

Two Layer Mapping from Database to RDF

Martin Svihla, Ivan Jelinek

Department of Computer Science and Engineering
Czech Technical University, Prague, Karlovo namesti 13, 121 35 Praha 2, Czech republic
E-mail: svihlm1@fel.cvut.cz, jelinek@fel.cvut.cz

Abstract

There is a huge mass of data stored in relational databases. Large part of these data is exposed on the web in HTML presentations. The idea of the Semantic Web is to add computer-readable meaning to data on the web to extend its possibilities. The automatic generation of Semantic Web content from relational databases is not new issue, but current approaches are not directly usable for common programmers of web presentations. This fact may inhibit wide spread of Semantic web concept. In this paper we present *METAmorphoses* - two layer model of ontology based mapping of relational database content to RDF as a proposal of easy-to-use solution.

1. Introduction

It is well known that the World Wide Web has become a huge source of information. These information are difficult to search or maintain not only because of their quantity but also because they are mostly presented only for humans to read and understand. Semantic Web is an initiative, which tries to give more structure and computer understandable meaning to the data on the web. This extension of current web should enable computers and people to work in better cooperation [1].

The three basic technologies for the Semantic Web are XML (eXtensible Markup Language) [6], RDF (Resource Description Framework) [7] and ontologies. Language of metadata is RDF, which uses XML tags to write triples. By these triples, a semantic is formulated like an elementary sentence consisting of subject, verb and object. In this way we can make assertions that a subject, e.g. 'person', has a property, e.g. 'is a friend of', with a particular value, such as another 'person'. Ontologies were added to the concept of the Semantic Web to formally describe terms used in metadata. Ontology defines terms used in RDF document, relations between them and also provides inference rules, which allows a software agent to deduce new facts from those asserted in metadata.

Success of the Semantic Web also depends

on mass creation of semantic data, that are expected to cover existing Web by machine-readable meaning [3]. To reach this goal, several approaches were designed. Some of them assume that data sources are static and propose manual or semiautomatic annotation of existing HTML presentations. However, the most of web content today is generated dynamically from relational databases. That means that mapping from database schema directly to RDF is much faster, cheaper and usually more suitable for generation of Semantic Web content.

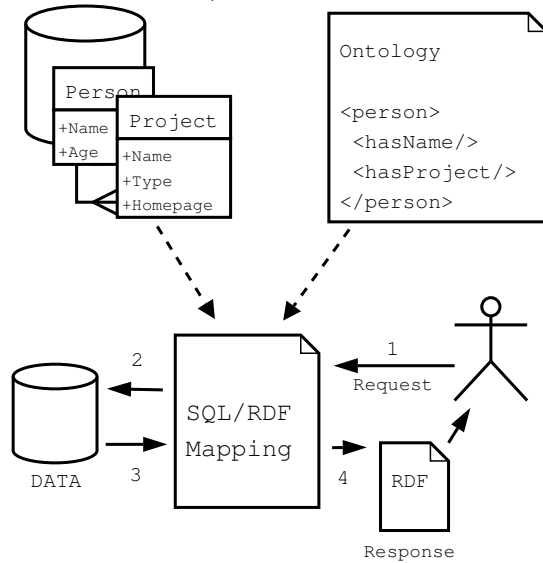
In our work we propose approach to such mapping, which is focused on usability.

2. Current solutions of mapping SQL to RDF

There are two main approaches of dealing with relational database in order to present data in RDF language [8]. In the first case data are stored in a database like RDF triples and there is system, which can query these data directly in RDF way. Other approach is when data are stored in classic relational schema and there is some mapping to RDF. As far as the most of data today are stored in classic relational schema, our work is focused to the latter approach.

Figure 1 illustrates common logic of SQL

Fig. 1: RDF/SQL mapping schema



to RDF mapping. Mapping is created according to SQL structure and ontology. When there is a request for RDF, it is translated to SQL query. Results of query are translated to RDF and this is sent as response.

There are some existing projects solving the question of SQL/RDF mapping (e.g. [5], [4], [3] and others). They do what they were designed for, but they don't make RDF creation much more easier for common programmers.

An ideal providing RDF information is as simple as providing HTML [2] is still far away. As an answer to this challenge, we propose *METAmorphoses*, database to RDF mapping design for easy use.

3. *METAmorphoses* - proposal of two layer mapping

In this section we present *METAmorphoses*, two layer model of SQL to RDF mapping based on given ontology. This concept is designed to allow flexible mapping and easy production of RDF.

