The Role of Design Rationale in the Ontology Matching Step during the Triplification of Relational Databases

Rita Berardi[1], Marcelo Schiessl[2], Matthias Thimm[3] , Marco A. Casanova[1]

[1] Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
Rio de Janeiro, RJ – Brazil CEP 22451-
{rberardi, casanova}@inf.puc-rio.br
[2] Faculdade de Ciência da Informação
Universidade de Brasilia
Brasília, DF - Brazil CEP 70910-900-
schiessl@unb.br
[3] Institute for Web Science and Technologies (WeST)
Universität Koblenz-Landau
Koblenz, Germany
thimm@uni-koblenz.de

Abstract. A common task when publishing relational databases as RDF datasets is to automatically define a vocabulary using the schema data, called *D*atabase *S*chema *O*ntology (DSO). To automatically reuse terms from existing vocabularies, Ontology Matching (OM) algorithms are used to generate recommendations for the DSO vocabulary. However, when a relational database is partially published due to privacy reasons, the DSO vocabulary loses contextual information and OM results in quite general recommendations, such as *FOAF* or *Dublin Core* vocabularies. To reduce the loss of context and to achieve recommendations better related to the DSO context, this paper proposes to use the design rationale captured during the publication of relational databases to RDF, represented as annotations. The case studies show that this annotation strategy helps find recommendations better related to the DSO context.

Keywords: triplification, private relational databases, design rationale

1 Introduction

One of the most popular strategies to publish structured data on the Web is to expose relational databases (RDB) in the Linked Data format, that is the RDB-to-RDF process - also called *triplification*. Broadly stated, there are two main approaches for mapping relational databases into RDF: (1) the *direct mapping* approach where the database schema is directly mapped to ontology elements and (2) the *customized mapping* approach where the schema of the RDF may differ significantly from the original database schema. Here, we consider the first approach where a vocabulary is directly defined from the relational database schema, resulting in a Database Schema Ontology (DSO). In order to promote interoperability of Linked Data it is

recommended that this vocabulary should reuse terms from existing vocabularies [4]. For that, Ontology Matching (OM) algorithms are useful to find recommendations to reuse terms from existing vocabularies. In this scenario, occasionally, only part of a relational database can be published as RDF, sometimes for privacy reasons or just because the rest of the data is not considered as interesting for other users. In the following, the part of the database that is not to be published is called *private*. In these cases, the DSO may lose important contextual information, leading to an OM recommendation of terms from more general vocabularies, such as FOAF or Dublin Core. For example, let us consider two OWL classes of a DSO with the terms "dso:publication" and "dso:author". It is not possible to identify which kind of publication the term "dso:publication" is referring to: it could refer to a research publication, a book, an article of a newspaper, or even to a song. The same happens with the term "dso:author". Without any additional information it is very hard to recommend terms from vocabularies of some specific context. This additional information could be their related classes or properties in the complete DSO, or their related entities or relationships in the complete RDB, or even typical instances in the RDB. In this case, OM algorithms are able to only recommend general terms like "dc:creator" (from Dublin Core vocabulary) or "foaf:person" (from FOAF vocabulary) for "dso:author". These recommendations are not wrong, but they do not provide the complete advantage of Linked Data since they represent a semantically weak description of these concepts. We argue that design rationale (DR) captured during the RDB-to-RDF mapping can improve the triplification process in many aspects, such as to analyze changes in the RDB context during the triplification, to verify loss of information and to improve the quality of vocabulary recommendations. In general, DR consists of decisions taken during a design process, the accepted and rejected options, and the criteria used. Concretely, we present in this paper a process, called StdTrip 2.0, that captures DR and uses it to reduce the loss of context when relational databases are partially published as RDF. We focus on the DR represented as a traceability of the original RDB form and how to use it to improve the quality of vocabulary recommendations. We assume that the private data must remain private, but the access to the schema information about the private part would not harm privacy policies. The DR is represented by systematically annotating the published data with the private schema information of the RDB. StdTrip 2.0 is an evolution of StdTrip+K [2, 1 12]. It improves StdTrip+K by adding the following features: (i) it enriches the DSO by annotating it with the private part of the relational database using *rdfs:comment*; (ii) it keeps a trace of all RDB-to-RDF transformations by storing the design rationale (DR) in a simpler directed graph and (iii) it includes a specific annotation strategy. Quite a few tools have been developed to the mechanical process of transforming relational data to RDF triples, such as Triplify [1], D2RQ [3], Virtuoso RDF view [9], RDBtoOnto [5] and RBA with support to R2RML [8][14], for both direct and customized mapping. However, they do not provide any design rationale capturing or any discussion for partial triplification of relational databases. The rest of this paper is organized as follows. Section 2 introduces the StdTrip 2.0 process. Section 3 presents case studies. Finally, Section 4 contains the conclusions and suggestions for future work.

