Creating an RDF Graph from a Relational Database Using SPARQL

Ayoub Oudani, Mohamed Bahaj*, Ilias Cherti
Department of Mathematics and Informatics, University Hassan I, FSTS, Settat, Morocco.

* Corresponding author. Tel.: 0660250506; email: mohamedbahaj@gmail.com
Manuscript submitted May 11, 2014; accepted October 20
doi: 10.17706/jsw.10.4.384-391

Abstract: Despite the trend towards a Semantic Web, the current Web still lacks semantic data. Most information is modeled and stored in relational databases and therefore is out of reach for many Semantic Web applications. In this context, we propose an automatic method to extract data from an RDB and create a graph oriented RDF structure using a mapping process based on SPARQL.

Key words: Ontologies, semantic web, relational database, RDF, RDFS, SPARQL, OWL.

1. Introduction

The generalization of RDF structure on the Internet has increased the demand for efficient storage of RDF data. This is also the case for the various tools and software agents dedicated to the recovery and the exploitation of RDF graphs. Several works have been developed to narrow the gap between XML/RDB and OWL ontology [1]-[4]. However, the vast majority of these data are still stored in relational databases (RDB). Mapping data in relational databases in a semantic web has been a very important research topic due to the need to migrate to the Semantic Web. In [5] the authors presented "Relational.OWL" which automatically extracts the semantics of relational databases and transforms it into RDF/OWL. However, many problems arise. First, while DBMS systems remain the most efficient for storing and processing data, some approaches used for storing RDF triples results of their mechanism [6]-[8]. Second, where the aim is to use the DBMS as a data source, other approaches tend to design mechanisms for translation of SPARQL to SQL [9], [10] which requires learning these specific languages and creates the problem of the availability of RDF source files that remain dependent on the DBMS.

In this paper we propose an approach to map an RDB to an existing ontology while conserving of the structure of the database. The most popular approach to this class of mapping is Relational. OWL [11]. To extend this approach we propose a more semantic mapping, as well as the treatment of uninsured constraints by taking into consideration cardinality and relationship tables bearing properties. Therfore, we present a process to map from relational databases to OWL ontology. We then propose OWL ontology for the representation of the structure of a relational database in general, which we adapt to the structure of a personalized targeted database. The result can be exploited later for a secondary mapping to an external ontology using SPARQL.

2. Description of the Process

The aim of our work is to map the data from a relational database to an existing ontology while

preserving the semantic relationships between the tables defined in the structure of the source RDB and to use the generated structure for a reconstruction of the source RDB. Our method starts by extracting the structure of the database and saving it by creating an instance of the ontology 'Abstract.OWL' described in the next section. Once the schema is created, the RDB records are exported to a file RDF following the rules defined in this schema. The final phase of our method is execution a SPARQL query for the construction of the new RDF graph of the target ontology using the "CONSTRUCT" clause as illustrated in Fig. 1. In the next section we present a detailed description of the steps of our mapping process.