3.1 Problem overview

Two main presumptions in our work are: data are stored in 'classical' relational database

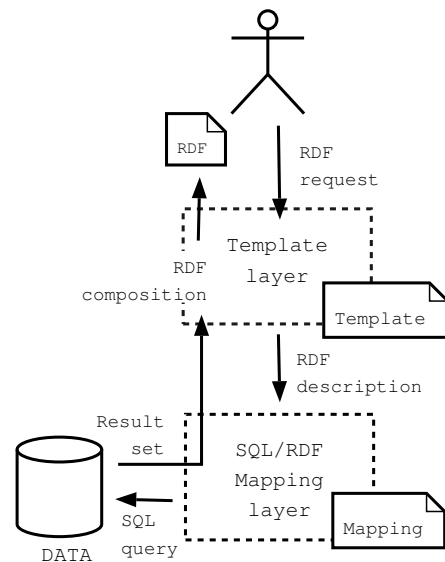
schema and we have an ontology that specifies format of RDF metadata we want to produce from the database. In this paper we propose a system which maps data from the database to an RDF document. This system should be simple and efficient. Our main goal is to design system providing simple programming interface, so that programmers who creates web presentation can compose RDF fragments of document as easy as they composes HTML fragments using Java servlets, JSP, ASP or PHP.

3.2 System architecture

In order to achieve flexible mapping and high usability, we divided *METAmorphoses* logic into two separated parts that we call mapping and template layer. The mapping layer maps RDF classes to SQL structures. The template layer uses this mapping and produces RDF instances in the way which is driven by a template.

Both layers are described in following sections.

Fig. 2: *METAmorphoses* architecture



3.3 Mapping layer

This layer is responsible for mapping of database content to RDF document accord-

ing to given ontology. The core of the layer is simple SQL to RDF mapping language. Language is slightly inspired by D2R MAP [4], but it can also serve for other purposes than mapping according to the system design.

Listing 1: Mapping language sample

```
<Mapping>
  <Class templateName="person"
    rdfLabel="foaf:Person" sql="
      select * from person">
    <ClassCondition templateName="
      username" whereString="
        username ="/>
    <Variable templateName="personId"
      sqlName="id"/>
    <Property templateName="
      firstName" rdfLabel="foaf:
        firstName" sqlName="
          first_name"/>
    <Property templateName="surname"
      rdfLabel="foaf:surname"
      sqlName="family_name"/>
    <Property templateName="
      currentProject" rdfLabel="
        foaf:currentProject" sqlName
          ="/>
  </Class>

  <Class templateName="project"
    rdfLabel="foaf:Project" sql="
      select * from project p,
      person_project pp where p.id =
      pp.project_id">
    <ClassCondition templateName="
      personId" whereString="pp.
        person_id ="/>
    <Property templateName="
      projectHomepage" rdfLabel="
        foaf:homepage" sqlName="web
          "/>
  </Class>
</Mapping>
```

Mapping language is based on XML. It is designed to map classes of given ontology to database structure, as it can be seen from sample code fragment (listing 1). This means that we create a mapping only for RDF classes we are going to need, so we don't have to map necessarily all database structures.

Each class contains SQL statement and optionally set of possible properties of the class, conditions and variables. Properties of `class` tag in mapping stands for RDF properties in ontology, conditions and variables controls RDF generation and are used in tem-

plate layer.

To establish a communication with the template layer, there are attributes called `templateName` in each of the elements. Mapping elements contain also particular RDF tag names. That means RDF generation can be processed without support of any RDF API. This is important efficiency factor.

Once a proper mapping is created, it preserves consistency of generated RDF according to given ontology.

The mapping contains also information about database connection and information about namespaces in produced RDF.

3.4 Template layer

In *META*morphoses design, template layer produces RDF instances according to information in mapping layer. It can call all elements of mapping in order to build RDF document. This control is done by another XML document - template. This template is written by programmer of Semantic Web presentation. In template is specified how an RDF instance will be composed, which properties it will contain and so.

Listing 2: Template language sample

```
<Template>
  <putInstance name="person" id="1">
    <Condition name="username">
      svihlm1</Condition>
    <putProperty name="firstName"/>
    <putProperty name="surname"/>
    <putProperty name="
      currentProject">
      <putInstance name="project">
        <Condition name="personId">
          <Variable id="1" name="
            personId"/>
        </Condition>
        <putProperty name="
          projectHomepage"/>
      </putInstance>
    </putProperty>
  </putInstance>
</Template>
```

Each element in template has an attribute `name` which corresponds with `templateName` in mapping. This is the way both layers are linked.

Template contains tags `putInstance` and `putProperty` to put fragments of RDF produced by mapping layer to final RDF document. However, rules how to use these tags are stored in mapping. Tags `Condition` and `Variable` are control elements that controls using of the previous two tags. For example, following fragment writes to RDF document all persons from database with their first name.

```
<putInstance name="person">
  <putProperty name="firstName"/>
</putInstance>
```

In the next modification there is the condition that reduces selected persons to one with username *svihlm1*. Tags `putProperty` says that first name and surname will be written in person RDF element.

```
<putInstance name="person">
  <Condition name="username">svihlm1
  </Condition>
  <putProperty name="firstName"/>
  <putProperty name="surname"/>
</putInstance>
```

Content of condition tag can be literal, as in listing above, or `Variable` tag, as in listing 2. Variables have `id` attribute for referring to corresponding instance (see listing 2). This is the way *METAmorphoses* can handle more complex constructions.

The result from mapping (listing 1) and template (listing 2) samples will be following piece of RDF document.