2 The StdTrip 2.0 Process

StdTrip 2.0 includes a new step, for the cases where the relational database is partially published. The process receives as input a relational database (RDB), a set of mapping rules (MR), and known vocabularies of domain ontologies in the LOD cloud. It outputs an RDF vocabulary to represent the partial data from the RDB, alignments between DSO terms and terms of known vocabularies according to their context, and the final design rationale captured.

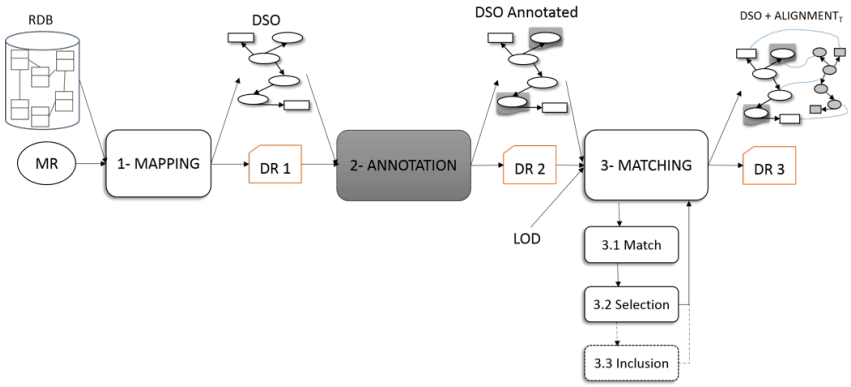

Fig. 1. The StdTrip 2.0 process

After each main step, a DR is recorded. We use the step number to refer to each DR; for instance, in step 1, we use DR1. Figure 2 depicts the publication database that we use as an example. Fig. 2 (b) shows the corresponding Entity Relationship model of the relational database. To exemplify a partially published relational database, we consider as private the grey part of Fig.2. (a) and (b)).

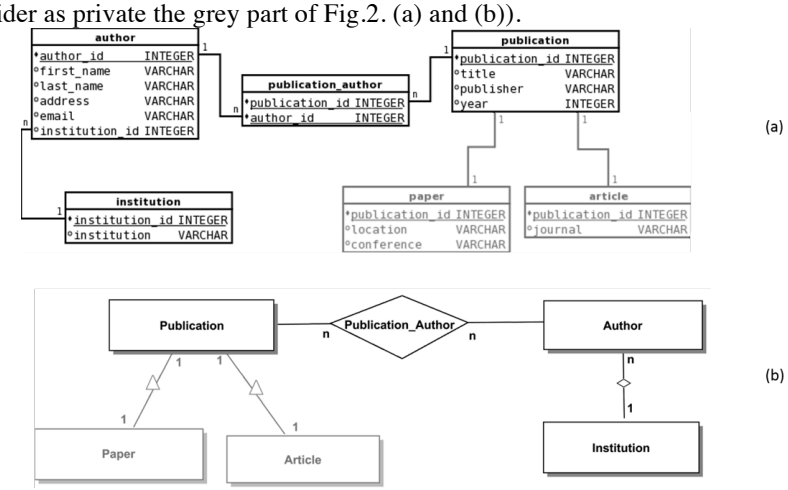

Fig. 2. The Author-Publication ER diagram [11]