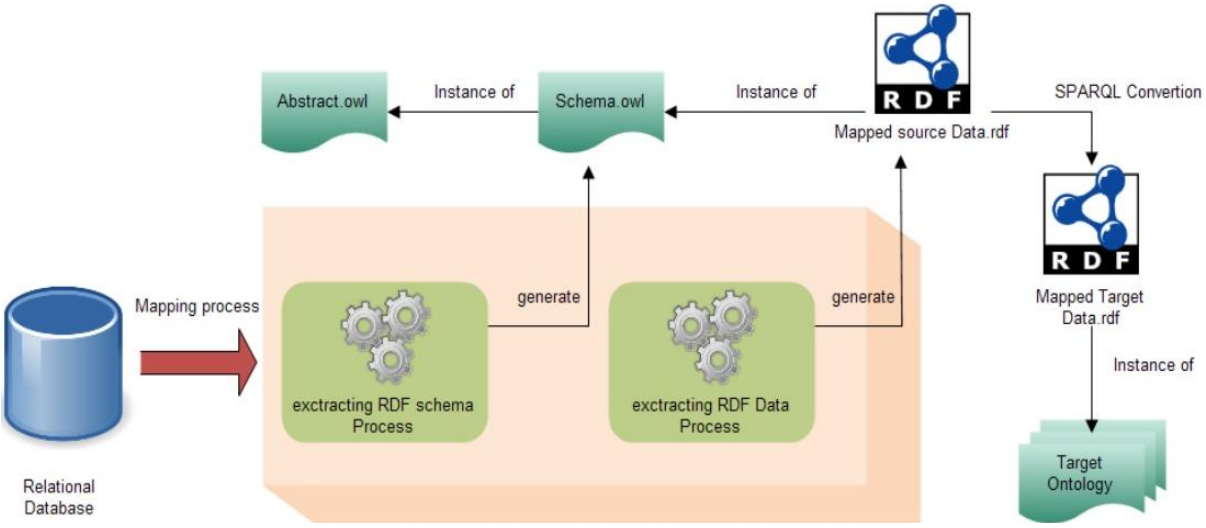

Fig. 1. Description of the mapping process.

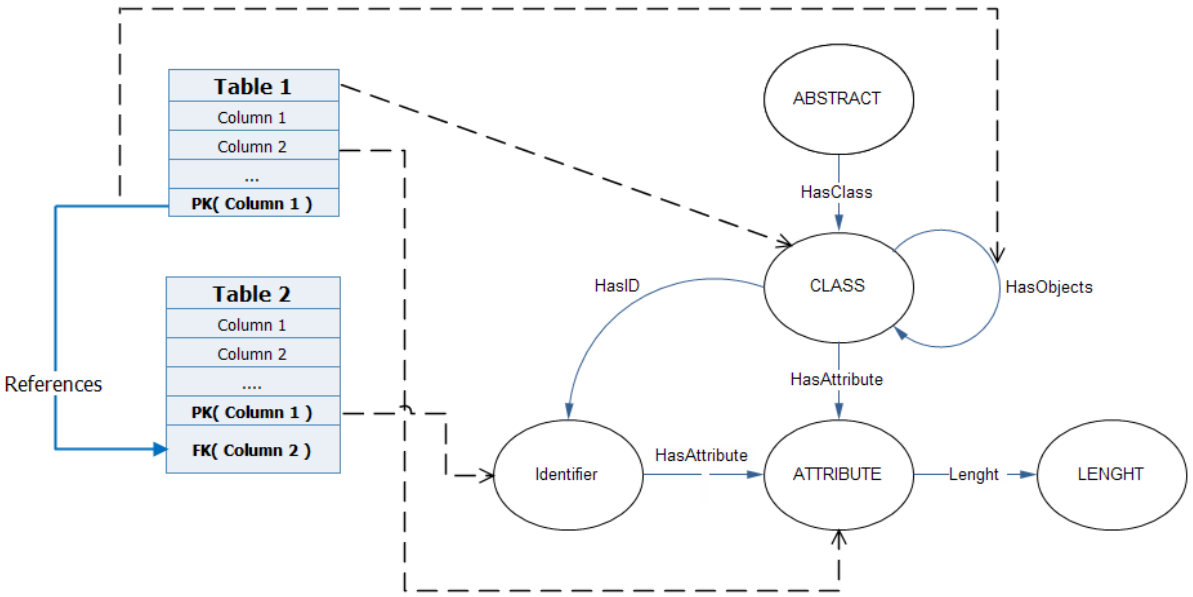

Fig. 2. Representation of classes of "Abstraction" and properties.

