A generic approach for data integration using RDF, OWL and XML

Miguel A. Macias-Garcia*, Victor J. Sosa-Sosa, and Ivan Lopez-Arevalo

Laboratory of Information Technology (LTI)
CINVESTAV-TAMAULIPAS
Km 6 Carretera Cd. Victoria - Monterrey
Ciudad Victoria, Tamps. C.P. 87276, Mexico
{mmacias,vjsosa,ilopez}@tamps.cinvestav.mx
www.tamps.cinvestav.mx

Abstract. This article describes the architecture of a data integrator that is proposed to support the development of information systems that require to extract and integrate information stored in heterogeneous data sources, such as relational databases, coma-separated value files (CSV) and XML documents. It is described an example step by step applying this integrator. It highlights issues such as modularity and simplicity, following a combination of an automatic and semi-automatic approach. The data integrator is a modular tool that takes advantage of the semantic Web technologies such as RDF, OWL, XML to tackle the problem of integrating heterogeneous data sources.

Key words: RDF, Ontology, XML, data integration.

1 Introduction

At present, the great development of Internet make de access to multiple information sources easier. Many merging companies need to integrate their various data sources in order to perform an analysis of the data with a global vision. Typically, these sources have different structures and semantics which complicate their integration. A situation found in a scenario like this is known as the problem of information integration. This problem has been addressed for a long time in the database area, without obtaining conclusive results [1]

Data integration offers, as a major advantage, an overview of the information, making transparent the complexity of accessing each data source separately, providing a comprehensive analysis of the information. The integration process usually is a handcrafted work, which makes it slow, error prone and inefficient. The next sections describe a proposal for a data integrator that supports the information integration process following a semi-automatic approach, a use case, and a final section presenting conclusions and future work.

* This research was partially funded by project number 51623 from "Fondo Mixto Conacyt-Gobierno del Estado de Tamaulipas".

2 Data integration process

The process of data integration has, in general, four main phases (Figure 1, adapted from [1]):

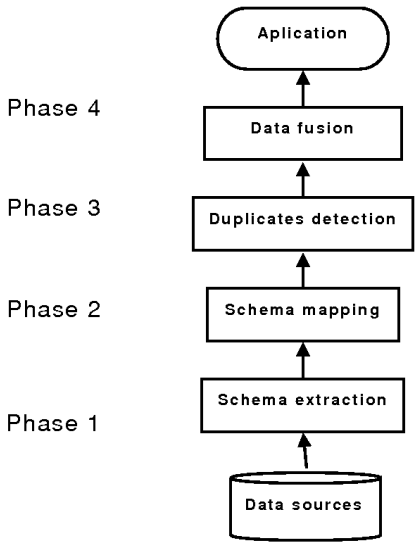

Fig. 1. Data integration process

Schema extraction: The process of integration requires to obtain the structures or metadata from the involved data sources. These metadata are usually extracted, for instance, from the database (DB) dictionary, the header of a coma-separated value (CSV) file or from XML schema definition (xsd) files. They are useful to describe the data sources content. After the extraction of schemas or descriptions, they need to be rewritten using a formal language e.g. XML-RDF. Describing schemas in a formal and homogeneous language allows machines to better understand and manipulate those schemas

Schema mapping: The schema mapping process tries to merge the schemas of different data sources to outline a global schema that represents the integrated information. Schema mapping assumes a given objective scheme, whereby a set of combined heterogeneous sources are perceived as an integrated information system.

There are various techniques that support the schema mapping process, also known as the schema matching process [2]. The match operation is defined as the function that takes two schemas S1 and S2 as input and gives as output a

mapping between the two schemas, called the outcome of the correspondence. Each element specifies a mapping relationship between any of the elements of the source S1 with elements of the source S2. There are many issues to be considered in this process, some of them are: attribute name similarity, data type similarity, semantics, etc.

Duplicates detection: The next step in the process of data integration is the duplicates detection, also called records linkage or references reconciliation. This stage identifies multiple representations of the same object in the real world. This detection is performed by comparing each pair of objects of different sources, looking for a level of similarity to establish a threshold, where the threshold is exceeded then we found duplicate items.