2.1 Step 1 - Mapping

The Mapping step receives an Entity-Relationship (ER) extracted from an RDB schema (Fig.2) and a set of mapping rules (MR), defined by a domain expert. It

outputs a DSO and the corresponding design rationale (DR1). The DR1 records the original elements of the ER and how they were mapped to ontology elements. DR1 is created according to the definition of the trace proposed in [2]. The original ER is stored as a graph node and the trace regarding the mapping of each node is stored as node attributes. For example, in Fig. 3, the RE "Publication" is a node with attributes "**Element**", "**Map**" and "**Term**" that represent the trace of its mapping. These node attributes represent the "questions" of the DR1 model defined in [2], respectively: "Which **element** is it in the RDB?"; "How is it **mapped**?"; and "Which **term** is used to map it?". The answers to each of these questions (node attributes) obey a controlled vocabulary. For the question "**Element**", the possible answers are abbreviations of elements of the Entity-Relationship model of a relational database. The edges in all DR are named according to a controlled vocabulary to the relation elements in the original database. An answer to the question "**Map**" is any OWL element, such as "owl:Class" or "owl:ObjectProperty", when the element is mapped to the DSO, otherwise the answer is just "NOT". The question "**Term**" is answered only when the question "**Map**" was answered with something different from "NOT". The answer is naturally the term used for the mapping.

2.2 Step 2 – Annotation

The annotation process is a new step in StdTrip 2.0 and aims at using the *rdfs:comment* property to add information about the private database schema in the DSO. The input of this step is: (i) a DSO transformed from a corresponding ER model of an RDB and the DR1 containing the trace of this transformation; and (ii) the private schema data of the original relational database. The output of this step is the annotated DSO and the trace of the annotation in DR2. The DR2 is incrementally built from the preceding DR1. The benefits of using the DR graph instead of directly using the relational database are: (1) Since the DR graph is created for provenance purposes, it can be accessed without having to create a new graph based in the RDB to know what has to be annotated, (2) Since the DR graph is always created in the StdTrip 2.0, it can be consumed when needed, without having to re-execute the mapping step.

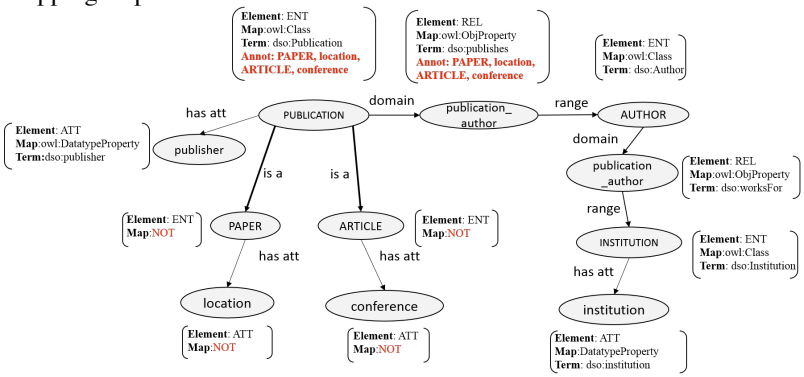

Fig. 3. Design Rationale 2 with annotations marked in red

The annotation is executed according to the *neighboring mapping*, that is: for each mapped element, look for its neighbors in the DR graph that were unmapped (Map:NOT) and, if they exist, the label of the unmapped node is annotated as a literal

of the *rdfs:comment* property. The search for unmapped neighbors is executed according to the annotation strategy and can be done at any depth in the DR graph. We adopted, as an initial strategy, the consideration of at most two levels based on empirical observation. More specifically, the empirical evidence is that we noticed that for automatic annotations, including more than two levels becomes superfluous, as the additional levels are more likely out of context. As an example of output, the annotations in the DR2 are ready to be added in the DSO ontology resulting in DSO_A in Fig. 3.

