Pontifícia Universidade Católica do Rio de Janeiro

Rita Cristina Galarraga Berardi

Design Rationale in the Triplification of Relational Databases

Tese de Doutorado

Thesis presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática, PUC-Rio as partial fulfillment of the requirements for the degree of Doutor em Informática.

Advisor: Prof. Marco Antonio Casanova

Rio de Janeiro
February 2015

Pontifícia Universidade Católica do Rio de Janeiro

Rita Cristina Galarraga Berardi

Design Rationale in the Triplification of Relational Databases

Thesis presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática, PUC-Rio as partial fulfillment of the requirements for the degree of Doutor.

Rio de Janeiro, February 4th, 2015

Rita Cristina Galarraga Berardi

Graduated in Computer Science at the Federal University of Pelotas in 2006 and obtained her M.Sc. Degree in Computer Science from the Pontifical Catholic University of Rio de Grande do Sul in 2009.

Bibliographic data

Berardi, Rita Cristina Galarraga

Design rationale in the triplification of relational databases / Rita Cristina Galarraga Berardi ; advisor: Marco Antonio Casanova. – 2015.

100 f. : il. (color.) ; 30 cm

Tese (Doutorado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2015.

Inclui bibliografia

1. Informática – Teses. 2. BDR-para-RDF. 3. Triplificação. 4. Design rationale. 5. Mapeamento. 6. Matching. 7. Publicação parcial. 8. Mapeamento direto. 9. Mapeamento direto. 10. Mapeamento customizado. 11. R2RML. I. Casanova, Marco Antonio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CCD: 004

To my mother, my father and my love.

Acknowledgements

First of all, to God that gave me life and energy to persist since the beginning until the “end” of my studies way. To my family, specially my mother that offered me all the support and encouragement to always continue independently of the obstacles.

To my love Fillipp Hoefling Borba, who met me in the middle of this journey but accepted follow it to me as if it were his own, with love, courage and patience.

To my advisor Prof. Marco Antonio Casanova for the support and technical knowledge. To all professors of the Department of Informatics, PUC-Rio for their knowledge and help, especially to Karin Breitman who was my advisor in the beginning of the PhD at PUC-Rio. I thank Professor Simone Barbosa for the patience, attention and encouragement before going to the PhD Sandwich in Germany.

To CNPq and PUC-Rio, for the grants which made this work possible.

To all colleagues at PUC-Rio, especially my colleague and friend Percy Sallas who helped me with patience and with lots of laughs while we worked together in the room 522. Also, to my colleague and friend Marília Guterres Ferreira, for the several hours of conversation talking about tears and fears; she several times encouraged me to not give up. I would like to thank my friend Carlos Juliano for the your friendship of for the Sunday afternoons full of coffee and pão de queijo.

To Professor Steffen Staab, my advisor in the Sandwich period at West Institute at Universität Koblenz-Landau, Germany. Thanks for receiving me, for devoting part of your time to my research and also for the lunches under the sun of the German summer. Thanks Matthias Thimm for the very good meetings and all other colleagues at West Institute. To my colleague and new friend Marcelo Schiessl who became my (phycologist) counselor in Germany. Thanks for your time, your friendship, your family and for your ears.

Finally to all people that directly or indirectly participated of this journey in Brazil and in Germany.

Abstract

Berardi, Rita Cristina Galarraga; Casanova, Marco Antonio (Advisor). **Design rationale in the triplification of relational databases.** Rio de Janeiro, 2015. 100p. D.Sc. Thesis - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

One of the most popular strategies to publish structured data on the Web is to expose relational databases (RDB) in the RDF format. This process is called in RDB-to-RDF or triplification. Furthermore, the Linked Data principles offer useful guidelines for this process. Broadly stated, there are two main approaches to map relational databases into RDF: (1) the direct mapping approach, where the database schema is directly mapped to an RDF schema; and (2) the customized mapping approach, where the RDF schema may significantly differ from the original database schema. In both approaches, there are challenges related to the publication and to the consumption of the published data. This thesis proposes the capture of design rationale as a valuable source of information to minimize the challenges in RDB-to-RDF processes. Essentially, the capture of design rationale increases the awareness about the actions taken over the relational database to map it as an RDF dataset. The main contributions of this thesis are: (1) a design rationale (DR) model adequate to RDB-to-RDF processes, independently of the approach (direct or customized) followed; (2) the integration of a DR model in an RDB-to-RDF direct mapping process and in an RDB-to-RDF customized mapping process using the R2RML language; (3) the use of the DR captured to improve the recommendations for vocabularies to reuse.

Keywords

RDB-to-RDF; *triplification*; Design Rationale; mapping; matching; partial publication; directed mapping; customized mapping; R2RML

Resumo

Berardi, Rita Cristina Galarraga; Casanova, Marco Antonio (Orientador). ***Design rationale* na triplificação de bancos de dados relacionais.** Rio de Janeiro, 2015. 100p. Tese de Doutorado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Uma das estratégias mais populares para publicar dados estruturados na Web é expor bases de dados relacionais (BDR) em formato RDF. Esse processo é chamado BDR-para-RDF ou *triplificação*. Além disto, princípios de *Linked Data* oferecem vários guias para dar suporte a este processo. Existem duas principais abordagens para mapear bases de dados relacionais para RDF: (1) a abordagem de mapeamento direto, onde o esquema das bases de dados é diretamente mapeado para um esquema RDF, e (2) a abordagem de mapeamento customizado, onde o esquema RDF pode ser significativamente diferente do esquema original da base de dados relacional. Em ambas abordagens, existem vários desafios relacionados tanto com a publicação quanto com o uso de dados em RDF originados de bases de dados relacionais. Esta tese propõe a coleta de *design rationale* como uma valiosa fonte de informação para minimizar os desafios do processo de *triplificação*. Essencialmente, a coleta de *design rationale* melhora a consciência sobre as ações feitas no mapeamento da base de dados relacional para um conjunto de dados no formato RDF. As principais contribuições da tese são: (1) um modelo de *design rationale* (DR) adequado para o processo de BDR-para-RDF, independente da abordagem utilizada (direta ou customizada); (2) a integração de um modelo de DR para um processo que segue a abordagem direta de BDR-para-RDF e para um processo que segue a abordagem customizada usando a linguagem R2RML; (3) o uso do DR coletado para melhorar recomendações de reuso de vocabulários existentes através de algoritmos de *Ontology Matching*.

Palavras-chave

BDR-para-RDF; *triplificação*; Design Rationale; mapeamento; matching; publicação parcial; mapeamento direto; mapeamento customizado; R2RML

Table of Contents

1
Introduction

1.1
Motivation

Exposing relational databases (RDB) in the RDF format is one of the most popular strategies to publish structured data on the Web. The publication of relational databases as RDF is known as *RDB-to-RDF process*, also called *triplification* (McGuiness and Harmelen, 2004; Prud'hommeaux and Hausenblas, 2010). The process can be divided in two independent tasks: the *RDB-to-RDF mapping* task or, simply, *mapping* and the *RDB-to-RDF conversion* or, simply, *conversion*. The mapping is a fundamental step in the RDB-to-RDF process since it defines how to represent database schema concepts in terms of RDF classes and properties through the construction of a vocabulary. Furthermore, the Linked Data principles [1]offer useful guidelines for this process.

There are two main approaches to map relational databases into RDF: (1) the *direct mapping* approach, where the database schema is directly mapped to ontology elements (Sequeda et al., 2011); and (2) the *customized mapping* approach, where the RDF schema may significantly differ from the original database schema.

Independently of the approach (direct or customized), there are many challenges regarding publishing and consuming RDF data generated from relational databases. We highlight here some of these challenges:

1. How to select vocabularies to publish the relational data as RDF, following the Linked Data principles[2]?
2. How to maintain the RDB-to-RDF mapping that was created by another designer in the past?
3. How to use the R2RML language to express RDB-to-RDF mappings?

[12] http://www.w3.org/DesignIssues/LinkedData.html

4. How to re-execute just part of the RDB-to-RDF process to accommodate changes in the relational schema?
5. How to know whether the relational database lost information during the mapping process?
6. How to know whether the original relational database suffered changes during the mapping process that impact its quality?

The majority of these challenges are related to the ontology design to represent the relational database in RDF, required in both approaches (direct and customized). Designing an ontology involves a knowledge about the relational database, a decision about what data should be published, a decision about some manipulation in the data before publishing it, identifying possible terms from known domain ontologies to reuse, analyzing the possibilities and making a decision about the final vocabulary to represent the RDF data. In this ontology design the domain expert (who is publishing her or his dataset in RDF triples) empirically embeds all his or her knowledge about the context and his or her experiences around the original database through the choices during the vocabulary construction. In order to promote interoperability, it is recommended that the design of the final ontology should reuse other ontologies, at least in part (Breslin et al., 2009). For that, ontology matching (OM) algorithms are useful to find recommendations of known ontologies to reuse in the design of the final ontology. This is fundamental to reduce the effort involved in the consumption and integration of the published linked datasets (Cordeiro et al., 2011).

The final ontology and the published RDF data represent only the result of a RDB-to-RDF process for a specific relational database. However, it does not represent the knowledge that supported this result, such as the reasons behind the decision about reusing a term instead of reusing another term from another known domain ontology. In addition, it does not represent the original context of the relational database. In other words, both these aspects do not represent the final result of a RDB-to-RDF process.

In general, the decisions taken during a design process, the accepted and rejected options, and the criteria used are called *design rationale* (DR) (Lee, 1997). A complete definition for DR is proposed by (Lee, 1997): *design rationale* includes not only the reasons behind a design decision but also the justification for it, the other alternatives considered, the tradeoffs evaluated, and the argumentation that led to the decision. Thus, the collected rationale comprises the positive decisions, the choices that were abandoned as well as their justifications.

Therefore, we propose in this thesis to incorporate a *design rationale* (DR) model into RDB-to-RDF processes that use both direct and customized approaches. Briefly, the design rationale of an RDB-to-RDF process consists of the decisions taken during the design process, the accepted and rejected options, and the criteria used. We argue that the DR captured during the RDB-to-RDF process may be used to address the aforementioned challenges. In general, such DR allows to:

- Analyze which kind of transformations the data suffered during the RDB-to-RDF process;
- Verify loss of information;
- Improve the quality of the vocabulary recommendations.

Specifically, the DR of an RDB-to-RDF process captures:

- The original schema of the relational database;
- Which entities of the relational database are mapped to ontology elements;
- How each entity of the relational database is mapped as an ontology element;
- What is the chosen ontology;
- The accepted and rejected options recommended by an ontology matching technique that the user agreed (or did not agree).

1.2 Contributions

This thesis contributes to a conscious publication and consumption of RDF data generated from relational data. By capturing the design rationale, it helps publishers to document the design process and final users to consume the published data by giving them evidences to answer questions about the mapping process.

In more detail, the first contribution of this thesis is a DR model for RDB-to-RDF processes, called Kuaba+W, which simplifies a DR model, called Kuaba (Medeiros and Schwabe, 2008). Briefly, Kuaba is an argumentation-based approach for representing DR using a defined vocabulary that is integrated with the formal semantics of the artifacts, provided by the metamodels of the design. In this thesis, we simplify the Kuaba vocabulary to cover the RDB-to-RDF context, by eliminating elements not present in the RDB-to-RDF context. The model Kuaba+W is briefly introduced in Chapter 2 and then presented in detail together with the RDB-to-RDF processes in which it was coupled.

The second contribution is the incorporation of Kuaba+W into StdTrip (Salas et al., 2010a; 2010b), an existing RDB-to-RDF direct mapping approach. The modified process, called StdTrip+K, is a direct mapping process with 4 main steps. For each step, the DR model records all information about the actions and the decisions. As an example, we show how to record the DR model by using an Author-Publication relational database in Chapter 3.

The third contribution is an extension of StdTrip+K, called StdTrip 2.0, which addresses the problem of defining a vocabulary for relational database that is partially published in RDF format. 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 these cases, the vocabulary may lose important contextual information, leading to an ontology matching term recommendation from more general vocabularies, such as FOAF or Dublin Core. We present a strategy that shows that the DR collected during the RDB-to-RDF process can be used to provide contextual information that leads to better recommendations from an ontology matching process in the sense that the recommendations are more related to the context of the complete database.

StdTrip 2.0 was developed separately from StdTrip+K to generate a complete new version of the process. StdTrip 2.0 keeps the 4 steps of the StdTrip+K and adds a new one, the Annotation step for the cases where the relational database is partially published.

Although it was not a priority of this thesis, the development of StdTrip 2.0 also lead to an additional contribution regarding the measures used in the context of ontology matching algorithms. A common layer in the OM algorithms is the terminological layer, where a bag of words is compared with groups of words with different sizes. We developed a measure for this layer that tries to prioritize the similarity value over the difference of sizes between the groups that are compared.

The final contribution of this thesis is the incorporation of the DR model into RBA approach (R2RML by assertion), a customized RDB-to-RDF approach that uses R2RML, the standard language proposed by W3C RDB2RDF Working group. The resulting approach, called R^2BA, also includes OM algorithms to generate recommendations for reusing vocabularies and it is detailed in Chapter 5.

Finally, we remark that Chapters 3, 4 and 5 reflect papers published respectively in the 2013 and 2014 International Conference on Database and Expert Systems Applications (DEXA) (Berardi et al., 2013; Berardi et al., 2014). The results reported in Chapter 5 were published in ICEIS 2015 (Berardi et. al., 2015).

1.3 Thesis Outline

The remainder of this thesis is structured as follows. Chapter 2 presents a brief description of Kuaba and a brief introduction of Kuaba+W. Chapter 3 explores the StdTrip+K process with a DR model incorporated. This chapter further details the Kuaba+W model. Chapter 4 introduces an evolution of StdTrip+K with a new strategy to use the DR captured as an annotation step in a direct mapping approach, generating the StdTrip 2.0 process. Chapter 5 describes a customized mapping approach with the DR model and the annotation strategy incorporated. Chapter 6 concludes with a summary of the thesis and outlines future work.

Finally, we observe that the mechanical process of transforming relational data to RDF triples is a well-known task, backed up by a wide range of tools. Related work is discussed along with the chapters, according to each research problem addressed in the chapter.

2 Background

This chapter briefly reviews the basic concepts of design rationale. Then, it outlines Kuaba (Medeiros and Schwabe, 2008), a generic design rationale model. The core of the chapter introduces a simplification, Kuaba+W, which is based on the RDB-to-RDF context. Chapter 3 explores the application of Kuaba+W in a direct RDB-to-RDF process and Chapter 5 in a customized RDB-to-RDF process.

2.1 A Brief Review of Design Rationale

Design rationale (DR) is an explanation of why an artifact, or some part of an artifact, is designed the way it is (Lee, 1997). The author still says that design rationale includes not only the reasons behind a design decision but also the justification for it, the other alternatives considered, the tradeoffs evaluated, and the argumentation that led to the decision. Researches in the DR area seek to provide models and tools that allow explicitly recording these reasons, in order to support its use in the design of new artifacts and in the reuse of already designed artifacts.

An important definition is necessary to better understand the main objectives around the design rationale concepts: *what does design means*? Currently, there is no consensus in the literature about the definition of *design*. Different definitions can be found in Herbert (Herbert, 1981), Donald (Donald, 1983), Vinod and Peter (Vinod et al., 1989), Vladimir and Ernst (Vladimir et al., 1996) and Terry (Terry, 1996). In the scope of this thesis, design is the activity carried out to create an ontology from a relational database so that it can be published in the Linked Data format.

DR has a potential value for supporting design reuse because all the experience acquired during a design can be transmitted and augmented by the use of recorded DRs in new designs. In the context of linked data this is a very important characteristic once reuse is a key feature of this domain. When a dataset is published in the web, it is expected that it is available to be reused, but what may happen is a confusing interpretation about what is exactly the domain of the published dataset, and it can cause mistakes for the linking activity through the property owl:sameAs [3], for instance. In addition the reuse can happen in the context of the generation of an ontology derived from a database. All experience negative and positive can be consulted during the process of creating new ontologies. Nevertheless, despite much research, DR has not been very much used for Linked Data domain. One of the reasons may be the time consumption and the cost generally required for the capture and representation of DR. This is due the lack of a representation approach that enables the development of an integrated tool that supports the capture, representation and use of DR as part of the process of publishing databases in RDF triples. The main threshold for the adoption of DR is its capturing, because if it is not captured during the process of interest, lots of important rationale information is lost and, in the other hand, asking for the designers to declare them rationale after the design already, it is very laborious activity and occurs loss of information in the same way.