At this stage two metrics can be evaluated, effectiveness and efficiency. The effectiveness is given by the quality of the similarity algorithms and the threshold parameter used to calculate the similarity. Efficiency is determined by the size of the data set and the runtime similarity algorithm. The result of the duplicate detection process is the assignment of an identifier for each object representation. Two representations with the same identifier indicate duplicity.

Data fusion. The final step is to make the union of the records obtained through a global schema in the schema mapping. This process includes attributes renaming, restructuring and detection of inconsistencies.

As we can see, the integration process includes various stages that lead to a complete data integration. In this sense, our work proposes to start with a schema extraction phase. The extracted schemas are initially represented in XML and then turned into triplets of the RDF language. The OWL language is used to describe the relationships between elements of the involved schemas and facilitates the creation of a global schema, where queries, looking for global information, can be executed accessing data regardless of the involved data sources. Our data integrator gives support to develop phase 1 and 2 (see Figure 1), following an automatic approach and offers a semi-automatic approach for phase three and four.

3 Semantic Web

The evolution of the Web has brought new technologies that make possible to understand information not only for humans but also for machines. The introduction of semantic representations in information allow machines to find and relate it more efficiently. This form of representing the Web is known as the Semantic Web. Such technologies also can be used as a new way to represent information in data sources that manage heterogeneous structures. This situation motivates us to include these technologies as a fundamental part of our data integrator, facilitating the process of data integration in a semi-automatic way.

The following elements are the core of these technologies.

RDF: It is a Web Consortium Specification(W3C) that allows the description and conceptual modeling of information in the form of metadata. It describes resources, specially Web resources, through the form of sentences with subject-predicate-object, which are known as triplets. The subject represents the resource, the predicate the characteristic or feature of the resource and the relationship it has with the object. RDF provides interoperability between applications that exchange machine readable information. It is highlighted by the simplicity of enabling automatic processing of Web resources.

RDF allows to define a mechanism for describing resources that do not create any assumption about a particular application domain, nor defines (*a priori*) the semantics of an application domain. The definition of the mechanism must be neutral with respect to the domain. However, the mechanism should be suitable for describing information about any domain.

Ontology: In philosophy, ontology is "the study of being ", a general branch of metaphysics that deals with defining what exists and to establish basic categories for all things according to their properties. In Artificial Intelligence, Ontology refers to an explicit representation and is expressed in a formal language of the conceptual model of a certain domain of knowledge. Gruber gives the following definition: "A formal and explicit specification of a shared conceptualization. This is a description of the concepts and relationships that can exist for an agent or a community of agents "[3]. The sharing feature makes possible to reuse the semantic vocabulary of an ontology in the construction of different knowledge bases and even different intelligent systems, while they are working on the same domain of knowledge.

OWL: A language defined by the Web Consortium (W3C) that can be used to explicitly represent the meaning of terms in vocabularies and the relationships between them. This representation of terms and their interrelationships are called Ontology. OWL has a greater capacity for expressing meaning and semantics than XML, RDF, and RDF-S, and thus OWL goes beyond these languages in its ability to represent content interpretable by machines.

In short, the Semantic Web technologies provide us with mechanisms to give explicit meaning to information so that machines can automatically process and integrate it. These technologies take advantage of the XML ability to define customized schemas/tags, the RDF flexibility to represent data and the OWL to describe relationships among data.

The top abstraction level in the information integration process is built on an ontology language, which can formally describe the meaning of the terminology used in the schemes extracted from the data sources. If machines are expected

to do useful reasoning tasks with these schemes, the language should go beyond the semantics of the basic RDF schema known as RDF-S. Reason why OWL is proposed as a good solution.

4 Data integrator architecture

Figure 2 shows the process flow diagram that is followed in the data integrator.