3. Detailed Description

3.1. 'Abstract.OWL' Ontology

In order to describe the schema of a relational database with the techniques provided by RDF and OWL, we define OWL reference classes, to which any documents describing this database can refer. The abstract representation of classes is designed to specify the possible relationships between all tables in a database and their structures. We call this basic representation 'Abstract.OWL.' It contains the semantic reasoning

and data structure of the relational database. The relational database is represented by a main class «Abstract». Tables are represented by the class "Class", columns by the "Attribute" class and primary keys by the "Identifier" class. Thus, we can create representations of schemas adapted to the BDR based on this ontology. These can, in turn, can be used as anthologies for data representation. Thus, using the representation of the schema created, we can use our 'Abstract.OWL' ontology individuals as classes. In this manner the representation of the schema created belongs to OWL Full.

In the remainder of this article, we abbreviate our main namespace 'http://localhost/RDF/abstract.owl #' prefixed with 'abs'. 'Rdf', 'rdfs' and 'owl' correspond to commonly used prefixes for RDF, RDF Schema and OWL.

A summary of all classes represented in the ontology 'Abstract.OWL' is provided in Table 1. Table 2 presents a list of relationships that link these classes. Definitions of classes and properties can be accessed online at the URI specified above. For each relational database, a corresponding semantic web abs (S, D) is created, where S is the schema and D the instance of data representation.

Table 1. Defined Classes in the 'Abstract.owl' Ontology

rdf:ID	rdf:SubClassOf	rdfs:comment
abs:Schema	rdf:Bag	Class of Schemas(Databases)
abs:Class	rdf:Seq	Class of relations(Tables)
abs:Attribute	rdf:Resource	Class of properties(colomns)
abs:Identifier	rdf:Bag	Class of identifiers(PK)

Table 2. Properties Defined in the 'Abstract.owl' Ontology

rdf:ID	rdfs:domain	rdfs:range	rdfs:comment
abs:has	owl:Thing	owl:Thing	A Thing can have other Things inside
abs:hasClass	abs:Schema	abs:Class	A Schema has a set of classes
abs:hasAttribute	abs:Class or Identifier	abs:Attribute	A class has a set of attributes
abs:hasIdentifier	abs:Class	abs:Identifier	A class has an identifier
abs:hasObjects	abs:Class	abs:Class	A class contains others objects (references)
abs:length	abs:Attribute	xsd:nonNegativeInteger	A class has an identifier

3.2. Schema Extraction 'S':

Ontology

```
< . . .>
<owl:Classrdf :ID=''Author''>
<rdf :type rdf:ressource= ''&abs ;Class''/>
<abs :hasAttributerdf:ressource=''#Author.IdAuthor''/>
<abs :hasAttributerdf:ressource=''#Author.Name''/>
<abs :hasAttributerdf:ressource=''#Author.Affiliation''/>
<abs :hasObjectrdf:ressource=''#Author.Country''/>
<abs :hasIDrdf:ressource=''#Author.IdAuthor''/>
</owl : Class>

<owl : DatatypePropertyrdf :ID= ''Author.Name''>
<rdf :type  rdf:ressource= ''&abs ;Attribute''/>
<rdfs :domain  rdf:ressource= ''#Author''/>
<rsds :range  rdf :ressource=''&xsd ;string''/>
<abs :lenght>30</abs :lenght>
</owl :DatatypeProperty>
</ . . .>
```

Fig. 3. Sample instances of ontology representing the new 'Abstract.owl' ontology.

For its construction, the schema**S** contains a subclass of "Abstract" named **A,** referring to a domain represented in the database. For each relation R1…Rm in RDB, a subclass C1, ..., Cm of "Class" is created and included in S. The relationship between RDB and Ri is then added using the property “hasClass” in Class A. When creating our ontology, the tables are analyzed to identify their nature; as the structure of the table is not a table relationship, a subclass is created, otherwise the mapping procedure applies another treatment that will be discussed below. For data columns, a property is created and linked to the same class using the Object Propety “has Attri but”, as shown in Fig. 2 and Fig. 3.