There are several argumentation-based proposals for representing DR, such as IBIS (Kunz and Rittel, 1970), DRL (Lee and Lai, 1991), QOC (Maclean et al., 1991), and PHI (McCall, 1991), however most of them generate incomplete or informal representations, not enabling the effective and efficient access of DR for reuse. In Linked Data context the main emphasis in the sense of representing extra information in the mapping has been done towards the development of provenance approaches[4] in W3C initiatives. Although these initiatives are very relevant for Linked Data, there is a deficiency of history recording that are not covered by provenance approach that is in terms of options, arguments, decisions and justifications, original format of relational data, what is not the scope of current provenance approaches[5]. In this sense, we present in thesis Kuaba+W, an approach for representing DR in the publication of relational database data in RDF datasets. Kuaba+W simplifies an existing generic DR model called Kuaba that we briefly introduce in the next Section.

[3] http://sameas.org/

[4] http://www.w3.org/2011/prov/wiki/Main_Page

[5] http://openprovenance.org/model/opmo

2.2 Kuaba

The vocabulary of Kuaba is composed of a set of elements (classes, properties, restrictions) to express the design rationale domain. Kuaba extends the notation of IBIS (Kunz and Rittel, 1970) by adding an explicit representation for the decisions and justifications taken during a design process and by representing the relation between the argumentation elements and the artifacts produced. In addition, Kuaba adds the information about the history of the artifact designed in terms of who made the decisions, when they were taken and so on.

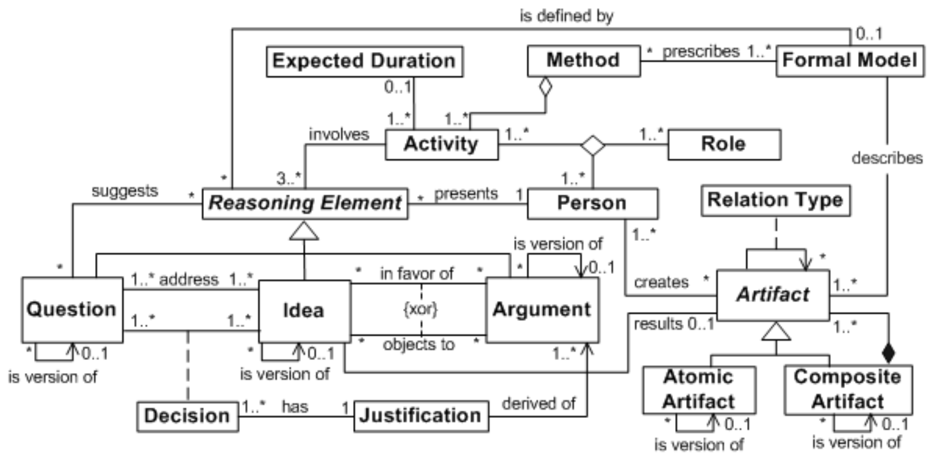


Figure 2.1 – Kuaba vocabulary (Medeiros and Schwabe, 2008).

The elements present in the Kuaba vocabulary, depicted in Figure 2.1, represent the reasoning elements used during a design process, the decisions taken during a design process, information about the artifacts that are result of the design process, and information about specifically each activity done during a design process.

Briefly, during a design process, several *Reasoning Elements* are used to design an artifact. Figure 2.1 illustrates the properties of the Reasoning Elements that can be involved in an activity of design. A *Question* represents a design problem that has to be resolved. An *Idea* represents a possible design solution, or part of a design solution for the problem addressed by the element *Question*. An *Argument* represents a reason against or in favor of an *Idea* as a solution for the problem addressed by the element *Question.* A *Decision* is related to an *Idea* about a *Question* and it is taken based on the *Arguments* presented in favor of or against the *Ideas*. The *Justification* can be inserted by a domain expert to justify the reason behind the decision.

In order to improve the evolution of design artifacts, Kuaba integrates the design rationale with the description of the artifact. In Figure 2.1, the element *Artifact* corresponds to a final design solution. Kuaba represents the artifact as an atomic artifact or a composite artifact, composed by other artifacts. This *Artifact* must be related to an *Idea* that was accepted.

Kuaba allows recording the *Method* used to design an artifact, the *Activities* executed, the *Persons* involved in the activity, the Role they play during the execution of the activity and the *Duration* of the activity. A Person can play different roles and build an artifact according to a *Formal Model* that describes the artifact.

2.3 Kuaba+W – A Design Rationale Vocabulary for RDB-to-RDF Processes

Kuaba+W simplifies the Kuaba approach in the sense that it eliminates elements which are not necessary in the RDB-to-RDF context, such as: (i) the recording of specific *activities* to create an artifact since, in the RDB-to-RDF process, it is already determined that we are tracing the mapping and matching activities; (ii) the *duration* of these activities; (iii) the *method* used to execute the activity, since in the RDB-to-RDF process there is no rigid method, but instead it is expressed through the arguments and the justifications for the decisions; (iv) the *role* played by who is performing the design, since it is already determined that the database domain expert will perform all activities; and (v) the *artifacts* generated by the design and its decompositions since, in the RDB-to-RDF process, there is only one artifact generated (the final ontology) and, hence, it is not necessary to record it. We did not use Kuaba directly because the non-instantiation of some elements of Kuaba would generate inconsistencies, since some elements are not present in the RDB-to-RDF context as discussed before.

Furthermore, the Kuaba+W adds the *Description* element, which is related to a *Justification* and carries information regarding the reasons for the domain expert to accept or reject an idea. During the modeling of the rationale, the same question can be repeated several times for different reasoning elements; hence, the

Description element permits to uniquely identify each one to keep the traceability of what question is being answered for an idea.

Also, the Kuaba+W simplification relates a description to a *metamodel*, since there is more than one metamodel involved in the RDB-to-RDF process. Indeed, a metamodel records which formal artifact was involved in each step of design process, such as the entity-relation (ER) and the RDF metamodels. A description is also related to a reasoning element to give a complete definition regarding the reasoning element, which can be a clarification about an idea or a more complete definition of an argument.

Kuaba+W is prepared to record the design rationale of direct and customized mappings of relational databases. An important characteristic of the customized approach is that the customization depends on the audience of the published dataset; therefore, an important information to record is to which audience the dataset is targeted.

Figure 2.2 shows the main elements of the Kuaba+W ontology, using a UML-like graphical notation to help visualization and the correspondent XML is depicted in Table 2.2.

The rest of this section describes the classes and properties of the Kuaba+W ontology (we named the vocabulary as Rationale Vocabulary – rv), followed by a brief discussion.

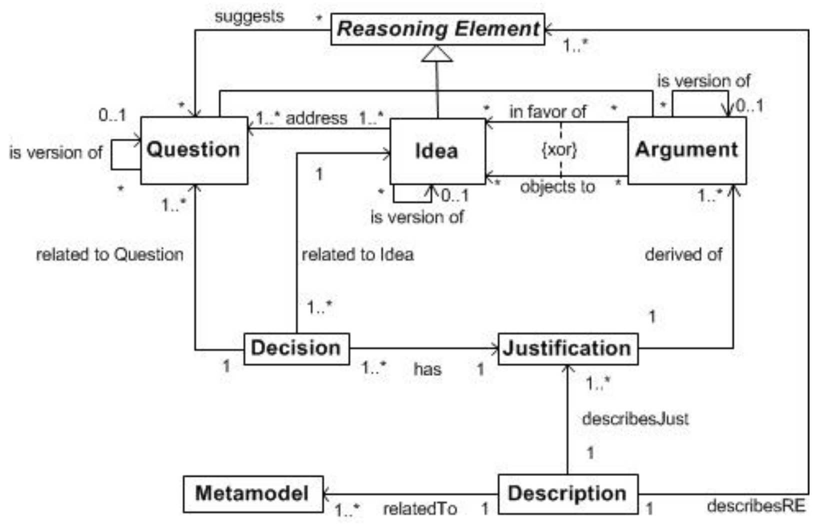


Figure 2.2 – The Kuaba+W ontology elements.

A *Reasoning element* (Class `rv:ReasoningElement`) represents the design issue that the ontology designer should deal with (question, ideas and arguments).

A Reasoning element suggests (Property `rv:suggests`) a Question (Class `rv:Question`), Sub class of `rv:ReasoningElement`) that represents a mapping or matching issue to be solved regarding each element of the RDB.

An *Idea* (Class `rv:Idea`, Sub class of `rv:ReasoningElement`) addresses (Property `rv:address`) the answer for a `rv:Question` element and represents a potential solution for the mapping or matching issue presented by the reasoning element `rv:Question`.

The *Argument* (Class `rv:Argument`, Sub class of `rv:ReasoningElement`) represents the criteria used to present an `rv:Idea` for a `rv:Question`. It can be in favor the acceptance of the `rv:Idea` (Property `rv:inFavorOf`) or objecting to it (Property `rv:objectsTo`).

A *Decision* (Class `rv:Decision`) represents the acceptance or the rejection of an idea as a solution to a question. A `rv:Decision` has to be related to a question (Property `rv:relatedToquestion`) and to an Idea (Property `rv:relatedToidea`) because it is a relation between a `rv:Question` and `rv:Idea`.

A *Justification* (Class `rv:Justification`) indicates the justification for each `rv:Decision` that explains why an `rv:Idea` was accepted or rejected as a solution for a particular `rv:Question` (Property `rv:hasJustification`). Every `rv:Justification` is derived from an `rv:Argument` (Property `rv:derivedOf`);

A *Description* (Class `rv:Description`) contains details about any reasoning element and justification, depending on the step of the mapping process. It indicates to which `rv:ReasoningElement` is the `rv:Description` about (Property `rv:describesRE`) as well as indicates to which `rv:ReasoningElement` is the `rv:Justification` about (Property `rv:describesJus`);.

A *Metamodel* (Class `rv:Metamodel`) indicates which metamodel is accessed in the mapping process to automatically build the RDF rationale. It indicates from what `rv:Metamodel` the `rv:Description` originates (Property `rv:relatedTo`).

Table 2.1 – Summary of the Kuaba+W vocabulary.

- **Reasoning element**
  Class: `rv:ReasoningElement` **Sub class of:** `owl:Thing`

  The reasoning element represents the design issues that the ontology designer should deal with (questions, ideas and arguments).

  Property: `rv:suggests` (`rv:ReasoningElement` -> `rv:Question`)

  Indicates that question originated from the reasoning element.

  Property:

  `rv:isVersionOf` (`rv:ReasoningElement` -> `rv:ReasoningElement`)

  Indicates that a reasoning element is version of an existing reasoning element.

- **Question**
  Class: `rv:Question` **Sub class of:** `rv:ReasoningElement`

  A question represents a mapping or matching issue to be solved regarding each element of the RDB.

  Property: `rv:isVersionOf` (`rv:Question` -> `rv:Question`)

  Indicates that a reasoning element is version of an existing reasoning element.

- **Idea**
  Class: `rv:Idea` **Sub class of:** `rv:ReasoningElement`

  An idea represents a potential solution for the mapping or matching issue presented by the element `rv:Question`.

  Property: `rv:address` (`rv:Idea` -> `rv:Question`)

  Indicates the question addressed by an idea.

  Property: `rv:isVersionOf` (`rv:Idea` -> `rv:Idea`)

  Indicates that a reasoning element is version of an existing reasoning element.

- **Argument**
  Class: `rv:Argument` **Sub class of:** `rv:ReasoningElement`

  An argument represents the criteria used to present an `rv:Idea` for a `rv:Question` .

  Property: `rv:inFavorOf` (`rv:Argument` -> `rv:Idea`)

Indicates that the Argument is in favor of the acceptance of the Idea.

Property: `rv:objectsTo` (`rv:Argument` -> `rv:Idea`)

Indicates that the Argument is contrary to the acceptance of the idea.

Property: `rv:isVersionOf` (`rv:Argument` -> `rv:Argument`)

Indicates that a reasoning element is version of an existing reasoning element.

- **Decision**
  Class: `rv:Decision`

  The acceptance or the rejection of an idea as a solution to a question is recorded by the Decision class. It is a relation between a Question and an Idea.

  Property: `rv:relatedToquestion` (`rv:Decision` -> `rv:Question`)

  Indicates to Question the Decision is related to.

  Property: `rv:relatedToidea` (`rv:Decision` -> `rv:Idea`)

  Indicates to Idea the Decision is related to.

- **Justification**
  Class: `rv:Justification`

  Each decision must have a justification that explains why an idea was accepted or rejected as a solution for a particular question.

  Property: `rv:hasJustification` (`rv:Decision` -> `rv:Justification`)

  Indicates to which Decision the Justification is regarding to.

  Property: `rv:derivedOf` (`rv:Justification` -> `rv:Argument`)

  Indicates from wich Argument has derived the Justification

- **Description**
  Class: `rv:Description`

  A description contains details about any reasoning element and justification, depending on the step of the mapping process.

  Property: `rv:describesRE` (`rv:Description` -> `rv:ReasoningElement`)

  Indicates to which Reasoning Element is the Description about.

  Property: `rv:describesJus` (`rv:Description` -> `rv:Justification`)

  Indicates to which Reasoning Element is the Justification about.

- **Metamodel**
  Class: `rv:Metamodel`

  Depending on the step of the mapping process a corresponding metamodel is accessed to automatically build the rationale RDF.

  Property: `rv:relatedTo` (`rv:Description` -> `rv:Metamodel`)

  Indicates from what Metamodel is originated the Description.

2.4
The illustration of Kuaba+W

Table 2.2 shows the RDF rationale vocabulary derived from Kuaba+W ontology in Figure 2.2 . In the block of lines 2 to 56 are the definitions of Classes that will compose the final rationale RDF; in lines from 58 to 102 are the properties.

Table 2.2 – Rationale Vocabulary.

```
1: Prefix: rv:http://purl.org/rationalevocab#.
2: ##Classes##
3:rv:ReasoningElement rdf:type owl:Class;
4:                    rdfs:label “ReasoningElement”@en;
5:                    rdfs:subClassOf owl:Thing;
6:   rdfs:comment “The reasoning element represents the design problems
7:                 that the designer should deal with”@en.
8: rv:Idea rdf:type owl:Class;
9:         rdfs:label “Idea”@en;
10:        rdfs:subClassOf rv:ReasoningElement;
11:        rdfs:comment “An idea represents a potential solution, or
part of a design solution for
12:           the problem presented by the element Question.“@en;
13:        rdfs:subClassOf [ rdf:type owl:Restriction;
14:                                owl:onProperty rv:address;
15:                                owl:onClass rv:Question;
16:                    owl:minCardinality ”^^xsd:nonNegativeInteger;
17:                    owl:maxCardinality “1”^^xsd:nonNegativeInteger
18:                                          ].
19: rv:Question rdf:type owl:Class;
20:             rdfs:label “Question”@en;
21:             rdfs:subClassOf rv:ReasoningElement;
22:             rdfs:comment “A question represents a design problem to
be solved. “@en.
23: rv:Argument rdf:type owl:Class;
24:             rdfs:label “Argument”@en;
25:             rdfs:subClassOf rv:ReasoningElement;
26:             rdfs:comment “An argument represents a reason in favor
```

```
of or objects to the
  adoption of an idea as solution to the respective Question.“@en.
rv:Decision rdf:type owl:Class;
            rdfs:label “Decision”@en;
            rdfs:comment “The acceptance or the rejection of an idea
as a solution to a
question is recorded by the Decision class. It is a relation between
a Question            and an
Idea.“@en;
rdfs:subClassOf [ rdf:type owl:Restriction;
            owl:onProperty rv:justification;
            owl:onClass rv:Justification;
            owl:minCardinality “1”^^xsd:nonNegativeInteger;
            owl:maxCardinality “1”^^xsd:nonNegativeInteger;
                                        ].
rv:Justification rdf:type owl:Class;
            rdfs:label “Justification”@en;
            rdfs:comment “Each decision must have have a
justification that explains why an idea was accepted or rejected as
a solution for a particular question.“@en;
            rdfs:subClassOf [ rdf:type owl:Restriction;
            owl:onProperty rv:derivedOf;
            owl:onClass rv:Argument;
           owl:minCardinality “1”^^xsd:nonNegativeInteger;
           owl:maxCardinality “1”^^xsd:nonNegativeInteger;
                                        ].
rv:Description rdf:type owl:Class;
      rdfs:label “Description”@en;
      rdfs:comment “A description contains details about any
reasoning element
  and justification depending the step of the mapping process.“@en.
rv:Metamodel rdf:type owl:Class;
            rdfs:label “Metamodel”@en;
            rdfs:comment “Depending the step of the mapping process
a corresponding metamodel is accessed to automatically build the
rationale rdf.“@en.
##Properties##
rv:suggests rdf:type owl:ObjectProperty;
                    rdfs:label “suggests”@en;
                    rdfs:domain rv:ReasoningElement;
                    rdfs:range rv:Question.
rv:isVersionOf rdf:type owl:ObjectProperty;
                        rdfs:label “isVersionOf”@en;
                        rdfs:domain rv:ReasioningElement;
                        rdfs:range rv:ReasoningElement.
rv:inFavorOf rdf:type owl:ObjectProperty;
                     rdfs:label “inFavorOf”@en;
                     rdfs:domain rv:Argument;
                     rdfs:range rv:Idea;
                     owl:inverseOf rv:objectsTo.
rv:objectsTo rdf:type owl:ObjectProperty;
```

```
                     rdfs:label "objectsTo"@en;
                     rdfs:domain rv:Argument;
                     rdfs:range rv:Idea.
rv:address rdf:type owl:ObjectProperty;
                  rdfs:label "address"@en;
                  rdfs:domain rv:Idea;
                  rdfs:range rv:Question.
rv:hasJustification rdf:type owl:ObjectProperty;
                              rdfs:label "hasJustification"@en;
                              rdfs:domain rv:Decision;
                              rdfs:range rv:Justification.
rv:derivedOf rdf:type owl:ObjectProperty;
                     rdfs:label "derivedOf"@en;
                     rdfs:domain rv:Justification;
                     rdfs:range rv:Argument.
rv:relatedToquestion rdf:type owl:ObjectProperty;
                                  rdfs:label "relatedToquestion"@en;
                                  rdfs:domain rv:Decision;
                                  rdfs:range rv:Question.
rv:relatedToidea rdf:type owl:ObjectProperty;
                           rdfs:label "relatedToidea"@en;
                           rdfs:domain rv:Decision;
                           rdfs:range rv:Idea.
rv:describesRE rdf:type owl:ObjectProperty;
                         rdfs:label "describes"@en;
                         rdfs:domain rv:Description;
                         rdfs:range rv:ReasoningElement.
rv:describesJus rdf:type owl:ObjectProperty;
                        rdfs:label "describes"@en;
                        rdfs:domain rv:Description;
                        rdfs:range rv:Justification.
```