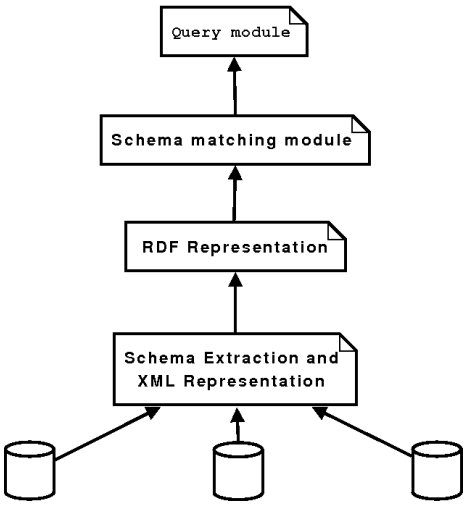

Fig. 2. Data integrator process flow

Schema/data extraction and XML representation module. The first step in our data integrator includes a module to obtain the schemas from different data sources. Our data integrator includes wrappers for each type of data source such as comma-separated value files (CSV), relational DBs and XML files. The DB wrapper includes different DB connectors, which allow it to get access to several SQL databases. The XML wrapper reads the xsd files that describe the information included in the XML data source. The CSV wrapper can read special CSV files, which include a header that defines the CSV file content. The *generate-mapping* tool from the *d2rq* platform [4] is used by the DB wrapper to extract DB schemas and store them in XML files that include a header with the DB metadata and a body with the data. This tool translates the DB schemas into text files having a ttl extension (Figure 3). These special ttl files will be then the input data for the *dumprdf* tool [4]. This tool will be used in the next step.

The CSV wrapper uses the JDOM tool [5] for translating CSV files into XML files. A similar process could be done with the XML files, however this it is no necessary.

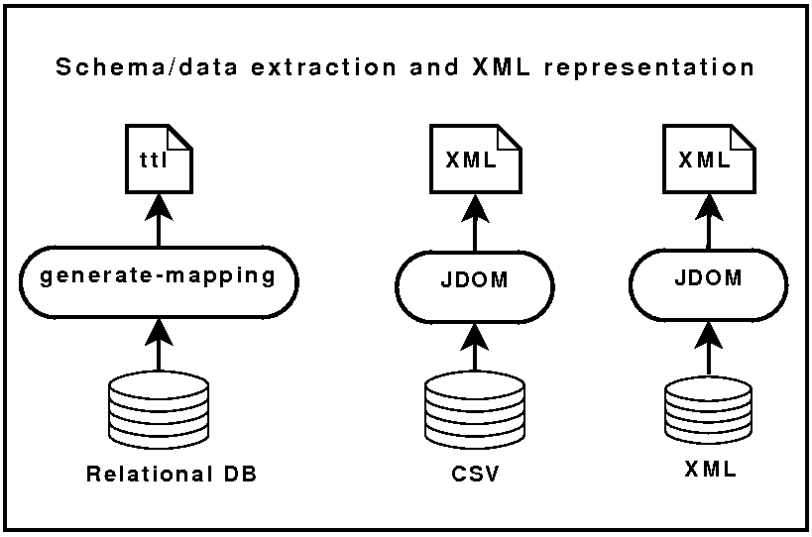

Fig. 3. Schema extraction and XML representation

RDF Representation. The next step in the data integrator is to convert XML documents into RDF documents. This conversion can be achieved by using the *datadump* command that is included in the *d2rq* platform (in cases when the databases are relational). The *datadump* tool translates ttl files into RDF files (Figure 4). The XML files obtained from the CSV and XML wrappers are turned into RDF files by using the XSLT tool [6].

Schema matching. The process of schema matching uses the names of the attributes of the data sources that were obtained in the first phase and a domain dictionary that contains stem words, synonyms and hyperonyms.

The output of this phase will be an OWL file, generated from the imported RDF files (data sources). This new OWL file includes the complete data and a global ontology that represents the global shema of the integrated data. The following steps describe the construction process of the global ontology.

The firs step is to compare the attribute names of the different extracted data sources with the stem words. Every match that is found generates two new definitions in the OWL file:

1. The attribute definition in the global schema, e.g.

```
<owl:DatatypeProperty rdf:about="global_(stem word)"/>
```

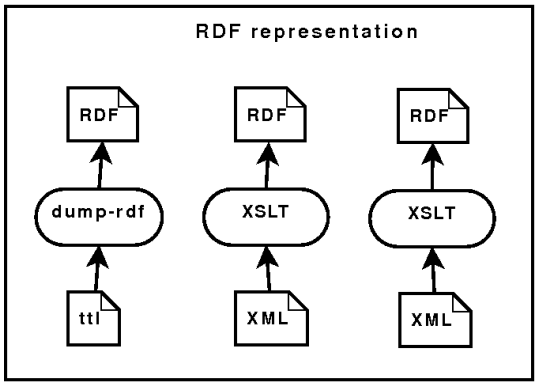

Fig. 4. Converting XML data to RDF

2. The equivalence relationship between the original attribute (attribute of the data source) and the global attribute (attribute included in the global schema) e.g.

```
<owl:DatatypeProperty rdf:about="global_(stem word)">
<owl:equivalentProperty rdf:resource="(domain)_(attribute)"/>
</ owl: DatatypeProperty>
```

If an attribute does not match with a stem word, a search in the synonyms and hyperonyms bases is triggered.