Listing 3: Sample RDF fragment

```
<foaf:Person>
  <foaf:firstName>Martin</
    foaf:firstName>
  <foaf:surname>Svihla</foaf:surname
  >
  <foaf:currentProject>
    <foaf:Project>
      <foaf:homepage>http://www.cgg.
        cvut.cz/~svihlm1/
        metamorphoses/</
        foaf:homepage>
    </foaf:Project>
  </foaf:currentProject>
</foaf:Person>
```

As far as the language used in template layer is also XML-based, programmer can

compose RDF document easily using template tags like they were fragments of RDF. No RDF query language is needed nor any additional RDF API.

This means the template can be included for example directly to JSP page by JSP custom tags. Then template tags can be combined with JSP tags in order to dynamic publishing of RDF document.

3.5 Mapping process

Process of mapping of database structure to RDF in *METAmorphoses* is simple. However, due to two layer design it has two phases. The first one is creating mapping document. This should contain all RDF classes with their properties how they are defined in particular ontology. These classes should be connected with mapping variables and conditions. When the mapping is complete, the simple list of its elements can be created to help with template building.

The second phase is writing a template. Each template contains reference to corresponding mapping and works with elements defined in it to create RDF fragment. It is possible to create many templates upon one mapping. The template can be contained in JSP page as part on (X)HTML document, or can be called from servlet container to produce standalone RDF document.

3.6 RDF production

Mapping and template documents are processed by *METAmorphoses processor*, which is written in Java. Processor is also divided into two logical parts - a mapping processor and a template processor.

When request for RDF is received, processor finds proper piece of template. Then it loads corresponding mapping, connects database and creates list of used classes.

In the next step template is processed hierarchically, considering conditions and variables to query database for data, which are used to build RDF fragments. In this phase an URI is created for RDF instances.

Once the template processing is over, RDF is sent to server as response.

4. Discussion and results

This paper discussed only basic ideas of *METAmorphoses* design. Some issues were not mentioned here, for example a way how the URI is created in RDF.

Main advantages of proposed concept are flexibility in database mapping and simple programmer interface. Flexibility means that the mapping language of *METAmorphoses* is capable to capture any ontology. However, it is not practical to write very complex constructions manually. For this purpose a mapping tool is planned that would ensure ontology as well as SQL integrity. This integrity is useful, because when the proper mapping is done, validity of RDF output is ensured without using additional RDF software packages.

Simple programmer interface of *METAmorphoses* is realized by templates. They are XML-based and can be part of e.g. JSP page, so that production of RDF fragment is as easy as production of HTML. And, in addition to that, as far as validity of RDF is guaranteed by mapping layer, one who writes template doesn't have to know anything about ontologies.

Question of consistency of our mapping model was also taken into account. RDF data model is based on triple schema of subject (RDF class), verb (RDF property) and object (literal or another RDF class). As far as our mapping schema allows including instances in properties of other instances, it can be seen as directed graph of RDF triples.

Of course, *METAmorphoses* model still has weak points. Due to two layer structure, it is possible that if ontology was updated, not only mapping, but also templates would have to be changed. The question of optimal database querying was not also taken into account during mapping language design.

We have done some tests yet with *METAmorphoses* on an experimental set of data and simple ontologies. Promising results of these experiments shows that our concept fulfills

our expectations.

5. Conclusion and future work

In this paper we have presented *METAmorphoses*, two layer mapping model of relational database schema to RDF according to given ontology. This approach is meant as solution, that should simplify work of authors of Semantic Web presentations.

It is necessary to mention that *METAmorphoses* is not an implementation issue. It is combination of two simple XML languages that cooperates in order to map SQL structure into RDF elements and to publish produced RDF easily. *METAmorphoses* processor is deployed just as proof-of-concept of our idea and testing tool for our work.

In future we plan implementation of the mapping tool for more complex mappings. Then we want to focus on testing our concepts in 'real' environment.

We hope that results from real-world experiments would be accomplished very soon, because they would help to improve design and implementation of *METAmorphoses*.

References

- [1] Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American, May 2001
- [2] Hjelm, J.: Creating the Semantic Web with RDF. Wiley Computer Publishing, New York, 2001
- [3] Handschuh, S., Staab, S., Volz, R.: On Deep Annotation. Proceedings of the 12th International World Wide Web Conference, WWW 2003, ACM Press, 2003
- [4] Bizer, Ch.: D2R MAP - A Database to RDF Mapping Language. Proceedings of the 12th International World Wide Web Conference, WWW 2003, Budapest, Hungary, 2003
- [5] Stojanovic, N., Stojanovic, L., Volz, R.: A reverse engineering approach for migrating data-intensive web sites to the Semantic Web, IIP-2002, Montreal, 2002
- [6] Extensible Markup Language (XML). March 2004, <http://www.w3.org/XML/>

- [7] Resource Description Framework (RDF).
March 2004, <http://www.w3.org/RDF/>
- [8] Beckett, D., Grant J.: Mapping Semantic
Web Data with RDBMSes. March 2004,
[http://www.w3.org/2001/sw/Europe/
reports/scalable_rdbms_mapping_report/](http://www.w3.org/2001/sw/Europe/reports/scalable_rdbms_mapping_report/)