This "rv" vocabulary is used during the mapping process to create the rationale in RDF triples as explained in the next session. Recording this information facilitates the processing and the interpretation of the represented rationale, providing part of the context in which the ontology was created.

3 Design Rationale for an RDB-to-RDF Direct Mapping Process

3.1 Introduction

The term Linked Data refers to a set of best practices for publishing and connecting structured data on the Web (Bizer et al., 2009).

One of the major challenges of publishing Linked Data is to investigate the value of information based on the trustworthiness of its sources, the time of validity, the certainty, or the vagueness asserted to specified or derived facts (Dividino et al., 2009). This challenge is associated with the lack of analytical information about the published Linked Data, i.e. information that answers questions such as:

- Did the original relational database suffer *changes* when published as Linked Data that could impact its quality?
- Is the *translation* from the original relational database to Linked Data *correct*?
- Is the *chosen ontology* the most appropriate to represent the original relational database?
- Did the original relational database *lose* some relevant *information* when it was published as Linked Data?

In the current RDB-to-RDF processes, there is no mechanism to track and record this information. Hence, our motivation is anchored in the determination of what details of the RDB-to-RDF process should be made available and how to do this so that the details can help in the use and reuse of the respective Linked Data. These details of the RDB-to-RDF process should answer the aforementioned questions, which refer to the decisions related to changes, correctness, choices and the information lost during the RDB-to-RDF process.

By analogy, the design rationale behind the *triplification* of a relational database is called *triplification rationale*. Besides helping the reuse of datasets, the *triplification rationale* has a potential value for supporting design of new ontologies because all the experience acquired during a design can be transmitted and augmented by the reuse of previous DRs in new designs. The designer would benefit from the experience – negative and positive – obtained during the definition of previous ontologies. In the context of Linked Data, this is a very important characteristic since reuse is a key feature in this domain, inasmuch as when a triple set is published in the Web, its use by other applications is expected.

Although there are several RDB-to-RDF engines, we are unaware of any previous work that applies DR in the Linked Data domain, i.e., that captures the *triplification rationale*. To partly mitigate this gap, we present the StdTrip+K process, which integrates a design rationale approach with an RDB-to-RDF strategy – specifically in the mapping and matching steps of the RDB-to-RDF process. StdTrip+K is based on an RDB-to-RDF approach, called StdTrip (Salas et al, 2010a), which addresses ontology reuse by considering the designer participation. For the DR representation, StdTrip+K follows Kuaba+W, introduced in Chapter 2.

The overall goal of StdTrip+K is to allow the computational collection and representation of DR about the mapping and matching step.

The mapping step of StdTrip+K addresses how to represent database schema concepts in terms of RDF classes and properties. The details intrinsically involved in the mapping activity should reflect all aspects related to how the concepts of the underlying conceptual schema are mapped to the RDF terms. Each choice involves options, criteria of choices, arguments, decisions, acceptances, and rejections taken by the designer, which explain why and how each element of the ontology was mapped. Furthermore, these detailed information can explicit some problems in the mapping process. For instance, if an *entity element* of an Entity Relationship diagram is mapped to a *property element* in RDF, the *attribute elements* of this entity may not be represented due to the lack of the domain representation, since the domain is represented as a property.

The matching step of StdTrip+K involves domain expert decisions regarding the construction of the vocabulary. A strategy, such as StdTrip, focused in reuse, is extremely important, because the more one reuses well known standards, the easier it will be to interlink the result with other existing dataset (Breslin et al., 2009). This is fundamental to reduce the effort involved in the consumption and integration of the published linked datasets (Breslin et al., 2009). The details inherent in the matching step should reflect aspects related to the choice of each term of the vocabulary that will be used to publish the database. The decisions of the designer involved in this activity will need to consider the database domain and context. For instance, considering a domain of an university publication database where the entity "Authors" has the attribute "name", the more adequate representation is *dc:creator* instead of *foaf:Person,* since *dc:creator* is more representative for the domain. Otherwise, if an entity "Students" has the same attribute "name", *dc:creator* is not the best choice although both entities are in the same domain "University".

The remainder of this chapter is organized as follows. Sections 3.2 summarizes the StdTrip+K process. Section 3.3 introduces a running example that shows how to represent the DR using the Kuaba+W vocabulary. Section 3.4 details the architecture of the StdTrip+K Tool and its output files are shown in Section 3.5. Section 3.6 discusses related work. Finally, Section 3.5 presents the conclusions of this chapter.

3.2 The StdTrip+K Process

The StdTrip+K process, overviewed in Figure 3.1, is anchored in the principle of ensuring interoperability through the use of standards in schema design and through the recording of the design rationale. The process receives the RDB relational database (RDB), the metamodels and Kuaba+W, the design rationale vocabulary. At each step, the respective design rationale (DR) is traced and incrementally recorded using Kuaba+W vocabulary. In the end, the process results in the RDB-to-RDF mapping File, the OWL ontology and the final DR, here represented by DR 4. The four steps (Mapping, Matching, Selection and Inclusion) of the StdTrip+K process are described in Section 3.3, using a motivation example.

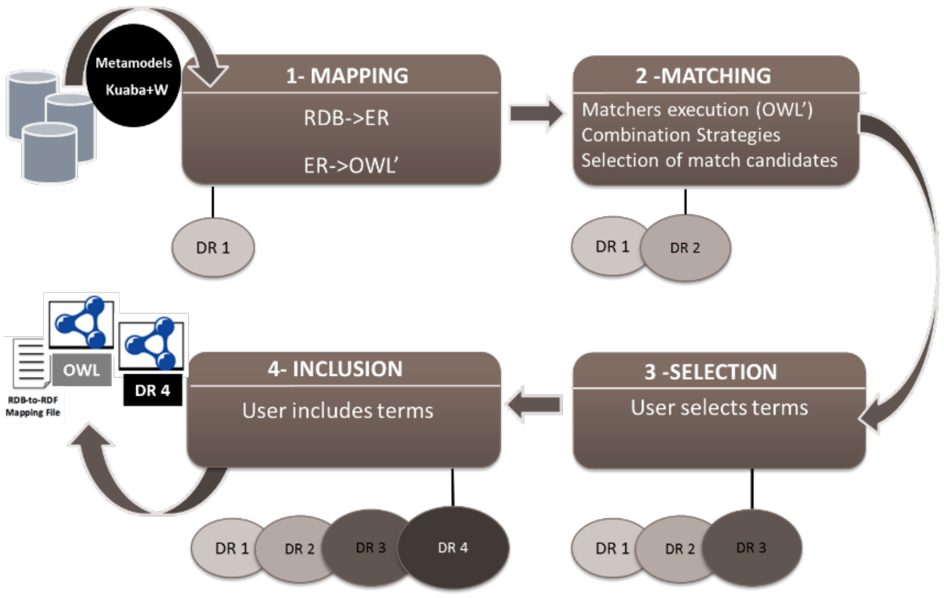


Figure 3.1 – The StdTrip+K Process.

3.3 StdTrip+K – An Example

3.3.1 Overview

We will use as example the publication database depicted in Figure 3.2, which will help explain the StdTrip+K process. We implicitly assume that the input database is fully normalized, i.e., the input to the mapping step must be in third normal form (3NF). Furthermore, we also assume that the user who follows this approach has some knowledge about the application domain of the databases.

We also incrementally introduce a visual representation of the design rationale captured during each step of the StdTrip+K process to facilitate understanding of the process.

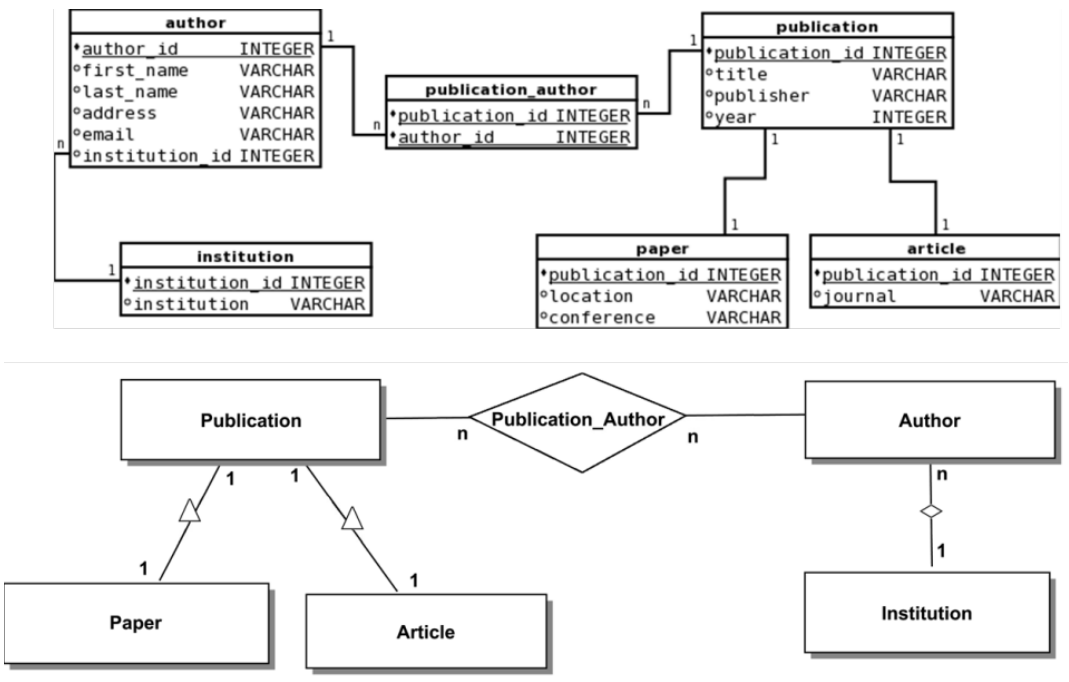


Figure 3.2 - Author-Publication relational schema and the corresponding ER diagram (Salas et. al.,2011).

3.3.2 Step 1 – Mapping

The general goal of this step is to map the structure of the input relational database schema into an intermediate database ontology (we call OWL') and to trace the design rationale of the mapping (referred to as DR1). OWL' is not the final ontology because no matching algorithms is executed at this step. To achieve the general goal, there are two sub-steps: **(1.1)** the *RDB-to-ER* sub-step transforms the relational database schema into an entity relationship (ER) schema; and **(1.2)** the *ER-to-OWL'* sub-step transforms the ER schema into an OWL ontology (OWL').

The RDB-to-ER sub-step works as a preparation for the ER-to-OWL' sub-step (mapping), where existing reverse engineering techniques are applied to define an ER schema from a relational database schema. According to the definition in the StdTrip process (Salas et. al., 2010b), the RDB-to-ER sub-step consists of transforming the Entity Relationship model into an OWL ontology. The reason for breaking down the mapping step into separate operations is that mapping the relational database model directly to OWL would not properly map some of the attributes, such as binary relationship to object properties. For example, the table publication author (Figure 3.2), using the direct mapping direct

RDB to OWL approach, would result in the Class Publication author with publication id and author id as subject, while the RDB to ER to OWL approach would correctly result in two object properties, publication author and the inverse property has publication author (Salas et. al., 2010b). As the reverse engineering techniques are consolidated and do not involve user decisions, the design rationale is not traced (Casanova and Sá, 1984; Batini et al., 1991; Heuser, 2004). In the *ER-to-OWL* sub-step, mapping rules are applied to transform the elements from the ER model into OWL elements. The mapping rules used in each scenario depend on the preferences of each designer and this flexibility makes it important to trace the design rationale for this sub-step.

Figure 3.3 illustrates the output of the mapping step for our running example. As can be seen, the resulting (still intermediate) OWL ontology simply mirrors the ER schema of the input relation database depicted in Figure 3.2.

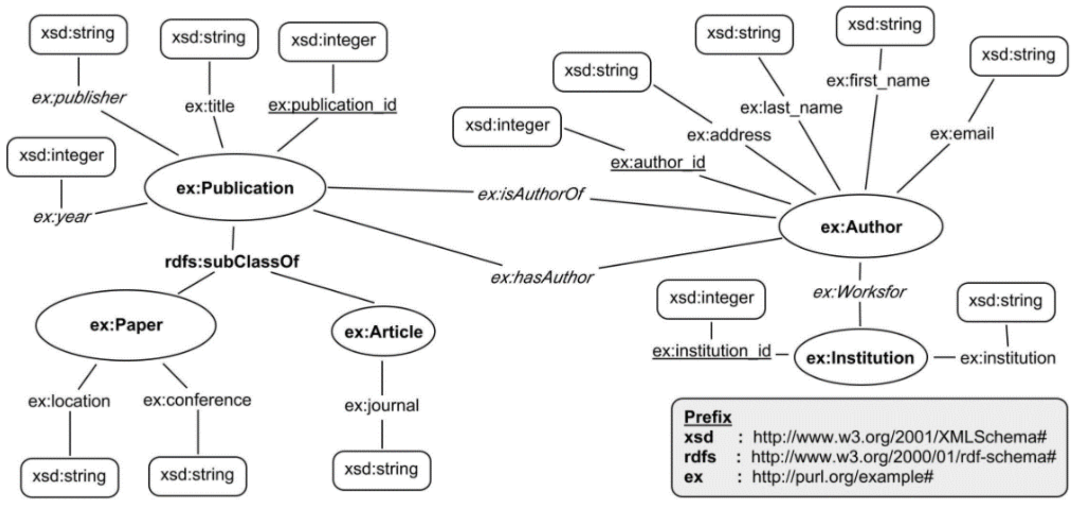


Figure 3.3 – Resulting intermediate OWL ontology after the second sub-step.

To illustrate the representation of the design rationale, we will consider only the mapping of the *Author* and *Institution* classes with their attributes and the relationship between them, *ex:WorksFor*, as shown in Figure 3.3.