If a match is found in th synonyms base, the stem word (of that attribute) is obtained and a new equivalence relationship is generated.

If the match is in the hyperonyms base a new subproperty relationship is created. An example of subproperty relationship is as follows.

```
<owl:DatatypeProperty rdf:about="domain/(attribute)">
<rdfs:subPropertyOf
rdf:resource="http://localhost/global_(stem word)"/>
 </owl: DatatypeProperty>
```

If there is no match with the stem, synonyms or hyperonyms word, the Jaro Winkler distance [7] is applied between the stem word and the attribute name. When the distance exceeds a defined threshold (0.8), an equivalence relationship is automatically generated. Otherwise the original attribute name is included in the global schema.

Query module. At this time, the final information base is a combination of all triplets and the global ontology. This information could be queried through a sparql [8] module (Figure 6). For this reason, the data integrator includes the use of a sparql engine called *pellet* [9] that takes advantage of the ontology characteristics included in the RDF document provided by the OWL language.

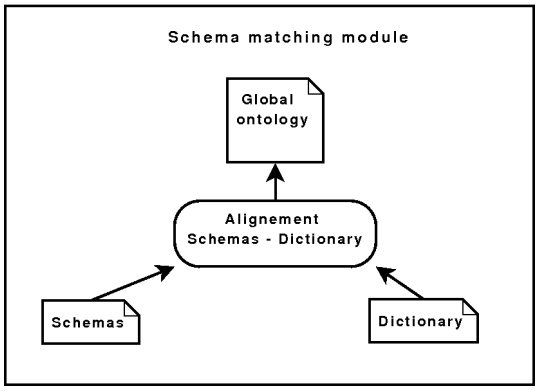

Fig. 5. Schema matching using a dictionary

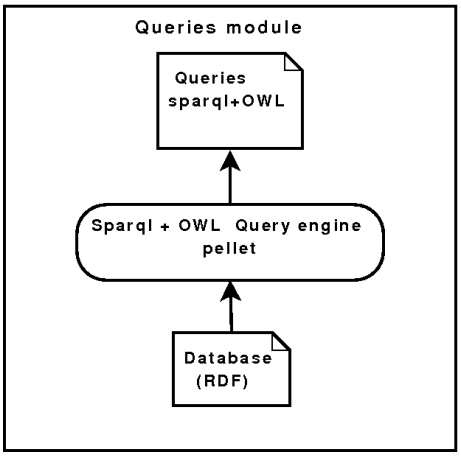

Fig. 6. Making queries to RDF data base.

5 Use case

A simple use case is presented, wherein have two data sources with heterogeneous structure and schema: A relational database and a XML file. The process begins with the extraction of the schemas. A reduced representation of the data sources storing information of patients in different hospitals is presented in Figure 7.

The schemas obtained automatically are described in XML files (step 1) and afterward these files are turned into RDF files, using the *datadump* command in cases where the data source come from a relational database or using XSLT when the data source comes from a XML file. After this translation, the process of matching between the dictionary and the obtained schemas generates a global

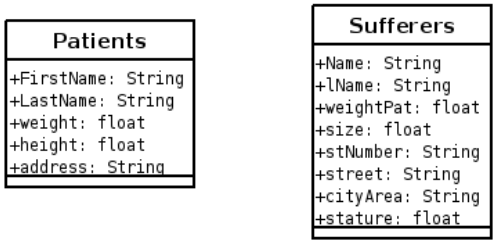

Fig. 7. Tables of patients taken from two differents data sources.

schema as an ontology in OWL, wherein the involved sources are imported. The following example shows some statements of the resulting global ontology.