For any primary key, a subclass of the class ‘Identifier’ is created and then connected to the reference class using the Object Propety ‘has Identifier’. The class ‘Identifier’ must be connected to appropriate attributes using the Object Property ‘has Attribut’. Fig. 4 shows an example of mapping a table using ‘Abstract

Cardinalities: For relationships between tables, we keep the reference fields by attributes while adding a link relationship between classes. This relationship is expressed by the Object Property“ has Object”, Fig. 5 shows the correspondence between a 1: n relationship and our ontology. For ‘N:M’cardinality, relationship between an entity R1 and R2 is represented by the creation of a relationship table R3with PK3 = FK (R1) U FK (R2), where PK3 is the primary key of the R3table and FK (R1) is the foreign key, which refers to table R1.FK (R2) of the foreign key references table R2. In such situations the R3 table represents a relation table that stores the links between records of R1 and R2; however, during the process of mapping, the structure of this table is translated by adding a Object Property to the classes C1 and C2 representing tables R1 and R1. Thenour scheme gives: C1 contains objects of C2 and C2 contains objects of C1. Some relationship tables can have additional data columns; a subclass is created to gather additional information on the relationship such as a date or other details. An example of such a situation is shown in Fig. 6.

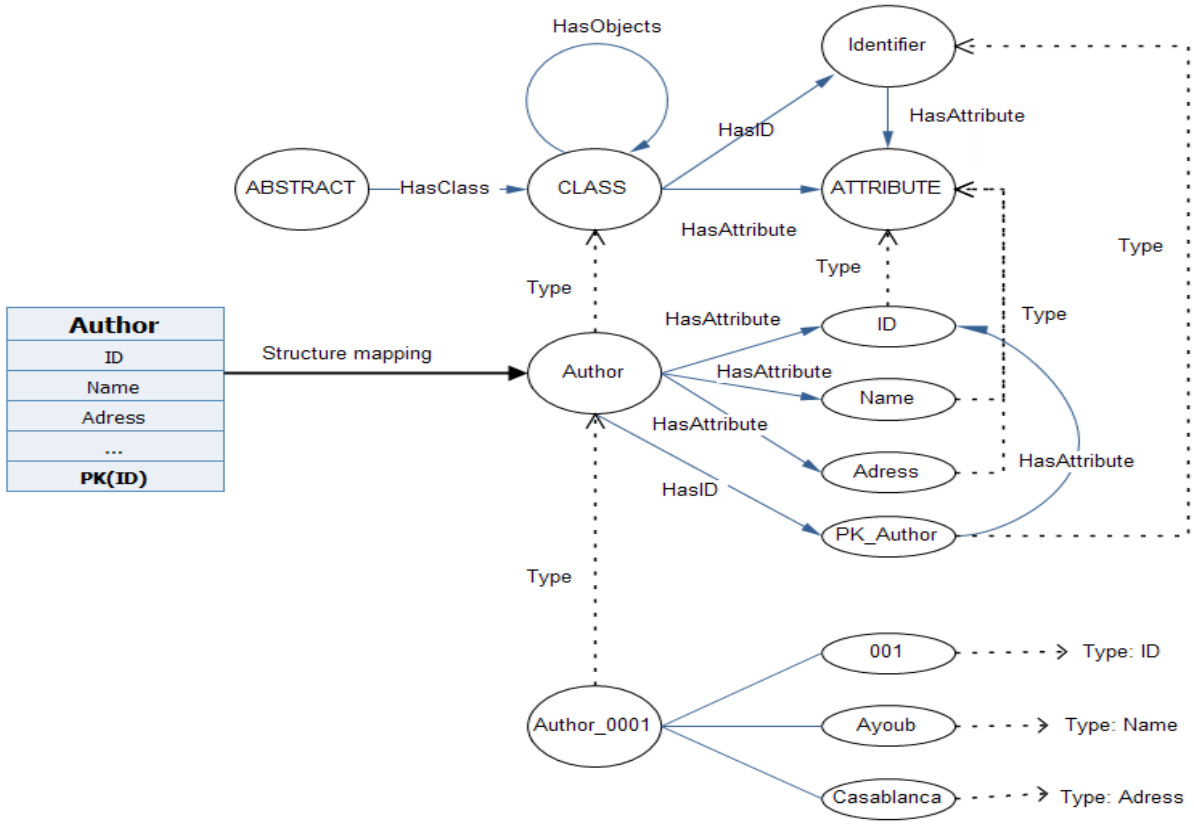

Fig. 4. Sample mapping a table using ‘abstract ontology’.

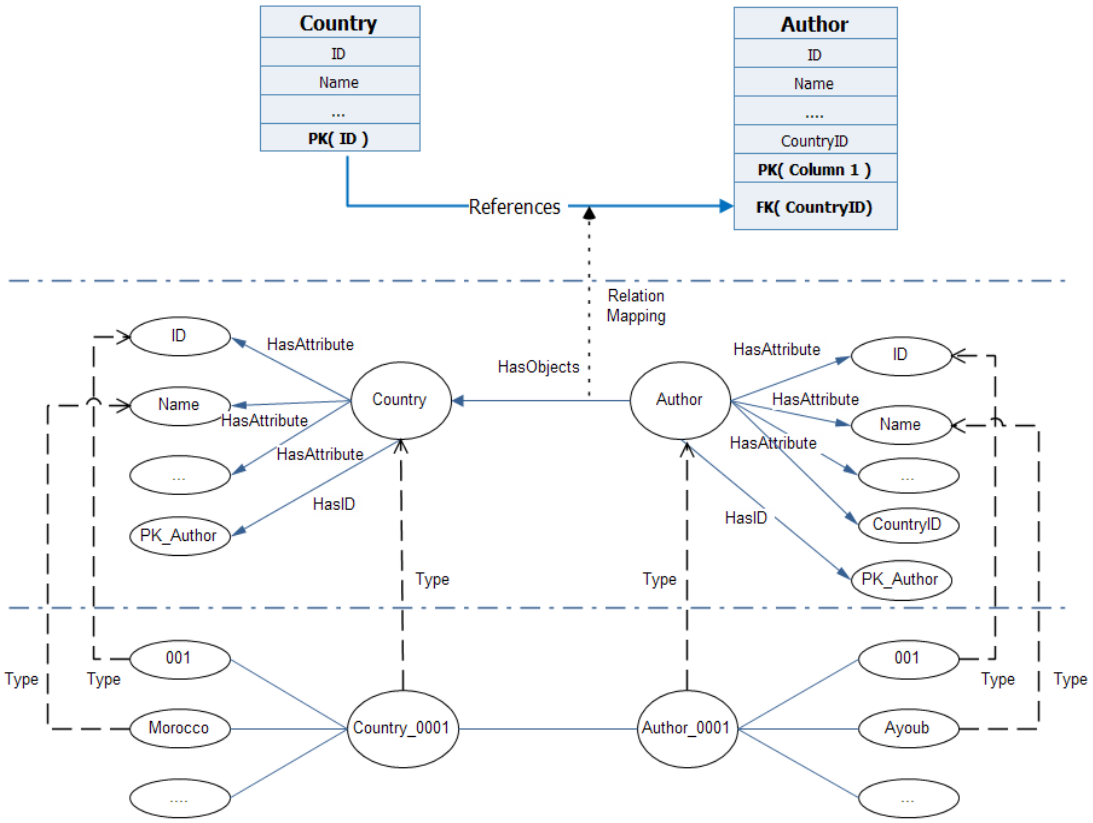

Fig. 5. Sample mapping a relationship 1: M bearer of properties.