To represent the rationale of the mapping step, StdTrip+K involves a list of steps that use the Kuaba+W model. All these steps are executed in a transparent way, because everything is automatically captured without interfering the central activity of the process, which is to build an OWL ontology. To do so, three artifacts are integrated to help in the automatic construction of the rationale model: the ER metamodel (Heuser, 2004); RDFS[6], which is the RDF metamodel; and the *mapping rules*, specific of each context. We list below the K-steps executed to capture the design rationale of the mapping step (DR 1), with the respective examples, for each step regarding the construction of DR 1 depicted in Figure 3.4, jointly presented with DR 2, which is the design rationale of Step 2. We use the step number to refer to each DR; for instance, in step 1, we use DR1.

K1 - Identify reasoning elements from the ER model. The reasoning elements *last_name*, *author*, *Author_Institution* and *Institution* were identified, because all of them are elements that will be mapped.

K2 - Identify the representation of the reasoning element in the ER model. After having identified each reasoning element, the rationale representation records which element (Entity - ENT, Attribute - ATT, Relationship - REL) it represents in the ER model to keep the traceability of each element. This representation is formalized through the *Element?* question, according to the Kuaba+W design rationale model.

K3 - Record the corresponding mapping of the ER element onto the OWL element. Having identified all the ER elements, the design rationale model records the correspondent OWL element mapped for each reasoning element, such as onto a class, onto an object property, and so on. This is represented by the question “Map?” according to the Kuaba+W design rationale model. For instance, the element ENT Author was mapped as the element Class “ex:Author”.

K4 - Record the argument for the mapping. For each reasoning element, the argument is the respective mapping rule used in the mapping. As the mapping rules are not rigid nor a consensus, this step records how each element was mapped as an argument form. The artifact of mapping rules is integrated in order to complete the rationale representation. For instance, the argument to map *last_name* onto *datatype property* is the rule “*Map each simple attribute of entity in the ER onto a functional datatype property*”.

[6] http://www.w3.org/TR/rdf-schema

K5 - Record the corresponding OWL intermediate term. According to the Kuaba+W model, the questions *Domain?* and *Range?* arise according to the specific element in the RDFS metamodel. For instance, for the mapping of the element REL “Author Instittuion” as an element “Object Property”, RDFS defines that every object property has a *domain* and a *range*, so these questions appear with the respective answers *ex:Author* and *ex:Institution*.

3.3.3 Step 2 – Matching

The general goal of this step is to find correspondences between the intermediate ontology obtained in the Step 1 - Mapping and standard well-known RDF vocabularies. This step comprises three sub-steps:

Ontology Matching. In StdTrip+K, this activity is supported by an ontology matching tool. For the running example, we used the K-match ontology alignment tool, which allows the use of different alignment matchers, including new ones, and the combination of the results by applying different strategies (Euzenat and Shvaiko, 2009; Euzenat et al., 2009). The inputs to K-match are the intermediate ontology obtained in Step 1 and a list of common OWL vocabularies; the outputs are similarity values between each element, ranging from 0 to 1 (Salas et al., 2010). For each element in the intermediate ontology, there are partial candidates according to each matcher, with their respective similarity values. For instance, at the end of this sub-step, for the element *ex:last_name* from the intermediate ontology, there are three different similarity values for the pairs *ex:last_name/foaf:first_name*, and *ex:last_name/foaf:givenName* according to the three matchers adopted (Lily, Aroma, Anchor-Flood).

As this sub-step is a preliminary matching and does not involve any user decision, the design rationale is not traced. Although it might be interesting to have the design rationale traced at this sub-step, it could overload the rationale representation, as this sub-step involves too many details with respect to similarity values, which can compromise the rationale exploration.

Combination strategies. In this sub-step, aggregation strategies are applied to define a unified similarity value for each pair of ontology terms. As we have used the K-match, one of the five aggregation strategies can be used – Max, Weighted, Average, Min and Harmonic mean. For instance, at the end of this sub-step, there is only one similarity value for the pair *ex:last_name*/*foaf:first_name* and another for *ex:last_name*/ *foaf:givenName*.

For the same reasons of the previous sub-step, the design rationale is not traced in this sub-step.

Selection of matching candidates. Until this sub-step, there is still more than one match for each term. Thus, the final sub-step aims at applying a selection strategy to choose one final match candidate for each ontology term. After having applied the strategy, a list of candidates for each term from the intermediate ontology is presented to the expert domain and then he or she is responsible for choosing the more adequate term according to the database context.

In this sub-step of Step 2, the user has an important participation in the StdTrip+K process, since he or she must make decisions regarding the final OWL ontology, mostly based in his or her experience on the database domain. For these reasons, the design rationale (DR 2) is captured.

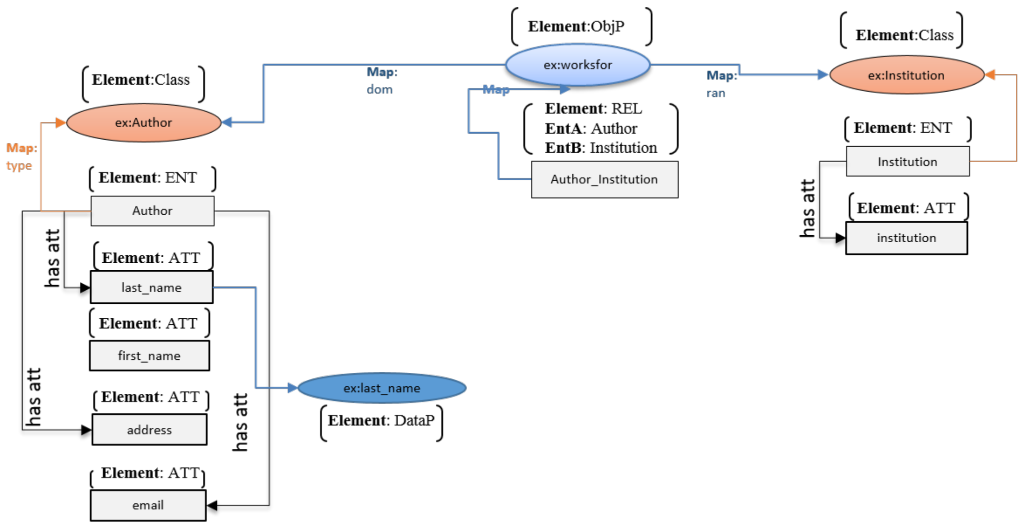


Figure 3.4 - DR1 and DR2 Resulting rationale model after Mapping and Matching steps.

The design rationale representation of Step 2, DR 2, increments the design rationale representation of Step 1 (DR 1) by executing the following steps, as shown in Figure 3.4:

K6 - Record the candidates for each intermediate term. After having applied the aggregation strategies, the Kuaba+W design rationale model records each candidate that is presented to the domain expert. This record is formalized through the *Match* question according to the definition of the StdTrip+K. For instance, the answers for this question for the term *ex:last_name* are *foaf:familyName* and *foaf:givenName* according to the K-match algorithms.

K7 - Identify and record the arguments (*in favor of* and *objects to*). For each candidate, there is a final similarity value that represents the reason for this candidate to be part of the list presented to the domain expert. This final similarity value is the argument for the design rationale representation. As the Kuaba+W design rationale model defines arguments as "*in favor of*" and "*objects to*", they have to be identified and traced to keep all options the user currently has to make his or her decision.

In StdTrip+K, the "in favor of" argument is the largest final similarity value, since it indicates a high possibility to be chosen by the expert domain. Conversely, lower values indicate the "objects to" arguments. This assumption does not affect the final decision about which term will be considered in the final ontology, but it is rather a way of automatically building the rationale model before the user decision. Due to space constraints, we illustrate only one case of different options with arguments *in favor of* and *objects to*, associated to *ex:last_name* example.

3.3.4
Step 3 – Selection

The general goal of this step is to select the terms resulting from the previous steps in order to build the final OWL ontology. In this step, user interaction plays an essential role. Ideally, the user should know the application domain because he or she has to select the vocabulary elements that best represent each concept in the database. For instance, in the case of the term *ex:last_name*, the user has to decide between the terms *foaf:givenName* and *foaf:familyName* with 0.83 and 0.77 similarity values respectively.

During this step, the most important decisions are made in StdTrip+K, where the domain expert chooses which term should be used. As this is the step has substantial impact on the final OWL ontology, the traceability of the design rationale becomes essential to minimize the possibility of errors insertion. Similarly to the previous DR models, the design rationale of this step (DR 3) is incrementally constructed from the preceding DR (DR 2) by executing the following steps:

K8 – Record the user decision domain about each term. In this step, the Kuaba+W model records all decisions involved in the acceptance (A) or rejection (R) of each term recommended by StdTrip+K. In the DR 3 model, these decisions are represented by the letters *A* and *R,* respectively.

K9 – Record the justification of the domain expert. After each decision, the user expert justifies his or her choices. This justification is the most important step of the final rationale representation. Figure 3.5 shows the rationale representation composed of the decisions and justifications regarding each expert decision. An example that represents the relevance of tracing the DR of this step is related to the term *ex:last_name*, for which the expert domain decided to use the term with the lowest similarity value, and without the DR it would not be possible to know why. In addition, without user expert interference, the final OWL ontology could be built inadequately according to the expert's expectations, since it would be natural to accept the term that has the highest similarity value.

3.3.5 Step 4 – Inclusion

The general goal of this step is to complete the final OWL ontology with terms that were not identified in the previous steps. This can happen when the selection step does not yield any results (there is no element in the standard vocabularies that matches the concept in the database) or when the user considers as inadequate all suggestions in the list. In such cases, StdTrip+K provides a list of terms from other vocabularies on the Web that might provide a possible match.

This step is executed only if a concept has not been covered by any of the known standards. In this case, the StdTrip+K recommends that the domain expert look around and see how other users dealt with this issue. As this step includes decision making, the rationale representation needs to be updated according to the last activity to complete the design rationale model DR 4, executing the following step:

K10 – Record the new term and the justification. The expert domain justifies the inclusion of a description which explains why this is an appropriate term in the input database context. In the Publication-Author example, Figure 3.5 shows that the term *ex:worksfor* is a term with no matching found, and in that case the same term of the intermediate ontology is kept. At this point, we have the entire rationale representation having executed the most important steps during the creation of the OWL ontology.

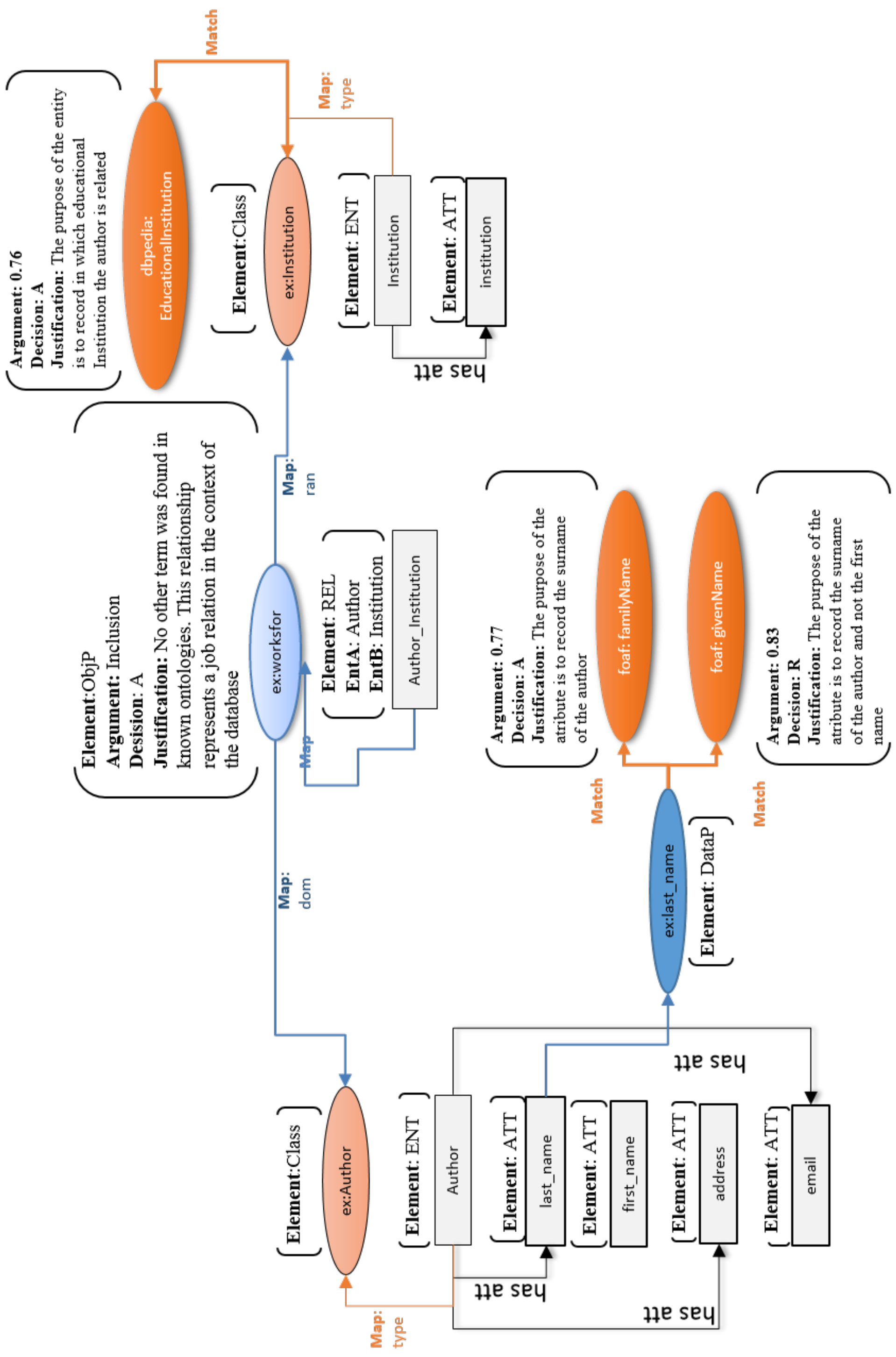


Figure 3.5 -Resulting rationale model, illustrating the rationale related to the ex:Author, ex:last_name and ex:Institution elements.

3.4
Architecture of the StdTrip+K Tool

Figure 3.6 shows the architecture of the StdTrip+K tool. It consists of five components composed by specific modules responsible for individuals activities, explained below:

I. *Conversion* – the component responsible for managing the extraction of metadata from the relational database, (such as table names, column names, column datatypes, primary key columns, foreign key columns and columns descriptions) and transforming from the relational database to the OWL ontology – these activities are organized in specialized modules, including the module named "Rationale Model" that is present in all components with the goal of collecting the rationale of the OWL ontology creation;

II. *Alignment* – the component responsible for executing alignment operations between the input database ontology and others standard ontologies through the implementation of K-Match tool (Do, 2006) – this component has also a module that, incrementally, collects the rationale of the process;

III. *Selection* – the component responsible for manipulating the match candidates obtained as the result of the Alignment component to elaborate the suggestions and then, a user may select the term that he or she considers the best match for each term – all these decisions, arguments and justifications are manipulated by the "Rationale model" module to record this model being built;

IV. *Inclusion* – responsible for giving support to definitions of new terms that no matching satisfied the expert user, or were not found – in this component we applied a keyword-based search for the ontology terms not considered in the previous modules – this inclusion module is also captured by the Rationale model module; and

V. *Output* – component responsible for generating the output, which is composed by 3 modules: in the ontology module the original data schema labels are substituted by ontology terms selected in the Selection module, in the Mapping File module we use the ontology terms selected or produced in previous steps to generate the mapping specification file and, finally, the Rationale Model that illustrates all the rationale behind the decisions made during the OWL ontology construction.