2.3 Step 3 – Matching

The general goal of this step is to find correspondences between the annotated DSO and standard well-known RDF vocabularies that really represent the context of the original database. This step comprises three sub-steps: (3.1) *Matcher execution*, where an ontology matching process is used to execute the matching; (3.2) *Selection of match candidates*, which creates, for each term of the DSO, a list of recommendations of terms from existing ontologies - here user interaction plays an essential role; and (3.3) *Inclusion*, where the domain expert can include other terms if he or she does not agree with the recommendations. Ideally, the user should know the database domain because he or she has to select the vocabulary elements that best represent each concept in the database. Similarly, to the previous DR models, the DR3 of this stage is incrementally stored.

3 Case studies

The case studies aim at validating whether the annotations generated by StdTrip 2.0 can help in finding recommendations better related to the context of relational databases when those are only partially published as RDF. For each relational database, two groups of recommendations are defined: one without using annotations and the other by using the suggested annotation step. The precision and recall measures are calculated for each group of recommendations that resulted from the OM. As we are evaluating the quality of the recommendations, sub-steps 3.2 and 3.3 (Selection and Inclusion) in StdTrip 2.0 are not executed in the case studies, where the domain expert decides which terms to reuse among the recommendations offered. We discuss in detail each point of the case studies in terms of the relational databases used and the Ontology Matching techniques adopted in the Matching step.

We executed StdTrip 2.0 until Step 3.1 (Match) to define three groups of recommendations for three relational databases that are different from each other in terms of context and number of tables: (1) *Publication* database, with 7 tables where 2 of them were considered as private [7], (2) *osCommerce*[2] database, with 48 tables where 36 of them were considered as private and (3) *phpBB*[3] database. The *OsCommerce* and *phpBB* databases were also used to evaluate Triplify that is one of the most popular tools to transform RDB in RDF [1]. In the Triplify evaluation [1], they affirm that only parts of these databases were interesting for publishing as RDF.

[2] http://www.oscommerce.com/

[3] http://www.phpbbdoctor.com/doc_tables.php

Besides that, the vocabulary to publish each one of them was completely defined by a domain expert reusing existing vocabularies, such as FOAF, SIOC and SKOS.

Regarding the Ontology Matching techniques used in the case studies, we propose a similarity measure of the terminological layer of OM. A terminological layer uses similarity measures to discover alignments by comparing labels, comments and definitions (annotations). Using OM algorithms from OAEI challenge [13], the results presented instability for matching a DSO that originally had no comments or definitions and comparing it with ontologies rich with definitions and comments. These measures, such as the Jaccard distance [6], "penalize" groups of strings with very different sizes and, consequently, result in low similarity values for them. We propose a *subset string measure-SbS* (1) that gives the similarity ratio minimizing the influence of the difference of sizes between the groups.

$$SbS(A,B) = \frac{|A \cup B||A \cap B|}{|A||B|} \in [0,1] \quad \textbf{(1)}$$

The Subset String (SbS) compares two lexical non-empty entries A, B and satisfies the following properties (i) SbS(A,B) = 1 iff $A \subseteq B \vee B \subseteq A$; (ii) SbS(A,B) = 0 iff $A \cap B = 0$ and (iii) SbS(A,B) $\uparrow \sim A \cap B \uparrow$, i.e., the SbS value increases as the intersection size also increases.

3.1 Result Analysis

We used the precision and recall measures to evaluate the quality of the results regarding our terminological SbS measure. For each relational database, these measures are calculated according to a Reference Alignment that has more contextual recommendations. The analysis is performed by comparing precision and recall measures for all databases, but when other interesting points appear besides these measures, they are also explored. Fig. 4 shows the results for the three databases.