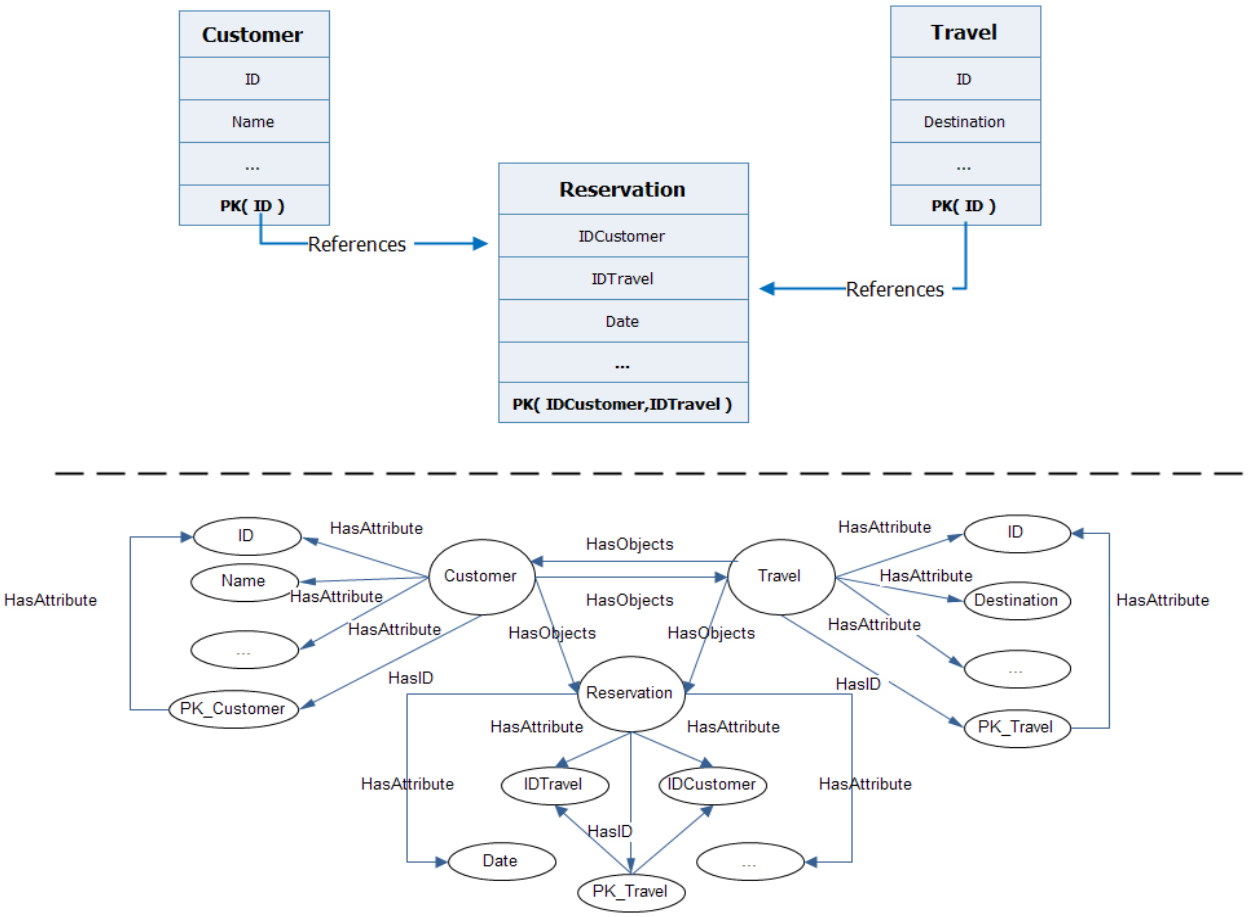

Fig. 6. Sample mapping a relationship N: M bearer of properties.