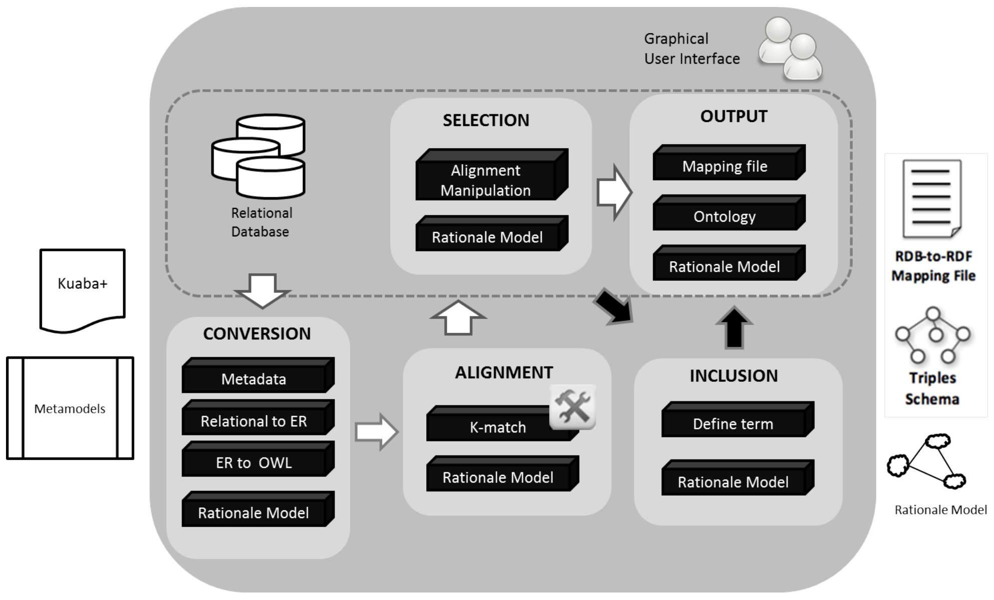


Figure 3.6 – StdTrip+K Architecture.

3.5
An Example of Output Files

To exemplify each one of the output files produced by the output component, consider our running example, the Author-Publication database depicted in Figure 3.2. Table 3.1 illustrates the Triple Schema fragment for the Author-Publication running example, which is the result of Ontology module. It contains the original input database schema in the OWL format, with corresponding restrictions, and maximizing the reuse of standard vocabularies.

Table 3.1 - "Triples Schema" for the Author-Publication example.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:bibtex="http://purl.org/net/nknouf/ns/bibtex#"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
...
<rdfs:Class rdf:about="http://purl.org/net/nknouf/ns/bibtex#Article">
<rdfs:label xml:lang="en">Article</rdfs:label>
</rdfs:Class>
...
<rdf:Description
rdf:about="http://purl.org/net/nknouf/ns/bibtex#Article">
<rdfs:subClassOf
rdf:resource="http://purl.org/net/nknouf/ns/bibtex#Entry"/>
</rdf:Description>
...
<owl:DatatypeProperty
rdf:about="http://purl.org/net/nknouf/ns/bibtex#hasJournal">
<rdfs:domain
rdf:resource="http://purl.org/net/nknouf/ns/bibtex#Article"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
<rdfs:label xml:lang="en">hasJournal</rdfs:label>
</owl:DatatypeProperty>
...
<owl:ObjectProperty rdf:about="http://purl.org/example#worksFor">
<rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
<rdfs:range
rdf:resource="http://dbpedia.org/ontology/EducationalInstitution"/>
<rdfs:label xml:lang="en">worksFor</rdfs:label>
</owl:ObjectProperty>
...
```

As a result of the Mapping File module, we have the fragment of the mapping specification file for the Triplify Tool (Auer et al., 2009), which is not a fixed option, being easily possible to customize for R2RML (Das et al., 2012) format, for instance.

Table 3.2 - Fragment of the Triplify mapping file for the Author-Publication.

```
$triplify['queries']=array(
'article'=> "SELECT
                publication_id as 'id'
                , journal as 'bibtex:hasJournal'
              FROM article",
 'author'=> "SELECT
                author_id as 'id'
                , institution_id as 'ex:worksFor'
                , first_name as 'foaf:firstName'
              , last_name as 'foaf:familyName'
              , address as 'dbpedia:address'
              , email as 'foaf:mbox'
          FROM author",
   ...);

  $triplify['classMap']=array(
              "article" => "bibtex:Article",
              "author" => "bibo:Person",
   ...);

   $triplify['objectProperties']=array(
            "ex:worksFor" => "institution",
            "bibtex:hasAuthor" => "author");
    ...
```

As a result of the Rationale model module we have the RDF representation of the rationale model built during the execution of all components in StdTrip+K according to the Kuaba+W ontology. Table 3.3 represents a fragment of the final rationale model created during the execution of the StdTrip+K process. To facilitate the association of the fragment with the model example, we show the correspondent fragment of the model example, as can be seen in Figure 3.7 .

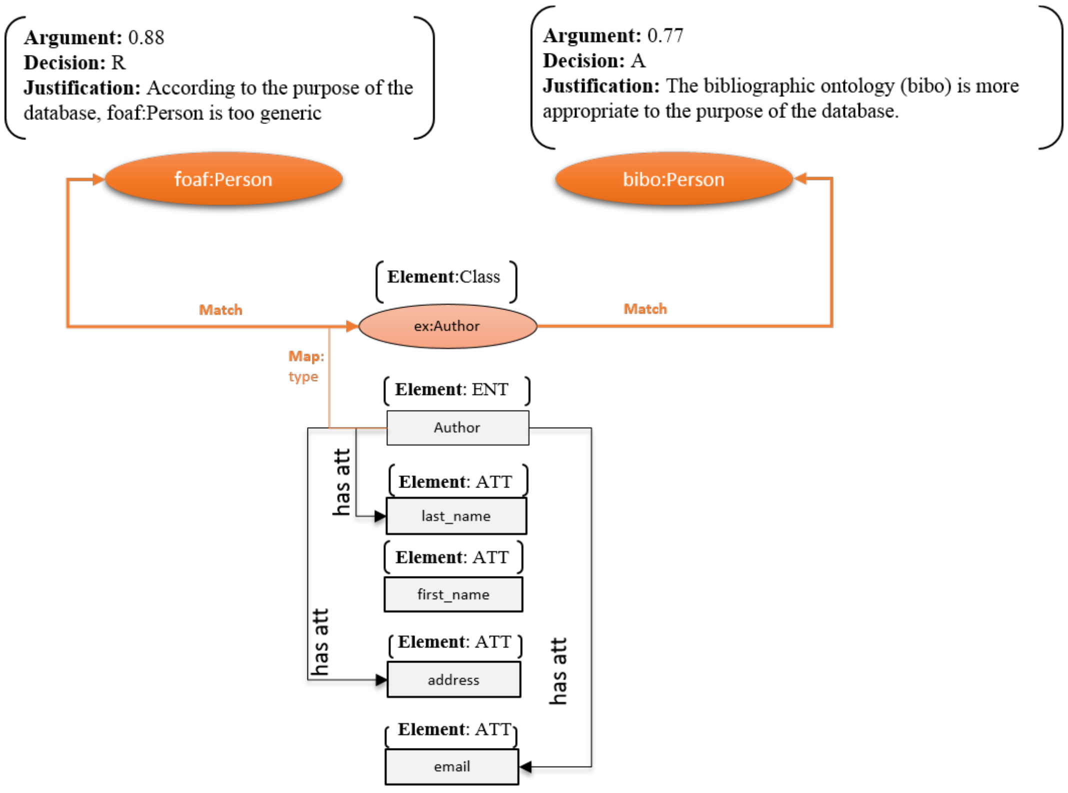


Figure 3.7 – Fragment of example correspondent to Table 3.3.

In Figure 3.7 we show an example for representing a design rationale of the Entity "Author" that was initially mapped with the same term of the relational database *ex:Author*, then the step Matching of StdTrip+K process was executed and it received as recommendations two terms *foaf:Person* and *bibo:Person*. The definition of *foaf:Person,* according to the FOAF ontology in *http://xmlns.com/foaf/spec/#term_Person,* says that "The Person class represents people. Something is a Person if it is a person. We don't nitpic about whether they're alive, dead, real, or imaginary." The definition of *bibo:Person* according to the BIBO ontology in *http://bibliontology.com/* says that "Used to describe a Person related to a bibliographic item such as an author, an editor, etc.". Observing the definitions and the purpose of the database, specifically the purpose of the class *Author* , it is easy to understand why the term *foaf:Person* was rejected (Decision:R) by the domain expert. However, this analysis is possible when the context of the relational database is known. After the relational data is already published, this context is lost. Then, the access of the information recorded by the DR in the question "Justification" becomes quite important. In this question the domain expert justifies the decision of Acceptance for the term *bibo:Person* with the sentence "The bibliographic ontology (bibo) is more appropriate to the purpose of the database.". Here, the design rationale is able to give a context of the relational database, and besides that, it gives information to other domain expert that is doing a mapping based on similar relational databases. One could argument that *foaf:Person* is not a wrong recommendation, what is a correct assertion. However, *foaf:Person* is a generic term and does not say anything about which kind of person the ontology is talking about, an author, a singer, an architect.

Another beneficial aspect of having the design rationale, is about the role of the justification question. Note that the term *foaf:Person* presents a greater similarity value (Argument:0.88) if compared to the similarity value of *bibo:Person* (Argument:0.77). Without the justification information, a user could think that the decision of rejection for the term with the greatest similarity value is mistaken. In addition, here is also important to highlight the importance of the presence of a domain expert with a contextual information about the relational database to decide which term better represent the data. Without knowing the relational database, the term *foaf:Person* would be chosen and the complete representation of the relational data in the RDF format could be affected.

Finally this example shows that the DR can help the users of the database to understand the context of the RDF dataset having access to the justification information. In addition, the DR can help to new domain experts that are defining ontologies to map similar relational database to RDF format.

Table 3.3 - Fragment of the Rationale model file for the Author-Publication.

```
Prefix ex:http://purl.org/example#.
              rv:http://purl.org/rationalevocab#.
              rdfs:http://www.w3.org/200001rdf-schema#.
              foaf:http://xmlns.com/foaf/0.1/
ex:author1 rdf:type rv:ReasoningElement;
                     rv:describesRE ex:Description_author;
                     rv:suggests ex:Element1;
ex:Description_author1 rdf:type rv:Description;
                          rdfs:label "author is a Reasoning Element to
be mapped";
                          rv:relatedTo ex:MetamodelER.
ex:MetamodelER rdf:type rv:Description;
                          rdfs:label "Metamodel ER".
ex:Element1 rdf:type rv:Question;
                rv:describesRE ex:Description_Element1.
ex:Description_Element1 rdf:type rv:Description;
                          rdfs:label "Which Element is this reasoning
element in the ER?";
                          rv:relatedTo ex:MetamodelER.
ex:Entity1 rdf:type rv:Idea;
               rv:describesRE ex:Description_Entity1;
               rv:address ex:Element1;
               rv:suggests ex:Map1.
ex:Description_Entity1 rdf:type rv:Description
                          rdfs:label "This Reasoning Element is an
Entity in the ER";
                          rv:relatedTo ex:MetamodelER.
ex:Map1 rdf:type rv:Question;
          rv:describesRE ex:Description_Map1.
ex:Description_Map1 rdf:type rv:Description;
                      rdfs:label "This Reasoning Element is mapped to
which element
in owl in the intermediate mapping?";
                      rv:relatedTo ex:MappingRules.
ex:MappingRules rdf:type rv:Metamodel;
                   rdfs:label "Metamodel Mapping Rules".
ex:Class1 rdf:type rv:Idea;
              rv:describesRE ex:Description_Class1;
              rv:address: ex:Map1;
          rv:suggests ex:Term1;
ex:Description_Class1 rdf:type rv:Description;
                      rdfs:label "This Reasoning Element is mapped
to owl Class in the
intermediate mapping.";
                      rv:relatedTo ex:MappingRules.
ex:Argument1 rdf:type rv:Argument;
             rv:inFavorOf rv:Class1;
             rv:describesRE ex:Description_Argument1.
ex:Description_Argument1 rdf:type rv:Description;
                         rdfs:label "Map each entity in the ER into
a Class in the owl ontology.";
                         rv:relatedTo ex:MappingRules.
ex:Term1 rdf:type rv:Question;
         rv:describesRE ex:Description_Term1.
ex:Description_Term1 rdf:type rv:Description;
                     rdfs:label "Which intermediate term is mapped to
this Reasoning Element?";
                     rv:relatedTo ex:MetamodelRDFS.
ex:MetamodelRDFS rdf:type rv:Metamodel;
                  rdfs:label "Metamodel RDF schema".
ex:author rdf:type rv:Idea;
          rv:describesRE ex:Description_ex:author;
          rv:address: ex:Term1;
          rv:suggests ex:Candidates1;
ex:Description_ex:author rdf:type rv:Description;
                         rdfs:label "ex:author is an intermediate
term.";
                         rv:relatedTo ex:StdTripProcess.
ex:StdTripProcess rdf:type rv:Metamodel;
                  rdfs:label "This idea is obtained by the expert
according the respective step in StdTripProcess.".
ex:Candidates1 rdf:type rv:Question;
               rv:describesER ex:Description_Candidates1.
ex:Description_Candidates1 rdf:type rv:Description;
                           rdfs:label "Which are the candidates for
the intermediate term?";
                           rv:relatedTo ex:StdTripProcess.
foaf:Person rdf:type rv:Idea;
            rv:describesRE ex:Description_foaf:Person;
            rv:address: Candidates1;
ex:Description_foaf:Person rdf:type rv:Description;
                           rdfs:label "foaf:Person is a candidate
term.";
                           rv:relatedTo ex:MetamodelStdTrip.
ex:Argument2 rdf:type rv:Argument;
             rv:inFavorOf foaf:Person;
             rv:describesRE ex:Description_Argument2.
ex:Description_Argument2 rdf:type rv:Description;
             rdfs:label "Similarity 0.88";
ex:Decision1 rdf:type rv:Decision
```