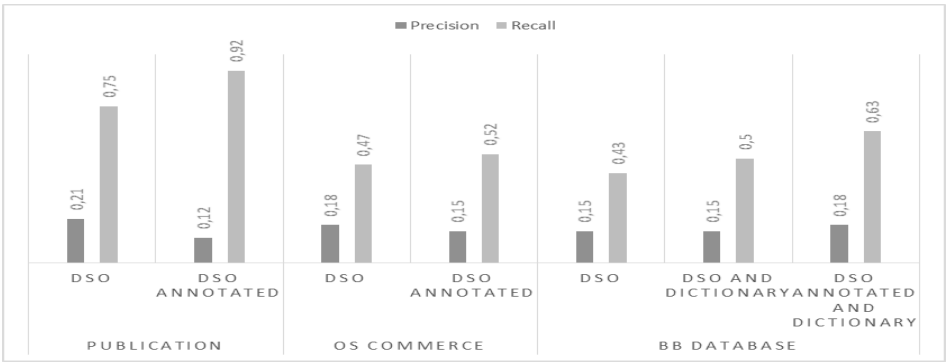

Fig. 4. Database results

The Reference Alignment for the *Publication* database was built according to our empirical knowledge about the publication context. The ontologies candidates for reusing were BIBO and FOAF. As we have previously discussed in Section 1, FOAF can be considered as a "general" context, then the ontology BIBO is more contextually related with the *Publication* database. Analyzing the results of *Publication* database, the recommendations for DSO ANNOTATED obtained lower

values of precision value, if compared with DSO without annotation. However, the recall of DSO ANNOTATED got higher values, when compared to DSO without annotations. This characteristic is common for approaches that use extra information [5]. The StdTrip 2.0 still returns, as a recommendation, general terms (like FOAF), but it additionally recommends more contextual terms (like BIBO). Although the recommendations are more related with the context, they do not point exactly the right term, thereby reducing precision. If we consider the complete StdTrip 2.0 process, recall values have more impact. Thus, if they are low, the domain expert has fewer options to find the most representative term for the context, or even no option if the exact term is not recommended. The precision would have more impact, if a domain expert participation is not taken into account in the following steps of the process. The Reference Alignment for the *osCommerce* database was built according to the Triplify indication about which vocabularies for each part were used during its evaluation [1]. Analyzing the precision and recall values, we had the same results as the *Publication* database. Note that, even with a much larger relational database (50 tables), the results are the same. Despite our positive interpretation for the recall values in both the *Publication* and the *osCommerce* databases, there is an important issue to explore. Even though the recommendations include a large group that are not precise (according to the precision values), the number of non-relevant terms not recommended is even greater. To analyze that issue, we used the *fall-out* measure [10] that gives the proportion of non-relevant terms recommended to reuse, out of all non-relevant terms available for reuse. The numbers of fall-out for *Publication* and *osCommerce* are respectively 0.21 and 0.01. It shows that, although the group of recommendations is larger than before, it still helps the domain expert avoid the need of analyzing huge amounts of terms that are available for reuse. For example, for a very small database like *Publication*, according to the fall-out measure, 79% of the available and the non-relevant terms were not recommended. To be more exact, in this case, receiving 217 non-precise recommendations is better than having to analyze 1003 available terms for reuse. Nevertheless, this number can be still improved by applying techniques of structural and semantic layers, which are subsequent layers in OM algorithms but they are not part of the discussion on this paper. The *phpBB* database has a different characteristic from the previous ones, since it has a quite complete data dictionary[4] that gives good comments about each table and their attributes. For that reason, we decided to also analyze if the data dictionary could replace our proposed annotations, since it somehow gives information about the context. To explore that point, we compared precision and recall among: (i) DSO without any annotations or dictionary data; (ii) DSO and DICTIONARY with dictionary data; and (iii) DSO ANNOTATED and DICTIONARY with dictionary data and annotations generated by StdTrip 2.0. If precision and recall would not differentiate between (ii) and (iii), it would mean that the annotations do not make a difference when the database has a dictionary. Analyzing the results, we may conclude that the data dictionary plays an important role, if compared to the DSO, but the annotated DSO, together the dictionary data, still has the best numbers for both precision and recall.