In that case, rdf: ID='Author' is equivalent to the table name in the original database. Each of the columns is defined using owl: Data type Property, where all the required properties are specified. The &abs; Class and &abs; Attribute objects are then connected using the abs: has Attri but property. The primary key of the table is represented by the abs: Identifier object and connected to the class representing the source table, using the property abs: has Identifier. For foreign keys, column reference is represented by the abs: has Objects property; this is so even for the type owl: Object Property, which connects the class to the reference object.

3.3. Data Extraction 'D'

After creating a representation schema of the relational database using OWL and 'Abstract.OWL' ontology, we can consider this representation to be a new ontology. With this relational database schema we are able to represent the data stored in the database. Consequently, these data can be represented as individuals in their own OWL schema. An example of data representation is shown in Fig. 7.

```
<... />
<rdf :Description rdf :NodeID=''Author_3248>
<rdf :type rdf :resource=''&schema ;Author.AuthorID>
<schema :Author.AuthorID>3248</schema :Author.AuthorID>
<schema :Author.Name>Ayoub OUDANI</schema :Author.Name>
<schema :Author.Affiliation>UniversitéHassan I,FSTS
</schema :Author.Affiliation>
<schema :Author.Countryrdf :resource=''Country_152''>
</rdf :Description>
<... />
<rdf :Description rdf :NodeID''Country_32''>
<rdf :type rdf :resource=''&schema ;Country''/>
<schema :Country.CountryID>32</schema :Country.CountryID>
<schema :Country.Name>Morocco</schema :Country.Name>
</rdf :Description>
<... />
```

Fig. 7. Sample nodes created in the RDF file after data extraction.

4. SPARQL Mapping

In this section, we present an example of mapping ontology extracted from the relational database to an existing ontology using a mapping language. This will be used to generate an RDF graph using the Construct clause. Because of its facility of use, based on SQL syntax and as potent as relational algebra in its expressiveness, we chose SPARQL. For this example we chose the vCard ontology as the target ontology for our mapping Fig. 8.

```
PREFIX vCard:<http://www.w3.org/2001/vcard-rdf/3.0#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX db: <http://localhost/RDF/schema.owl#>
CONSTRUCT { ?a rdf:type vCard:Individual;
                vCard:fn ?name;
                vCard:hasAddress :w;
             ?b rdf:type vcard:Adr;
                vcard:Country ?country}
WHERE { ?a rdf:type db:Author;
                db:Author.Name ?name;
                db:Author.Country ?b;
        ?b rdf:type db:Country;
                db:Coutry.Name ?country
                }
```

Fig. 8. Example of mapping using SPARQL query to the vCard ontology.

After defining prefixes vCard, rdf, and db in the PREFIX clause, the skeleton objects resulting RDF are defined in the Construct query part. First, a new anonymous node vCard:Individual is created. This object contains vCard:fn, vCard:has Address,.. attributs and could be easily extended to include other attributes, specified in the ontology vCard or from other RDF-Schema files. The values assigned to the given attributes are specified by the free variables contained in the WHERE clause as illustrated in Fig. 9.

```
<rdf :RDF
xmlns :rdf= ''http:/www.w3.org/1999/02/22-rdf-syntax-ns≠''
<rdf :RDF
xmlns :vCard=''http://www.w3.org/2001/vcard-rdf/3.0≠''>
<rdf :Description rdf :nodeID=''A0''>
<vCard :typerdf :resource=
''http://www.w3.org/2001/vcard-rdf/3.0≠Individual''/>
<vCard :fn>Ayoub OUDANI</vCard :fn>
<vCard :hasAdressrdf :resource=''A1''/>
<rdf :Description>
<rdf :Description rdf :nodeID =''A0''>
<rdf :typerdf :resource=
''http://www.w3.org/2001/vcard-rdf/3.0≠Adr''/>
<vCard :Country>Morocco</vCard :fn>
</rdf :Description>
</rdf :RDF>
```

Fig. 9. RDF graph result of mapping to vCard ontology.