```
80:              rv:relatedToquestion rv:Candidates1;
81:              rv:relatedToidea foaf:Person;
82:              rv:accepted "TRUE";
83:              rv:hasJustification ex:Justification1.
84:Ex:Justification1 rdf:type rv:Justification;
85:                  rdfs:label "According to the definition it is the
best choice".
86: …
```

As this output is the main contribution of the entire process presented here, we explain each association of the code fragment with the part of the visual model that we have used in the chapter. The metamodels used to build this output are the ER Metamodel and the StdTrip+K metamodel, that is the full description of the StdTrip process (Salas 2010b). This explanation is done in lines blocks, such as follows:

- Lines 2-9 : it declares the identification of the reasoning element (here lets mention this as RE) "author" and, according to the KUABA+W ontology, this RE suggests a question according to the ER Metamodel since this reasoning element were extracted from the input database ER model.
- Lines 10-14: as it is said in the KUABA+W, an ER suggests a question, so here the question "Element?" is described as "Element1". Note that in the RDF output we use the strategy to differentiate the same questions, such as "Element?" adding an ordinal in the final of the term. We identified this differentiation as a key of traceability in the final RDF, i. e., if we do not differentiate, it would be costly to identify to which reasoning element the question is regarding to and the Ideas consequently.
- Lines 15-21: this block contains the "Idea" according to the Kuaba+W ontology, that says that an Idea addresses a question. The idea "Entity" answers the question – extracted from the ER metamodel – "Which element the RE represents in the ER metamodel?". The answers for this question are limited in three options according to the ER metamodel: entity, attribute or a relationship; in this case it is an entity.
- Lines 22- 29: this block contains the question that the previous answer suggests, i.e. it describes the question "Map?" originated from the Mapping rules.

- Lines 30-37: this block describes the answer “Class” for the question in the previous block. In this case the mapping is for a class for reasons explained in block of lines 38-43.
- Lines 38-43: this block explicit the Argument for the mapping done of the reasoning element “author”, which is an Entity that is mapped to a Class according to the Mapping rules.
- Lines 44-50: according to the process after been identified to which RDF element is mapped the reasoning element of the input database, it is defined an intermediate term, so this block describes this question “Term?”, that arises from the answer “Class”.
- Lines 51-60: this block describes the Idea “ex:author” which addresses the Question previously described. Note that this idea is derived from the description of StdTrip process (StdTrip+K Metamodel), as mentioned in this block.
- Lines 61-65: outlines the Question “Candidates?” also from the description of the StdTrip+K process, that is suggested by the idea “ex:author” because the following step of the process is to define candidates terms through matching tools.
- Lines 66-73: this block is the Idea for the Question above described. The idea is a result of matching tools of StdTrip+K process.
- Lines 74-78: the Argument of the Idea above described is recorded in this block whith the explicit information of the similarity value declared.
- Lines 79-83: the Decision regarding the Idea that represents the term “foaf:Person” is declared in this block with the property “rv:accepted TRUE”.
- Lines 84-85: the Justification regarding the Decision is recorded in this block.
- Lines 86-…: the rest of the RDF regarding the other reasoning elements.

3.6 Related Work

There are several approaches, with different mechanisms, to tackle the RDB-to-RDF translation process. It is important to note, however, that old RDB-to-RDF approaches provide different, proprietary mapping languages for the mapping process.

As an alternative, there was an effort to establish standards to govern this process, such as W3C RDB2RDF WorkingGroup[7], which proposed a standard language to express RDB-to-RDF mappings, called R2RML (Das et al., 2013). Although it is an important initiative, this language does not support the record of decisions made during the mapping process.

Relevant RDB-to-RDF tools include: (i) Triplify (Auer et al., 2009) offers a simple mapping solution using SQL (Structure Query Language) as a mapping language and transforms database query results into RDF triples and Linked Data; the mapping is manually done with no support for design rationale; (ii) D2RQ (Bizer andSeaborne, 2004) automatically generates the mapping files, using the table-to-class and column-to-predicate approach, using a declarative language, implemented as a Jena graph, to define the mapping file; it also does not record the design rationale; (iii) Virtuoso RDF view (Erling and Mikhailov, 2009) supports mapping files, also called RDF views, automatically generated with the direct table-to-class approach; in this approach there is no reason to capture the rationale since it does not involve options, arguments and decisions; (iv) RDBtoOnto (Cerbah, 2008) brings a discussion on how to take advantage of database data in obtaining more accurate ontologies; it also uses the direct table-to-class and column-to-predicate approach to create an initial ontology schema, which is then refined through identification of taxonomies guided by the tool' although there is user interference, there is no record of the decisions made by the user. Other approaches, such as like DB2OWL (Cullot et al., 2007) and Ultrawrap (Sequeda et al., 2009), also do not cover the rationale issue.

In the context of rationale models and tools, there are argumentation-based models, such as IBIS (Kunz and Rittel, 1979), DRL (Lee and Lai, 1991), QOC (Maclean et al., 1991), that cover DR representation. However, they do not present a complete DR that includes acceptance and rejection options and the reasons for that.

We have not found research that specifically addresses design rationale in the context of Linked Data. There are provenance models, such as the Open Provenance Model (OPM[8]), that records the history of the creation of a dataset in

[7] http://www.w3.org/2001/sw/rdb2rdf/

[8] openprovenance.org/

general terms. Despite being very important and essential for Linked Data quality, OPM does not cover the decisions taken during the definition of a mapping file.

To summarize, neither the RDB-to-RDF approaches capture design rationale nor the DR models cover the essence of the RDB-to-RDF mapping process, including the reuse of existing standard vocabularies. Since we believe that the reuse of standard vocabularies and the record of DR are potential ways to reduce and to guarantee future interoperability (Breitman et al., 2006), we proposed the StdTrip+K process, detailed along the Section 3.3.

3.7 Conclusion

In this chapter, we introduced StdTrip+K, a process that allows the translation of a relational database to RDF triples, reusing standard vocabularies and recording the design rationale of the translation, thereby providing more objective information about the RDB-to-RDF process.

StdTrip+K helps solve the lack of an integrated approach with support for the capture, representation, and reuse of DR as part of the process of translating databases to RDF triples. With the help of an example, we have shown this integration. The capture is addressed through the insertion of K-steps among the StdTrip+K steps, deploying an automatic strategy for collecting DR according to Kuaba. The representation is dealt with through the use of the new vocabulary Kuaba+W, developed specifically to be able to explicit the actions to which the original database was submitted. The reuse of the DR is possible once it is published in the RDF triples format.

We have shown that the task of creating an ontology is still very dependent on domain knowledge, which sometimes only an expert can express in an adequate ontology. The contrasts found between the assumptions that an ontology is easily created by just transforming entity to classes, attribute to properties and using ontology matching algorithms, and the final decisions of the domain expert are the most interesting contributions of this research.

Via the StdTrip+K output artifacts – the mapping file, the triple set and the DR – it is possible to answer the questions that still arise when using triple sets in the Linked Data cloud. Consider first the questions:

- *Has the database suffered changes when published as RDF triples that could impact in its quality? May the original relational database have lost some relevant information when it is published as RDF triples?*

Since the DR captures the original design of the database (as an ER schema) and shows the mapping of each ER element to an RDF element, it is possible to answer these questions.

Consider now a second pair of questions:

- *Is the chosen ontology the most appropriate to represent the database? Is the translation correct from the original relational database to RDF?*

Since the DR shows the options abandoned and accepted and the reasons for that, it is possible to judge the choices made. Also, the DR allows access to one-to-one and one-to-many mappings, although the example in this chapter did not discussed these cases.

Regarding the implementation, the capture and representation of the DR in the StdTrip+K process was not yet implemented. The implementation can be consolidated following the architecture described in this Chapter (Section 3.4), by modifying the existing tool of StdTrip or developing new modules that can be coupled in this tool.

4
The Role of Design Rationale in the Ontology Matching Step during the Triplification of Relational Databases

4.1
Introduction

In this chapter, we consider the direct approach, where a vocabulary is directly defined from the relational database schema, resulting in a *database schema ontology* (DSO). 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 leverage the complete advantage of Linked Data since they represent a semantically weak description of these concepts.

We present in this chapter a process, called StdTrip 2.0, which 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 record 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 (Berardi et al., 2013; Salas et al., 2010b) presented in the previous chapter. 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 direct graph; and (iii) it includes a specific annotation strategy.

4.2 The StdTrip 2.0 Process

As depicted in Figure 4.1, StdTrip 2.0 includes a new step to cover 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.

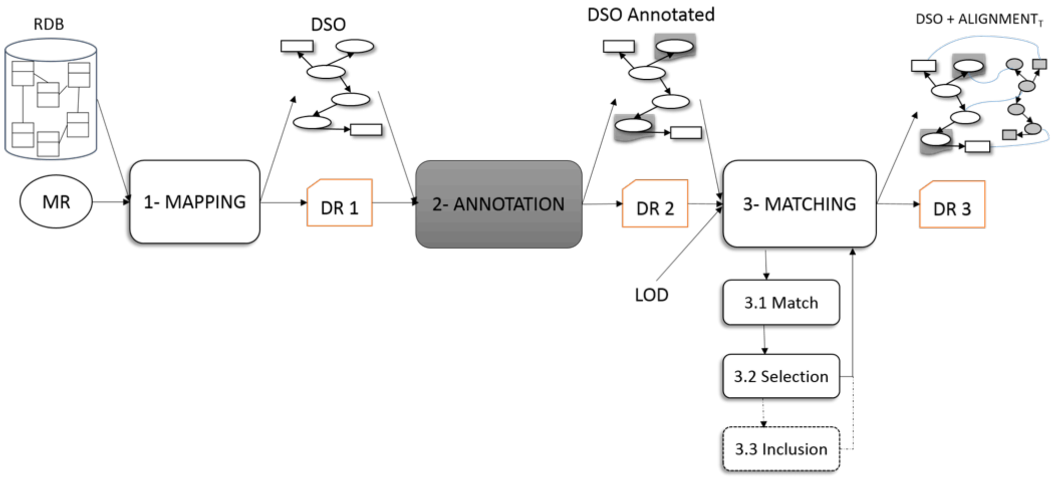


Figure 4.1 - The StdTrip 2.0 process.

We use as an example the same database used in the Chapter 2. Figure 4.2 depicts the publication database that we use as an example. To exemplify a partially published relational database, we consider as private the grey part of Figure 4.2 (a) and (b)).

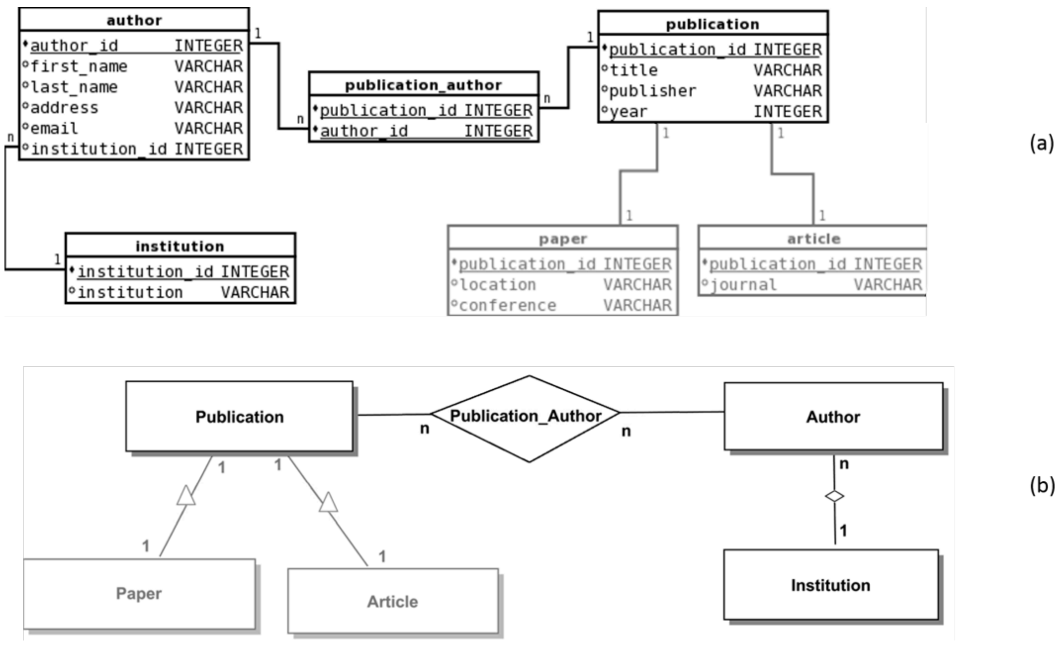


Figure 4.2 - The Author-Publication ER diagram (Salas et al., 2011).

The Mapping step of StdTrip 2.0 keeps the same activities of this step in the StdTrip+K process, discussed in Chapter 3. It receives an Entity-Relationship (ER) extracted from an RDB schema (Figure 4.2) and a set of mapping rules (MR), defined by a domain expert. It outputs a DSO and the corresponding design rationale (DR1).

4.2.1
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 step consists of two sub-steps: (1) preparing the annotation; and (2) adding the annotation to the DSO. Our proposal for the first sub-step is to use the

DR graph for applying the annotation strategy. 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.

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. For example, in Figure 4.3, consider the node "Publication" that is mapped (*Map=owl:Class*). The first level of unmapped neighbors (Map=NOT) for the node "Publication" contains "Article" and "Paper"; the second level contains the nodes "location" and "conference". We adopted, as an initial strategy, to consider a maximum of two levels (depth = 3) 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. For example, a publication database where the authors are postgraduate students, the more distant relations of the authors may contain information about their grades, and their grades are not part of the publication context. As an example of output, Figure 4.3 shows the DR2 annotations that are ready to be added to the DSO ontology, resulting in DSO_A.

The algorithm to prepare the annotations (shown in Pseudo-Code 1) applies the search for unmapped neighbors in the direct graph DR2 (Figure 4.3). Lines 1 to 4 of Pseudo-Code 1 take each mapped node as an "anchor" node, search for unmapped neighbors and use their labels to generate annotations. Lines 6 to 9 guarantee that, when entities are unmapped, their attributes are used for the annotation as well, even if they are beyond the level established in the algorithm. This "extra" mapping is also part of our strategy because if an unmapped entity is used for the annotation, its attributes should also be part of the context as they are characterizing that entity.

Pseudo-Code 1 - Recursive function that implements the annotation process.

```
Input:
         nodes: Node[],
         level: Integer,
         anchor: Node,
         digraph: (Node, Edges),

1.  Function annotate(nodes, level, anchor,digraph)
2.      notMappedNeighbors <-getNeighbors(digraph,nodes,'ALL_ELEMENT','NOT'),
3.      anchor.annot+=extractAnnotation(notMappedNeighbors),
4.      level --,
5.  if level == 0
6.      notMappedAttributes<-getNeighbors(digraph,notMappedNeighbors,
    'ATT_ELEMENT','NOT'),
7.      For each attribute in notMappedAttributes
8.         anchor.annot+=extractAnnotation(notMappedNeighbors),
9.      Return
10. else
11.     annotate(notMappedNeighbors, level, anchor, digraph)
```

Pseudo-Code 2 - Function that calls the recursive function for all nodes in the DR graph.

```
1.  Function navigate(digraph)
2.     listOfPubNodes <- getPubNodes(digraph),
3.     For node in listOfPubNodes
4.       annotate(node,depth,node,digraph)
```

In line 4 of Pseudo-Code 2, the parameter *depth* of the function "annotate" defines which level the user wants to annotate. As an example of output, the annotations in the DR2 are ready to be added in the DSO ontology resulting in DSO_A, as can be seen in Code 2.

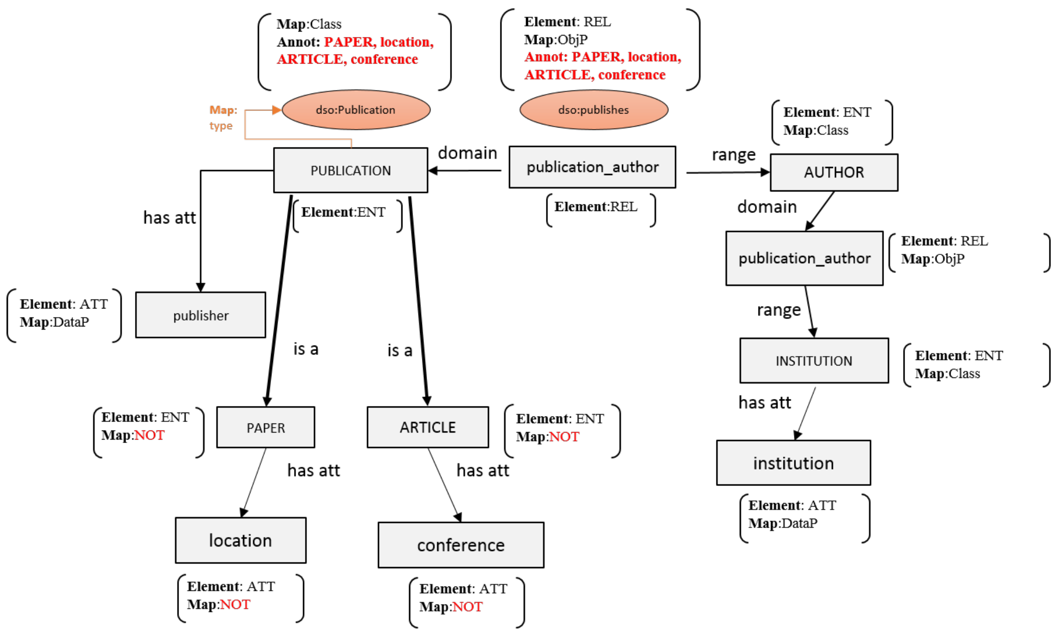


Figure 4.3 - Design Rationale 2 with annotations marked in red.

Code2: Example of a part of the DSO with the annotations marked in bold text DSO$_A$.

