From Relational Data to RDFS Models

Makym Korotkiy and Jan L. Top

Vrije Universiteit Amsterdam, Department of Computer Science, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{maksym, jltop}@cs.vu.nl

Abstract. A vast amount of information resources is stored as relational-like data and inaccessible to RDFS-based systems. We describe FDR2 – an approach to integration of relational-like information resources with RDFS-aware systems. The proposed solution is purely RDFS-based. We use RDF/S as a mechanism to specify and perform linking of relational data to a predefined domain ontology. The approach is transformation-free, this ensures that all the data is accessible and usable in consistence with the original data model.

1 Introduction

The RDF and RDFS languages have been developed to express machine understandable semantics to facilitate more intelligent ways of information processing. RDF/S languages provide a unified syntax, data model, well-defined semantics and enable separation of data (RDF) from meta-data (RDFS). The formal acceptance of RDF/S by W3C [1] stimulates their utilization in many areas and by many organizations.

In spite of an increasing acceptance of RDF/S, this is still a new technology. Most information resources are not available in RDF/S-format. The relational data model, on the other hand, is widely accepted and currently supported by thousands of applications ranging from simple spreadsheets to complex relational databases.

Within this paper we use *relational data model* to refer to data presented as a collection of records usually depicted as a table. We would like to note that within the paper we differ between *relational data model* and *relational database model*. The latter extends the former by assuming that columns represent attributes, rows represent entities, and the table contains a set of attributes uniquely identifying each entity. We did not accept such assumptions because our experience indicated that very often the users organize data in an "intuitive" tabular way supported by spreadsheet applications but incompatible with these database-specific assumptions. This made us to focus on the less restricted *relational data model*.

Our application interest is to bring RDF/S technology to R&D companies and institutes as a part of what has been labelled as e-Science. The idea is that results of scientific experiments and computations as such should be shared within the (global) scientific community, in addition to communicating through

N. Koch, P. Fraternali, and M. Wirsing (Eds.): ICWE 2004, LNCS 3140, pp. 430–434, 2004.

traditional scientific publications. The latter do not contain sufficient details of the work done and the information presented by them cannot be machine processed.

Sharing is to be supported by an ontology-based information system. The primary goal of the system is to assist with the transition from traditional experimental science to e-Science facilitating large scale collaboration between scientists. Since we intend to bring benefits of ontologies into the e-Science environment, we have to find a way to link the relational data to an RDF/S model.

The main objective is to allow ontology-based querying of the relational data: the original relational data must be made available to an RDFS reasoner and become queryable with a vocabulary predefined in a domain ontology.

Our approach to the linking problem is explained in Sect. 2. Section 3 discusses the presented method and related research and Sect. 4 summarizes the results of this work.

2 *FDR2* Approach

We present our approach as a general procedure that can be modified and extended to fit particular needs. Let us assume that we have a set of data, expressed in a relational way (e.g., as a spreadsheet table where the first row is a header) and a domain ontology (DO) expressed in RDFS.

The general problem of linking relational and RDF/S models can be broken down into two subproblems:

1. Expressing relational data in an RDF/S format to enable syntactic and structural interoperability with DO.
2. Linking the newly created serialization to the RDFS representation of DO.

We approached the first subproblem by performing a two-step serialization of the relational data. On *Step 1* we automatically create a *relational schema* (RS) to express and preserve structure of the original data (*data representation level* on Fig. 1) and to provide a foundation for interoperability with DO (*pre-RDFS level* on Fig. 1). The data representation level contains notions common for all relational data sources (a header, rows, columns and cells). Concepts defined on the pre-RDFS level are shared between data sources with the same structure (table header). On *Step 2* we use RDF to express the actual content of the table according to the RS.

The second subproblem cannot be solved automatically due to the undefined semantics of the relational data. We leave it to the user to define the relationships between RS and DO (*Step 3*).

Our approach consists of three major steps. Additional actions (minor) are needed to exploit the RDF/S documents created during those steps to enable run-time interoperability (ontology-based querying).

Step 1: Build a relational schema to explicitly define the underlying relation. Since we cannot assume that every column represents an attribute and every

row represents an entity, we have decided to build relational schemata upon a notion of class.

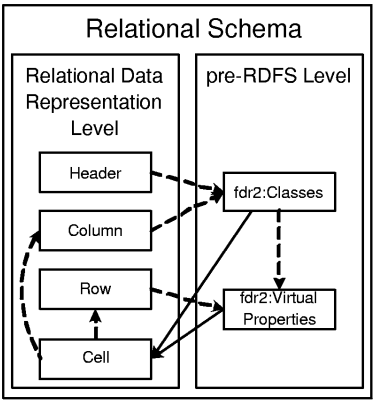

Fig. 1. The structure of the relational schema.

Every header cell represents a name of a class that is extensionally defined by the column cell values. An ordered set of all such classes $\boldsymbol{C}$ defines the underlying relationship $\boldsymbol{R}$ expressed within the table. RDFS models explicitly support binary relations only. Since any binary relation defined over a set $\boldsymbol{A}$ is a subset of $\boldsymbol{A}\boldsymbol{x}\boldsymbol{A}$, where $\boldsymbol{x}$ denotes a Cartesian product, we have decided to use $\boldsymbol{C}\boldsymbol{x}\boldsymbol{C}$ to obtain all binary *virtual relations* (*virtual properties*) defined over the classes presented in the relational schema.