[4] Data dictionary here consists of textual definitions of tables and attributes

4 Summary and future work

In this paper, we introduced StdTrip 2.0 as a process that automatically defines a vocabulary from a relational database (Database Schema Ontology) and supports the generation of design rationale represented by annotations, recorded when they are partially published as RDF. The annotations are the private schema data information that provides more contextual information about the DSO. This contextualization helps in the terminological layer of Ontology Matching techniques for recommending existing vocabularies better related to the database context. The analysis of the case studies showed that these annotations, in different relational databases, provide important contextual information and help finding vocabularies to reuse that are better related to the database. As future work we are interested in achieving even better quality vocabulary recommendations by taking advantage of techniques in the further layers of OM such as syntactic and semantic layers. Thus, we can also achieve a lower number of recommendations besides more contextualized terms.

References

1. Auer S., Dietzold S., Lehmann J., Hellmann, S. and Aumueller, D.: Triplify: light-weight linked data publication from relational databases. In: WWW '09, pp. 621–630. ACM, New York, NY, USA (2009)
2. Berardi, R., Breitman, K., Casanova, M. A., Lopes, G. R., and de Medeiros, A. P.: StdTrip+K: Design Rationale in the RDB-to-RDF Process. In: Database and Expert Systems Applications, Prague, Czech Republic (2013)
3. Bizer, C. and Seaborne, A.: D2RQ-treating non-RDF databases as virtual RDF graphs. In: Proceedings of the 3rd International Semantic Web Conference ISWC 2004 (2004)
4. Breslin, J. G., Passant, A., and Decker, S.: The Social Semantic Web, Inc. (2009)
5. Cerbah, F.: Learning highly structured semantic repositories from relational databases. In: The Semantic Web: Research and Applications, pp. 777–781 (2008)
6. Cheatham, M. and Hitzler, P.: String Similarity Metrics for Ontology In: ISWC (2013)
7. Cullot, N., Ghawi, R. and Yétongnon, K.: DB2OWL: A Tool for Automatic Database-to-Ontology Mapping. In: SEBD 2007, 491--494 (2007)
8. Das, S., Sundara, S., and Cyganiak, R.: R2RML: RDB to RDF Mapping Language, W3C Working Draft, http://www.w3.org/TR/r2rml/ (2012)
9. Erling, O. and Mikhailov, I.: RDF support in the virtuoso DBMS. Networked Knowledge-Networked Media, pp. 7–24 (2009)
10. Euzenat, J., Shvaiko, P.: Ontology Matching, Springer (2007)
11. Salas, P., Viterbo, J., Breitman, K., Casanova, M.A.: StdTrip: Promoting the Reuse of Standard Vocabularies in Open Government Data, In: D. Wood (ed.) Linking Government Data, Springer Verlag , pp. 113–134 (2011)
12. Salas, P., Breitman, K., Viterbo J., Casanova, M.A.: Interoperability by design using the StdTrip tool: an a priori approach. In: I-SEMANTICS (2010)
13. Shvaiko, P. and Euzenat, J.: Ontology Matching: State of the Art and Future Challenges. In: IEEE TKDE, Los Alamitos, CA, USA (2013)
14. Vidal, V.M., Casanova, M.A., Neto, L.E., Monteiro, J.M. A.: Semi-Automatic Approach for Generating Customized R2RML Mappings. (accepted to the 29th ACM Symposium On Applied Computing, Gyeongju, Korea - March 24 - 28, 2014)