```
DSOA=
{(dso:Author rdf:type rdfs:Class),
 (dso:Publication rdf:type rdfs:Class),
 (dso:Publication rdfs:comment 'PAPER location ARTICLE conference),
 (dso:publishes rdf:type rdfs:Property),
 (dso:publishes rdfs:comment 'PAPER location ARTICLE conference),
 (dso:publishes rdf:domain ex:Publication),
 (dso:publishes rdf:range ex:Author),
}
```

Regarding the representation, in Figure 4.3, the RE "Publication" is a node with attributes "**Element**", "**Map**" and "**Annot**" that represent the trace of its mapping. These node attributes represent the "questions" of the DR1 and DR2 model defined in (Berardi et al., 2013), respectively: "Which **element** is it in the RDB?"; "How is it **mapped**?"; and "Which **annot** this element received?". 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. For the question "**Element**", the possible answers are abbreviations of elements of the Entity-Relationship model of a relational database. For example, the node "*Publication*" has the abbreviation "ENT" as the answer for the question "**Element**", which means that it was originally an entity in the RDB database. Table 4.1 summarizes the controlled vocabulary for the answers to the "**Element**" question.

Table 4.1: Controlled vocabulary for the Element node attributes in the DR graph.

Controlled Vocabulary	Meaning
ENT	represents that the node was originally an Entity type
REL	represents that the node was originally a relationship
ATT	represents that the node was originally an attribute

The edges in all DR are named according to a controlled vocabulary of the relation elements in the original database. Table 4.2 shows the controlled vocabulary for the edges. For example, for the "*Publication*" element, the edges between "*Paper*" and "*Article*" are named "*is a*" because, in the database, they originally represented a hierarchy connection. Note that, although the specialized entities are not mapped, they are represented in the DR1 graph.

Table 4.2: Controlled vocabulary for the Edges in the DR graph.

Controlled Vocabulary	Meaning
x is a *y*	represents that originally in the database *x* specializes *y*
x has att *y*	represents that originally in the database *x* has the attribute *y*
x domain *y*	represents that originally in the database *x* is the domain of *y*
x range *y*	represents that originally in the database *y* is the range of *x*

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". As this process is anchored in the *a priori approach* (Salas et al., 2010b), the domain expert may have modified some terms when defining the mapping rules. For example, he or she may have mapped a relationship originally called "publication_author" to "*dso:publishes*".

4.2.2 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 is subdivided into the same sub-steps as Step 3 in StdTrip+K (Section 2.4.3): (3.1) Matchers 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 semantics of 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 step is incrementally stored.

4.2.2.1 Ontology Matching Techniques

The previous version of StdTrip 2.0 (Salas et al., 2010a; Salas et al., 2010b; Berardi et al., 2013) used Ontology Matching algorithms that were in the state of the art list of OM participants in the last years in the OAEI challenge (Euzenat and Schvaiko, 2013), such as AROMA (David, 2009), Lily (Wang, 2009), Anchor Aflood (Sediqui, 2009). However, in StdTrip 2.0, in the previous version of StdTrip the Ontology Matching algorithms did not consider extra information of the ontology, such as definitions and comments. In StdTrip 2.0 we are adding annotations to a DSO and comparing it with ontologies rich of definitions and comments. This new feature caused instability in the OM algorithms used in StdTrip 2.0 since they apply different techniques in the three different layers: terminological, structural and semantic (Seddiqui, 2009). Using known and complete ontology matching (OM) algorithms with the three layers is very valuable, but it inserts too many variables to analyze, which can interfere with our approach. For that reason, we decided to proceed using OM techniques in one layer at a time. Therefore, we started with a layer that is used by almost every OM algorithms: the terminological layer.

A terminological layer uses similarity measures to discover alignments by comparing labels, comments and definitions (annotations). One important challenge related to the success of this layer is to select the most appropriate similarity measure (Seddiqui, 2009). The appropriate similarity measure is directly related to the goal of the measuring. Our goal is to measure if the private schema annotations in the DSO ontology help in finding more contextual results. The measures used in OM that go into this directions can be classified as syntactic, since they are based on the characters in the strings or the rules of the language in which the strings are written (Bellahsene et al., 2013). These measures, such as the Jaccard distance (Cheatam and Hitzler, 2013) in (1), penalize groups of strings with very different sizes and, consequently, result in low similarity values for them.

$$J_\delta(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|} \qquad (1)$$

For example, comparing a very small group of strings to a large group of strings, even if the cardinality of the intersection is large, the Jaccard distance gives a low similarity value. Consider these three sets of strings:

A = {author, publication, conference, paper, title, email, address, institution, year}

B = {author, publication, conference, biology, humans}

C= {author, music}

Intuitively the set B is more similar to A than the set C because the intersection is greater ($A \cap B \ > A \ \cap C$). However, the result of Jaccard similarity between A and B is slower (0.75) than A and C (0.91) due the difference of size.

For that reason, we need a measure that compares groups of strings but that does not penalize the difference in sizes. We propose a *subset string measure-SbS (2)* that gives the similarity ratio minimizing the influence of the difference in sizes between the groups.

$$SbS\ (A, B) = \frac{|A \cup B||A \cap B|}{|A||B|} \in \ [0,1] \qquad (2)$$

Intuitively, the Subset String (SbS) compares two lexical, non-empty entries A, B and satisfies the following properties:

(1) SbS(A,B) = 1 iff A⊆B ∨ B⊆A

(2) SbS(A,B) = 0 iff A∩B = 0

(3) SbS(A,B) ↑ ~ A∩B ↑, i.e., the SbS value increases as the intersection size also increases.

To illustrate the difference between Jaccard and SbS, consider the same 3 groups, while Jaccard says that C is more similar to A, SbS says that B is more similar with a similarity value of 0.72 while the similarity value between A and C is 0.55.

For the reasons discussed, we used the SbS measure to run the study cases. We believe that if we can achieve some improvements in this layer, we will have already achieved a promising result for our StdTrip 2.0 proposal.

4.3 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 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 3.3.4 and 3.3.5 (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.

4.3.1 Relational Databases used

We executed StdTrip 2.0 until Step 3.3.3 (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 (Cullot et al., 2007).

(2) *osCommerce*[9] database, with 48 tables where 36 of them were considered as private.

(3) *phpBB*[10] 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 (Auer et al., 2009). In the Triplify evaluation (Auer et al., 2009), the authors affirm that only parts of these databases were interesting for publishing as RDF.

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

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

In addition, the vocabulary selected to publish each of these databases was completely defined by a domain expert reusing existing vocabularies, such as FOAF[11], SIOC[12] and SKOS[13].

With that in mind, we decided to use the same databases, following the directions of the transformation from RDB to RDF. But, instead of depending only on a domain expert to define the terms to reuse, as in the Triplify evaluation, we used StdTrip 2.0 to help the domain expert by providing some recommendations based on the database context (annotations). In addition, we also decided to use a *Publication* database since it is a very simple and known context, it may show some differences against the other databases.

1. *Publication database*. It is a small database that simply stores publications of a research group in 6 tables. To simulate the partial publication of this database we considered 2 tables as private. The published part exposes all publications (without differentiation between journal articles and conference papers) of all authors of a research group in an institution.

2. *osCommerce database*. According to Triplify (Auer et al., 2009), *osCommerce Online Merchant* was one of the most popular open-source online shop e-commerce solutions. The database schema of *osCommerce* contains 48 tables, of which just 12 present valuable public information, still according to Triplify. The parts published refer to a hierarchy of product categories, lists of products and manufacturers and lists of reviews for products.

3. *phpBB database*. According to Triplify (Auer et al., 2009), *phpBB* is a very popular open source Web forum solution. The current version of this database schema consists of 30 tables, but only those about users, groups and posts are relevant information to be published as RDF.

[11] http://www.foaf-project.org

[12] http://www.sioc-project.org

[13]http://www.w3.org/TR/2008/WD-skos-reference-20080829/skos.html#SKOS-REFERENCE

4.3.2 Result Analysis

We organized the test in the following steps:

1. Define a Reference Alignment that is the expected set of terms recommended by the SbS(A,B) measure. It was manually defined according to our expertise about each relational database domain. 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 (Auer et al., 2009).
2. Define which domain ontologies will be input as “B” to the SbS (A,B) measure. It was also defined according to each relational database domain.
3. Run the SbS(A,B) measure to get a set of recommendations of terms from domain ontologies chosen at step 2. We call this set $T_{<n><dso>}$, where T is the set of terms, <n> is the name of the relational database and <dso> is the situation of the DSO, annotated or not. When the DSO is not annotated, nothing is declared in T (T_P), when it is annotated <dso> is “a” (T_{Pa}). We used a threshold of 0.3 to select what is part of the recommendations among the total of terms recommended by the SbS measure for each relational database.

   For the database Publication, we run the SbS measure as follows:

   a. SbS (DSO, (FOAF, BIBO, BIBTEX)) = T_P
   b. SbS (DSO_a, (FOAF, BIBO, BIBTEX)) = T_{Pa}

   We got a set of terms recommended without using annotations (T_P) and a set of terms recommended using annotations (T_{Pa}).

   For the database osCommerce, we run the SbS measure as follows:

   a. SbS (DSO, (SKOS, GoodRelations, SIOC)) = T_O
   b. SbS (DSO_a, (SKOS, GoodRelations, SIOC)) = T_{Oa}

   We got a set of terms recommended without using annotations (T_O) and a set of terms recommended using annotations (T_{Oa}).

   For the database phpBB, we run the SbS measure as follows:

   a. SbS (DSO, (FOAF, SIOC)) = T_B
   b. SbS (DSO_a, (FOAF, SIOC)) = T_{Bd}
   c. SbS (DSO, (FOAF, SIOC)) = T_{Bda}

We got a set of terms recommended without using annotations (T_B), a set of terms recommended using the dictionary of the database (T_{Bd}) and a set of terms recommended using the dictionary and annotations (T_{Bda}).

4. Calculate the Precision and Recall measures for each one of the sets of terms recommended at step 3. To calculate these measures we used the results of the SbS measure from step 3 and the Reference Alignment (RA) defined in step 1 for each one of the relational databases.
    a. Precision (T_P, RA), Recall (T_P, RA) for the Publication database
    b. Precision (T_{Pa}, RA), Recall (T_{Pa}, RA) for the Publication database
    c. Precision (T_O, RA), Recall (T_O, RA) for the osCommerce database
    d. Precision (T_{Oa}, RA), Recall (T_{Oa}, RA) for the osCommerce database
    e. Precision (T_B, RA), Recall (T_B, RA) for the phpBB database
    f. Precision (T_{Bd}, RA), Recall (T_{Bd}, RA) for the phpBB database
    g. Precision (T_{Bda}, RA), Recall (T_{Bda}, RA) for the phpBB database

Figure 4.4 shows the results for the *Publication* database. The input ontologies for the *SbS(A,B)* measure were for "A" the DSO (annotated or not) and for "B" the BIBO[14] and FOAF[15] ontologies. As we have previously discussed in Introduction Section 4.1, FOAF can be considered as a "general" context, then the ontology BIBO (bibliographic ontology) is more contextually related with the *Publication* database.

The relational schema of this database is the same schema used in the Chapter 4. Analyzing the precision value, the recommendations using the annotations (T_{Pa}, RA) obtained lower values, if compared with (T_P , RA) without annotation, but the recall of (T_{Pa} ,RA) got higher values, when compared to (T_P, RA) without annotations. This characteristic is common for approaches that use extra information (Euzenat and Shvaiko, 2007). 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 to the right term, thereby reducing precision.

[14] http://bibliontology.com/

[15] http://xmlns.com/foaf/spec/

An example of the test in this Publication database is the match for the term *dso:title* that without annotation received only *foaf:title* as recommendation, and with annotation it received *foaf:title*, *foaf:publication*, *bibtex:title* and *bibtex:booktitle* . As we argue, with annotation, the recommendation became more related to the context, as the *dso:title* is about a title of a reference. The usage of *foaf:title* would not be wrong, but the information about the context would be lost.

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 interference was not taken into account in the following steps of the process.