- `Patients.FirstName is equivalent to Sufferer.Name`
- `Patients.LastName equivalent to Sufferer.lName`
- `Patients.weight is equivalent to Sufferer.weightPat`
- `Patients.height is equivalent to Sufferer.stature`
- `Sufferer.street is subproperty of Patients.address`
- `Sufferer.stNumber is subproperty of Patients.address`
- `Sufferer.cityArea is subproperty of Patients.address`

Finally, we build a global RDF data base that combines the content of the two data sources. The schema that describes this RDF data is the global ontology. With the scheme represented as Ontology and the data represented in RDF, it is possible to query the global information using the *sparql* language. In this way, we obtain transparency at the moment of accessing all the mixed information, thanks to the automatically generated OWL rules.

An example of this transparency can be seen in the following query: What is the address of the patient Armando Perez? For answering this query using a SQL statement in a relational data base, it is necessary to specify the name of the attribute "address "and ignore the fact that attributes such as street, stNumber and cityArea together represent an address. Since the OWL language allow us to define rules such as is-equivalent-to, or is-subproperty-of, a sparql query looking for the patient address in different hospitals does not need to say explicitly the name of the possible set of attributes that could describe an address. The following sparql statement can answer the mentioned query.

```
PREFIX pat: <file://Sufferer/> SELECT ?Subdir ?address
WHERE ( ?s  ?Subdir ?address.?s pat:address ?address.
?s  pat:Sufferer_lName "Perez."   ?s  pat:Sufferer_Name"Armando.")}
```

The figure 8 shows an instance of the different RDF databases where can be seen the information that the query refers.

```
          <rdf:Description rdf:about="#sufferer/Armando/Perez">
          <j.0:sufferer_name>Armando</j.0:sufferer_nombre>
          <j.0:sufferer_lname>Perez</j.0:sufferer_apellido>
          <j.0:sufferer_cityArea>Portillo</j.0:ufferer_cityArea>
          <j.0:sufferer_street>Bagdad</j.0:sufferer_street>
          <j.0:sufferer_stNumber>428</j.0:sufferer_stNumber>
-------------------------------------------------------------------------------
  <rdf:Description rdf:about="#patients/Armando/Perez">
  <j.0:patients_FirstName>Armando</j.0:patients_FirstName>
  <j.0:patients_LastName>Perez</j.0:patients_LastName>
  <j.0:patients_address>Bagdag 428, Portillo City Area</j.0:patients_address>
```

Fig. 8. An instance from RDF databases

We could obtain, for example, the following results produced by this query, considering that the patient information is stored in the two data sources using different structures.

```
Subdir                | address
=======================================================
pac: Sufferer_cityArea| "Portillo"
pac: Sufferer_street  | "Bagdad"
pac: Sufferer_stNumber| "428"
pac: address          | "Portillo"
pac: address          | "Bagdad"
pac: address          | "428"
pac: address          | "Bagdag 428, Portillo City Area"
```

The result of this query shows how the subproperties of the address attribute are included in the answer without explicitly be mentioned in the query. This and other properties can be viewed using the pellet query engine.

6 Conclusions

This article has briefly described the process of data integration, we also propose a data integrator that uses semantic Web technologies to enrich and integrate the information stored in different data sources. Our data integrator align the terms (stem words, synonyms and hyperonyms) in a dictionary with the attributes names creating a global ontology that will represent all of them. Following this global ontology, the data integrator tries to merge data coming from the different sources. A final information base is created. It includes the global ontology that makes possible to query the complete information, conciliating the differences related to structures and semantics.

References

1. Bleiholder, J., Naumann, F.: Data fusion. ACM Comput. Surv. **41**(1) (2008) 1–5

2. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB Journal: Very Large Data Bases **10**(4) (2001) 334–350
3. Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acquisition **5(2)** (1993) 199–220
4. Bizer, C., Seaborne, A.: D2rq - treating non-rdf databases as virtual rdf graphs. In: Proceedings of the 3rd International Semantic Web Conference. (2004)
5. Harold, E.R.: Processing Xml with Java. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2002)
6. OCLC: The open source xsltpro. http://www.oclc.org/research/software/xsltproc/default.htm (may 2009)
7. Winkler, W.E.: The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau (1999)
8. Prud'hommeaux, E., Seaborne, A.: Sparql query language for rdf. http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/ (May 2009)
9. Evren Sirin, Bijan Parsia, B.C.G.A.K., Katz, Y.: Pellet: A practical owl-dl reasoner. Web Semantics: Science, Services and Agents on the World Wide Web **5** (2007) 51–53