After this step we have obtained the relational schema expressed in RDFS and containing a definition of all classes and all binary relationships between them. The resulting RS serves as a compact (intensional) representation of the data and can be directly connected to DO.

The structure of the relational schema is depicted on Fig. 1. Dashed arrows indicate transformation of simple concepts into more complex ones. For example, a collection of cells becomes a column which at the end is generalized into a class. Solid arrows explicitly indicate that classes and virtual properties enable access to the actual table data.

Step 2: Construct an RDF representation of the relational data. At this step we are dealing with the actual table content - the cell values. We consider every row as an instance of $\boldsymbol{R}$, where every cell is represented as an instance of a class corresponding to its column. In addition, we instantiate all the *virtual relations* defined on the previous step.

This step provides us with instances representing table data and once hidden but now explicit relationships between cells of a row. At this point we have the advantage of being able to use general-purpose RDF/S repositories and querying engines but we still cannot employ the vocabulary defined in DO to access the relational data.

Step 3: Ask the user to link RS with the domain ontology. The user links concepts (classes and properties) from the relational schema to corresponding concepts in the domain ontology by identifying `rdfs:subClassOf` and `rdfs:subPropertyOf` relationships between corresponding classes and properties. A set of all such links constitutes an RDMap (Relational-RDFS Map). The RDMap directly links the relational schema to ontological definitions.

Having obtained the RDF/S serializations of the relational data and the RDMap, an RDFS reasoner will be able to deduct all necessary entailments to

perform the actual linking of the relational schema and data with the concepts defined in DO. We will illustrate this with an example in the next subsection. The RDFS reasoner is required to merge the separate RDF/S documents and to generate the entailments.

To test the proposed technique and to provide basic support to the user we have developed ***FDR2#Kit***[1] – a web-based toolkit consisting of a few utilities: ***FDR2#Generator*** takes a tab-delimited text file with tabular data and automatically generates RDF/S documents for the relational schema and relational data;***FDR2#Mapper*** assists the user with linking the relational schema to DO; ***FDR2#Tester*** allows to run simple queries over the resulting combination of schema, data, RDMap and DO.

3 Discussion and Related Work

Over-generated *virtual relations* pose a serious performance problem. For example, a 10-column table will result in a RS with 90 *virtual relations* and quite significant portion of them may be redundant. Such a RS does not require a lot of resources to handle but a corresponding RDF-serialization of the table content will be polluted with irrelevant data. This problem can be handled by introducing a separate step between automatic generation of the RS and serialization of the table content. This stage is needed to enable the user to remove the redundant *virtual relations* from the original RS. Another option would be to swap steps 2 and 3 and to exploit a created RDMap to (semi)-automatically remove *virtual relations* not linked with DO. The modified RS will determine the final structure of the table content serialization preventing from polluting it with irrelevant data.

The relational schema facilitates analysis of the relational data on an abstract, intensional level. A possible practical applications of this is that an RDFS-based information system can keep track of known relational schemata and corresponding linking maps. This allows automating the whole process of handling complex input data. Since the RS is constructed automatically, it is quite likely that once created, the RDMap can be reused by many users who even do not know anything about the details of linking procedure and still able to take advantage of RDFS inference.

In [7] the authors describe a "naive" approach for mapping RDBMS schemata onto RDF (although we would rather call it RDBMS data mapping onto RDF). Our work takes it to the next level where we are focusing on linking relational and RDFS schemata. RDF serialization of actual data is quite straightforward and can be done in different ways according to application specific restrictions.

4 Conclusions

In this paper we have introduced ***FDR2*** – a technique that enables us to link relational and RDF/S data models. According to ***FDR2*** a relational schema

[1] http://www.cs.vu.nl/~maksym#tools

is automatically created to explicate the structure and internal relationships between elements of a relational collection of data. Explication of *virtual relations* allows the user to construct a relational schema specific RDMap by defining `rdfs:[subClass|Property]Of` relationships between concepts from the relational schema and a domain ontology. The actual relational data are automatically expressed in RDF according to the generated relational schema. Run-time integration is achieved by applying an RDFS reasoner to merge the above-mentioned components into a single RDFS model and to deduct necessary entailments. A resulting run-time model allows to access the relational data with queries termed according to the domain ontology. ***FDR2*** is purely RDF/S-based and does not require any additional software components except an RDFS reasoner.

References

1. W3C: Resource Description Framework (RDF). (http://www.w3.org/RDF/)
2. Omelayenko, B.: RDFT: A Mapping Meta-Ontology for Business Integration. In: Proceedings of the Workshop on Knowledge Transformation for the Semantic Web at the 15th European Conference on Artificial Intelligence (KTSW2002), Lyon, France (2002) 77–84
3. Bizer, C.: D2R MAP - Database to RDF Mapping Language and Processor. (http://www.wiwiss.fu-berlin.de/suhl/bizer/d2rmap/D2Rmap.htm)
4. aidministrator.nl: Sesame Project. (http://sesame.aidministrator.nl)
5. ICS-FORTH: The ICS-FORTH RDFSuite: High-level Scalable Tools for the Semantic Web. (http://139.91.183.30:9090/RDF/)
6. HP Labs Semantic Web Activity: Jena Semantic Web Toolkit. (http://www.hpl.hp.com/semweb/)
7. Beckett, D., Grant, J.: Semantic Web Scalability and Storage: Mapping Semantic Web Data with RDBMSes. SWAD-Europe deliverable, W3C (2003)