khin Nweni Tun, "Automatically Generating RDF Instances from Relational Data Sources", ICCA 2013.

[12]Kyawt Kyawt San, Khin Nweni Tun, "Formalization of Direct Mapping for Relational Data to Semantic Web Representation", ICCA 2012.

[13]L. Chen, N. Yao, "Publishing Linked Data from Relational Databases Using Traditional Views", 3rd IEEE International Conference on Computer Science and Information Technology, ICCSIT, 2010, pp. 9-12.

[14]OpenLink Software. Openlink Virtuoso, http://virtuoso.openlinksw.com/. [Online, accessed 28/06/2013].

[15]W3C Working Draft, "A Direct Mapping of Relational Data to RDF", 29 May 2012, http://www.w3.org/TR/2012/WD-rdb-direct-mapping-20120529/.