5. Conclusion and Perspective

The results we have achieved by our prototype show the accuracy and performance of our approach. Indeed, we have shown that this approach effectively preserves the structure and data of a relational database and can be reused for a reconstruction of the source database. We have also shown, through an example, the ability to map the ontology of a relational database to an external ontology using SPARQL with our 'Abstract.OWL' ontology. This avoided join operations and provided optimization when mapping, unlike a Relational.OWL approach. Hence, we can say that we have presented an optimized and effective approach.

Also we have enriched it from the components of the RDB schema using different classes provided by the Jena API [12].

Our future work will address an automatic correspondence between the schema of a database and the target ontology in order to achieve full automation of the method proposed in this paper.

References

[1] Bedini, I., Matheus, C., Patel, S. P. F., Boran, A., & Nguyen, B. (2011). Transforming XML schema to OWL using patterns. *Proceedings of the 2011 IEEE Fifth International Conference on Semantic Computing* (pp. 102-109).

[2] Rodrigues, T., Rosa, P., & Cardoso, J. (2006). Mapping XML to exiting OWL ontologies. *Proceedings of the International Conference* of *WWW/Internet*.

[3] Cruz, C., & Nicolle, C. (2008). Ontology enrichment and automatic population from XML Data. *Proceedings of Joint ODBIS & SWDB workshop on Semantic Web, Ontologies, Databases.*

[4] Tang, J., Bringing relational databases into the semantic web. Tsinghua University, China

[5] Saleh, M. E. (January 2011). Semantic-based query in relational database using ontology. *Canadian Journal on Data*, *Information and Knowledge Engineering*, *2(1)*.

[6] Neumann, T., & Weikum, G. (2008). RDF-3X: a RISC-style engine for RDF. *Proceedings of the VLDB Endownment (PVLDB) (pp. 647-659)*.

[7] Chong, E. I., Das, S., Eadon, G., & Srinivasan, J. (2005). An E_cient SQL-based RDF querying scheme. *Proceedings of the 31st International Conference on Very Large Data Bases* (pp. 1216-1227).

[8] Sedighi, S. M., & Javidan, R. (2012). Semantic query in a relational database using a local ontology construction. *SAfr. J. Sci.*, *108(11/12)*.

[9] Bizer, C., & Cyganiak, R. (2006). D2r server-publishing relational databases on the semantic web. *Proceedings of the 5th International Semantic Web Conference*.

[10] Brizer, C., Cyganiak, R., Garbers, J., & Maresch, O. (2009). The D2RQ platform. Retrieved from the website: http://www4.wiswiss.fu-berlin.de/bizer/d2rq/

[11] Laborda, C. P., & Conrad, S. (2005). Relational owl-A data and schema representation format based on OWL. In S. Hartmann, & M. Stumptner (Eds.). *Proceedings of the Second Asia-Pacific Conference on Conceptual Modelling* (pp. 89-96), Newcastle, Australia.

[12] The Jena website homepage. Retrieved from the website: http://jena.apache.org/index.html

Ayoub Oudani is a PhD student in the Department of Mathematics and Computer Science, FSTS, University Hassan 1st, Settat, Morocco. His research interests include web ontologies and semantic web.

Mohamed Bahaj is a full professor in the Department of Mathematics and Computer Sciences, University Hassan 1st Faculty of Sciences & Technology, Settat, Morocco. He is a co-chair of the International Conference on Software Engineering, Databases and Expert Systems (SEDEXS'12), NASCASE'11. He has published over 60 peer-reviewed papers. His research interests are intelligent systems, ontologies engineering, partial and differential equations, numerical analysis and scientific computing.

Ilias Cherti is a full professor in Department of Mathematics and Computer Sciences, University Hassan 1st, Faculty of Sciences & Technology, Settat, Morocco. His research interests are intelligent systems, partial and differential equations. He has organized various national and international events.