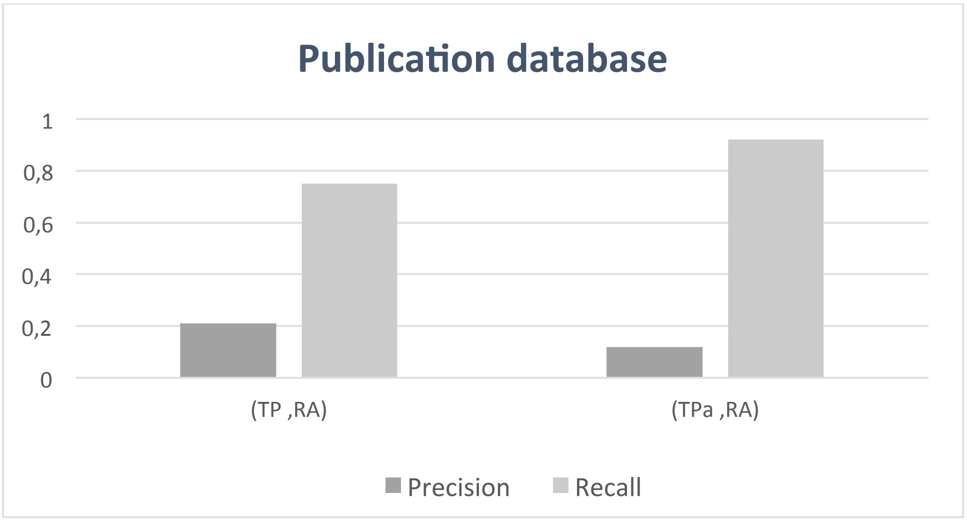


Figure 4.4 – Publication database results.

Table 4.3: Precision and Recall values for Publication database

	(T_P, RA)	(T_{Pa}, RA)
Precision	0,21	0,12
Recall	0,75	0,92

Figure 4.5 shows the results for the *osCommerce* database. The SKOS vocabulary was used to describe the hierarchy of products and categories,; for the lists of products and manufacturers, the Good Relations vocabulary was used; and for the list of reviews for products, the SIOC vocabulary was adopted. The list of vocabularies was defined according to the context of each part of the complete

osCommerce database. Due the size of the relational schema (50 tables) we do not show it here, but it can be accessed in the webpage[16].

Analyzing the precision and recall values, even with a much larger relational database (50 tables), we had the same results as the *Publication* database results.

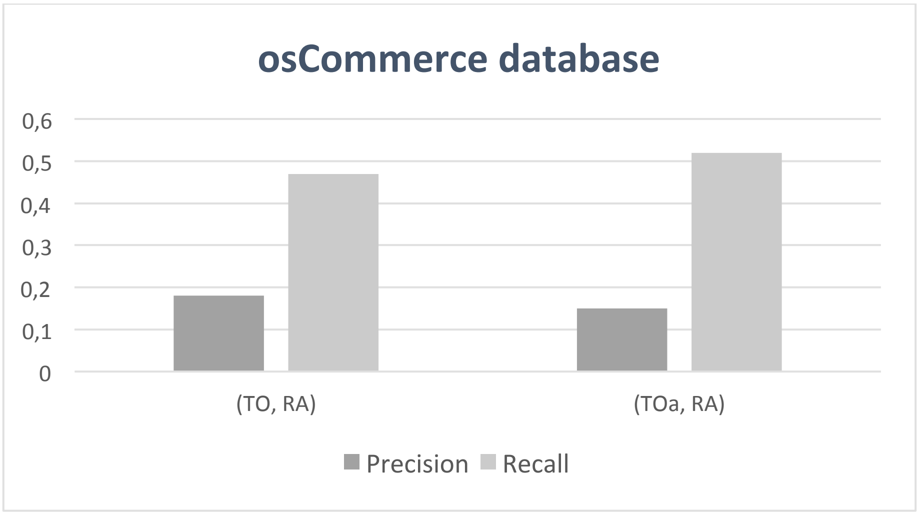


Figure 4.5 – osCommerce database results

Table 4.4: Precision and Recall values for osCommerce database

	$(T_{O,}$ RA)	$(T_{Oa}$,RA)
Precision	0,18	0,15
Recall	0,47	0,52

An example of an improvement in this database is the match for the term *dso:categories* – only using the annotations, the SbS was able to recommend the term *skos:OrderedCollection*, which is very specific for the context of this database. The *dso:categories* deals with classification of collection of products in a determined order.

Despite our positive interpretation for the recall values in both the *Publication* and the *osCommerce* databases, an important issue to explore is that, 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* (Euzenat and Shvaiko, 2007) measure that gives the proportion of non-relevant terms

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

recommended for reuse, out of all non-relevant terms available for reusing. The numbers of fall-out for *Publication* and *osCommerce* are respectively 0.21 and 0.01 which show that 1-0.21 (79%) of non-relevant data was not recommended, what is a good proportion. It shows that, although the group of recommendations is larger than before, it still helps the domain expert avoiding the need to analyze 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 chapter.

Figure 4.6 shows the results for the *phpBB database*. Similarly to the osCommerce database schema, phpBB schema database is quite big to show here, so it can be accessed in the website[17]. This database has a different characteristic from the previous ones, since it has a quite complete data dictionary[18] that gives good comments about each table and their attributes. For this reason, we decided to also analyze if the data dictionary could replace our proposed annotations, since it somehow gives information about the context. We used the ontologies FOAF and SIOC to get the recommendations. An example of usefulness of the annotation is the term *dso:forum_description*, which is related to the description of a forum discussion. With annotations, this term received a recommendation for the term *sioc:content*,, while without annotations it received no recommendations. This term content in the SIOC ontology describes a post in some online discussion, so it is quite related to the context of the database.

To explore that point, we compared precision and recall among: (i) recommendations resulted by the SbS without any annotations or dictionary data ($T_{B,}$ RA); (ii) recommendations resulted by the SbS with dictionary data ($T_{Bd,}$ RA); and (iii) recommendations resulted by the SbS with dictionary data and annotations . If precision and recall could not differentiate between (ii) and (iii), it would mean that the annotations do not make a difference when the database has a

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

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

dictionary ($T_{Bda,}$ RA). Analyzing the results, we may conclude that the data dictionary T_{Bd} plays an important role, if compared to the T_B, but the annotated T_{Bda}, together the dictionary data, still has the best numbers for both precision and recall.

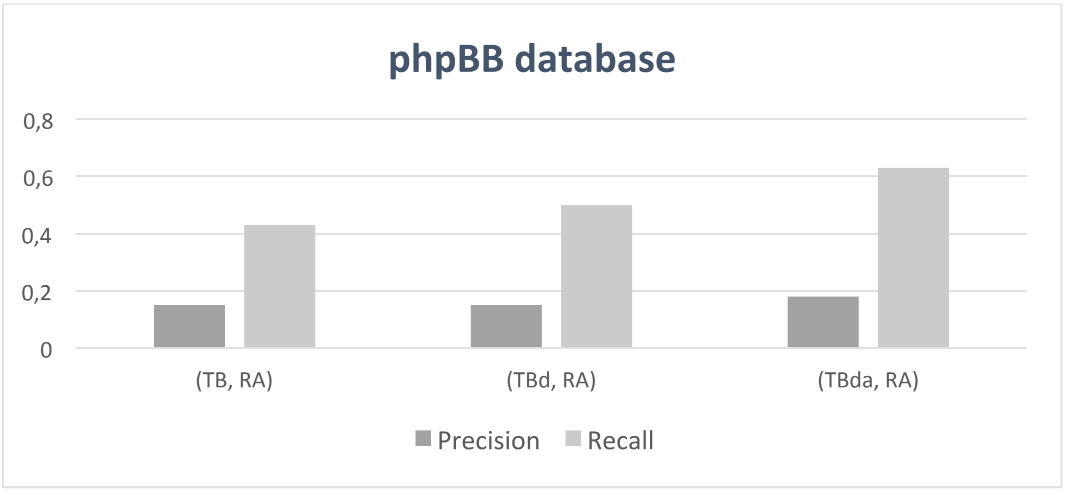


Figure 4.6- phpBB database results.

Table 4.5: Precision and Recall values for phpBB database

	($T_{B,}$ RA)	($T_{Bd,}$ RA)	($T_{Bda,}$ RA)
Precision	0,15	0,15	0,18
Recall	0,43	0,5	0,63

4.4 Related Work

Section 3.4 already covered related work about the RDB-to-RDF process.

We just observe in this section that the previous version of StdTrip 2.0 (Salas et al., 2010a; Salas et al., 2010b ; Berardi et al., 21013) used Ontology Matching algorithms that were in the state of the art list of OM participants in the last years in the OAEI challenge (Euzenat and Shvaiko, 2013), such as AROMA (David, 2009), Lily (Wang, 2009), Anchor Aflood (Seddiqui, 2009). However, in StdTrip 2.0, we are adding annotations to a DSO that originally had no comments or definitions and comparing it with ontologies rich of definitions and comments.

4.5 Conclusion

In this chapter, we introduced StdTrip 2.0 as a process that automatically defines a vocabulary from a relational database (Database Schema Ontology) and that supports the generation of design rationale represented by annotations, recorded when the database is 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 used to recommend 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.

The new step of Annotation in the StdTrip 2.0 was implemented as an extra module, written in Python, with functions that help manipulate graphs and libraries for text manipulation (nltk). This module receives as input an intermediate output of the StdTrip2.0 (the DSO ontology and the previous design rationale DR1) and delivers an output (DSO annotated and the design rational DR2) to continue the execution of StdTrip 2.0 until the last step.

5
Design Rationale for an RDB-to-RDF customized Mapping Process

5.1
Introduction

Two main approaches are widely used for mapping relational databases into RDF: the direct mapping approach, where the database schema is directly mapped to ontology elements (Sequeda et. al., 2011), and the customized mapping approach, where the schema of the RDF may differ significantly from the original database schema. As an alternative to proprietary mapping languages, the W3C RDB2RDF Working group proposed R2RML as a standard mapping language (Das, et. al., 2012).

R2RML mappings allow the designer to express customized transformations over the original data, which may affect how the published data is consumed. Hence, it would help the user understanding such transformations if a transparency layer were added to the publishing process. Adding transparency would also help the data publisher to trace all the RDB-to-RDF process for maintenance purpose.

This chapter therefore proposes a strategy, called R^{2}BA, to achieve transparency. R^{2}BA couples design rationale with a semi-automatic method to define R2RML mappings, called RBA (R2RML by assertion) (Vidal et. al., 2014). RBA adopts correspondence assertions as a convenient way to manually specify R2RML mappings and incorporates an automatic procedure to generate SQL Views and R2RML mappings from the correspondence assertions. Intuitively, R2BA rationalizes the R2RML mappings, in the sense that it makes explicit all the RBA process.

This paper has two major contributions. First, it extends the RBA method to include design rational, creating what we called the R2BA method. By capturing the design rationale, R2BA helps publishers to document the design process and final users to consume the published data by giving them evidences to answer the following questions: (1) Did the original relational data suffer changes, when published as RDF triples, that could impact its quality?; (2) Is the translation from the original relational data to RDF triples correct?; (3) Is the chosen ontology the most appropriate to represent the original relational database?; (4) Did the original relational data lose some relevant information when published as RDF triples?

Second, the chapter proposes to use the design rationale captured to enrich the vocabulary that will represent the original data as RDF. This enrichment can be used by ontology matching algorithms to find potential links to other existing vocabularies, thereby promoting interoperability.

This chapter is organized as follows. Section 5.2 outlines the semi-automatic method to define R2RML mappings and the design rationale model; it also introduces a motivating example. Sections 5.3 to 5.6 detail the R^{2}BA approach. Section 5.7 shows the final DR captured in the example. Section 5.8 discusses some related work and finally section 5.9 contains the conclusion.

5.2 Overview of the method

This section provides a brief overview of the semi-automatic method to define R2RML mappings and its extension to capture the design rationale. Sections 5.3 to 5.6 cover the details and give examples.

5.2.1 A Running Example

To illustrate the method, we will use the following example. Figure 5.1 depicts the relational schema *ISWC_REL*. Each table has a primary key, whose name ends with '*ID*'. *Persons* and *Papers* represent the main concepts. *Rel_Person_Paper* represents a N:M relationship between *Persons* and *Papers*. The labels of the arcs, such as *FK_Publications*, are the names of the foreign keys. Figure 5.2 depicts the ontology *CONF_OWL*, which reuses terms from *FOAF* (Friend of a Friend), *SKOS* (Knowledge Organization System), *VCARD* and *DC* (Dublin Core). The prefix *'conf'* is used for the new terms defined in the *CONF_OWL* ontology.

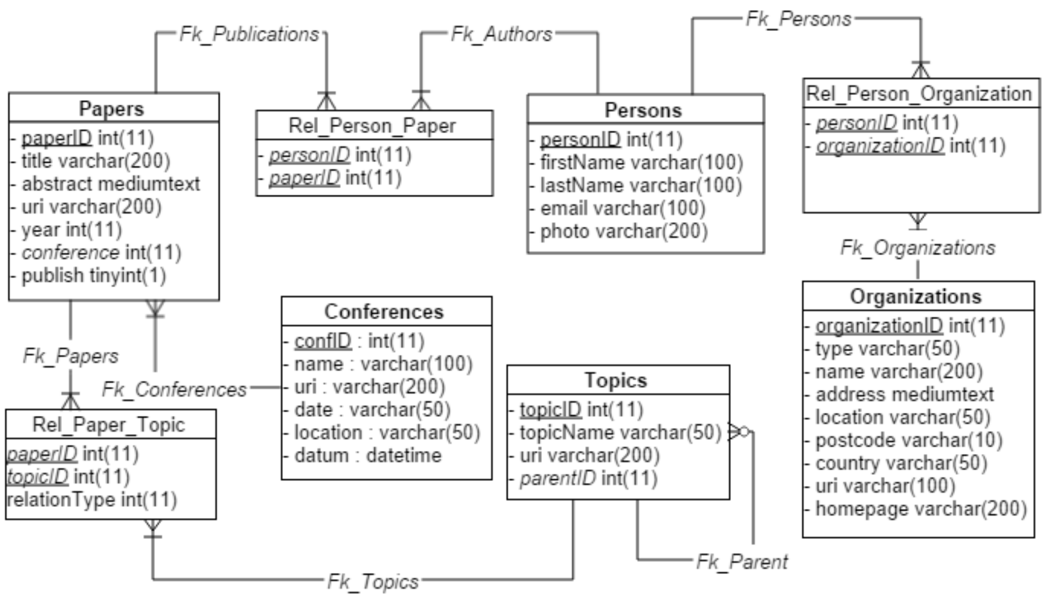


Figure 5.1 - The ISWC_REL database schema.

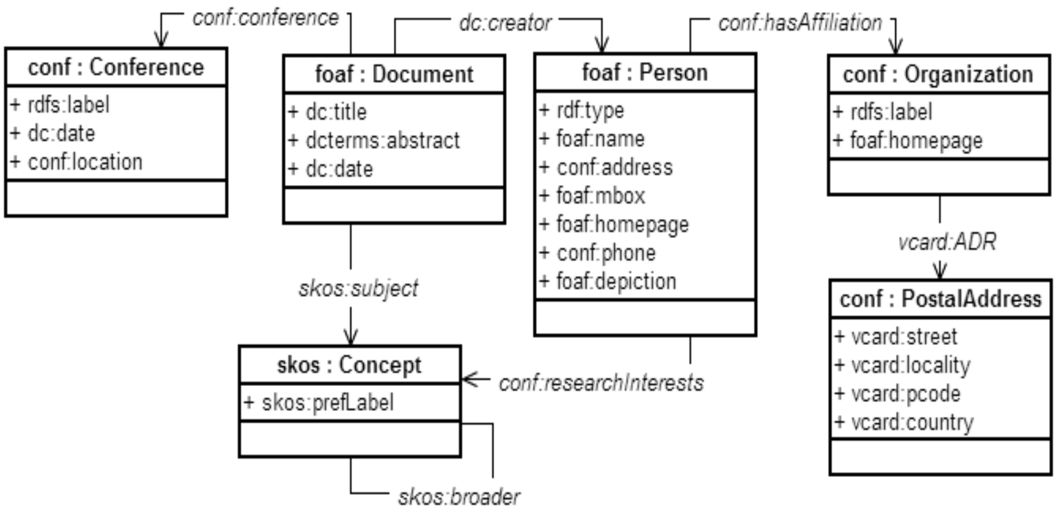


Figure 5.2 - The CONF_OWL ontology.

5.2.2
The R2RML Mapping by Assertion Method (RBA)

The RBA method proposes to generate customized R2RML mappings based on correspondence assertions (Vidal et al., 2014 ; Neto et al., 2013). The inputs of the method are a relational database schema that will be published as RDF and a set of domain ontologies. The output is an *exported ontology*, which represents part of the relational data in RDF, the R2RML mappings and a set of SQL view definitions.

The first step of RBA is manual and relies on the user to define mappings between the relational database and the domain ontologies using correspondence assertions (CAs), which are much simpler to understand than R2RML and yet suffice to capture most of the subtleties of mapping relational schemas into RDF schema (Vidal et al., 2014). A tool has also been developed to the designer in this step (Vidal et al., 2014 ; Vidal et al., 2005). Table 5.1 shows the abstract syntax and examples of the three types of CAs.

Table 5.1 – DR interpretation for Class, Object Property and Data Type Property Correspondence Assertions.

Type	Definition	DR interpretation	Examples of Correspondences Assertions
CCA	Ψ: $C \equiv R[A_1,...,A_n]\sigma$ *(σ is optional)*	*Class type Table[URI] FILTER (FILTER is omitted if so is σ)*	$\Psi_{1:}$*foaf:Person ≡ Persons[personID]* $\Psi_{2:}$ *skos:Concept ≡ Topics[topicID]* $\Psi_{3:}$*foaf : Document ≡ papers [PaperID], FILTER [papers.Year > 2002]*
OCA	Ψ: $P \equiv R / \varphi$ *(φ is optional)*	*ObjP mapped Table_Domain / Ref_Att_URI_Range* *(Ref_Att_URI_Range is optional)*	Ψ_4: *conf:researchInterests ≡ Persons / [Fk_Authors, Fk_Publications, Fk_Papers, Fk_Topics]*
DCA	Ψ: $P \equiv R / \varphi$ / $\{A_1,...,A_m\}$ *(φ is optional)*	*DataP mapped Table_Domain / Ref_Att_Range / {Att_Lit_Range$_n$}* *(Ref_Att_Range is optional)*	$\Psi_{5:}$*foaf:name ≡ Persons / {firstName, lastName}*

Class correspondence assertions (CCAs) (as in line 1 of Table 5.1) map tables into classes. Their abstract syntax is

$$\Psi: C \equiv R[A_1,...,A_n]\sigma$$

where Ψ is the name of the CCA, C is a class of a domain ontology, $R[A_1,...,A_n]$ is a relation schema with the attributes $A_1,...,A_n$ (attributes of the primary key of R) and σ is an optional selection over R.

Object property correspondence assertions (OCAs) (as in line 2 of Table 5.1) map tables into object properties. Their abstract syntax is

$$\Psi: P \equiv R / \varphi$$

where Ψ is the name of the OCA, P is an object property of a domain ontology, R is a relation name of the relational database schema and φ is an optional path from R. A *path* is a set of foreign keys that connect relations in relational databases.

Datatype correspondence assertions (DCAs) (as in line 3 of Table 5.1) map tables into datatype properties. Their abstract syntax is

$$\Psi\text{: } P \equiv R \,/\, \varphi \,/\, \{A_1,\ldots,A_m\}$$

where Ψ is the name of the DCA, P is a datatype property of a domain ontology, R is a relation name of the relational database schema, φ is an optional path from R and $A_1,\ldots,A_n$ are attributes of R.

The vocabulary of the *exported ontology* is simply the set of classes and properties of the domain ontologies used in the correspondence assertions. Figure 5.3 shows the ISWC_RDF exported ontology generated from the CAs that map the ISWC_REL database schema of Figure 1 to the CONF_OWL ontology of Figure 5.2.

The second step is automatic and compiles the correspondence assertions into R2RML mappings and SQL view definitions, as depicted in Figure 5.4.

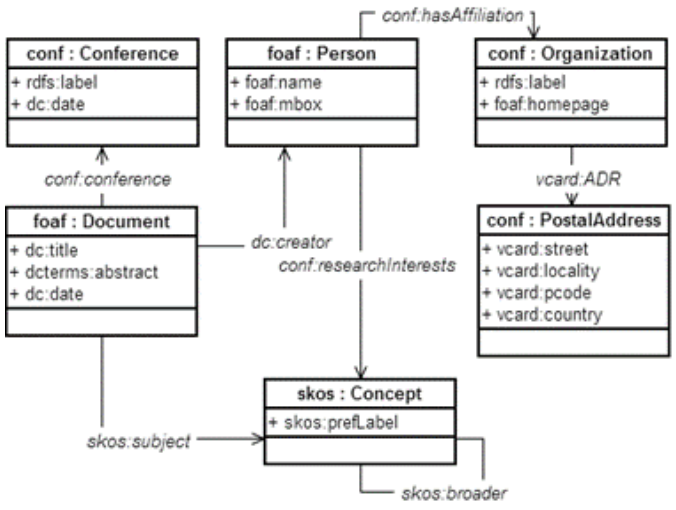


Figure 5.3 - ISWC_RDF exported ontology schema.

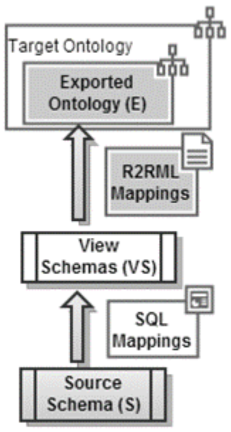


Figure 5.4 - Output of the design process.

5.2.3 The Rationalizing R2RML Mapping by Assertion Method (R^2BA)

R^2BA is an extension of the RBA method to include design rationale (Berardi et. al., 2015) . It uses the correspondence assertions to trace and record how the classes and properties are created in RDF. R2BA extends RBA to capture the design rationale, which is then used to enrich the exported ontology and to establish a link between similar classes and properties.

R^2BA consists of 6 steps, divided into 3 groups according to their goals. Each one of these groups is discussed in detail in Sections 5.3 to 5.6.

The first group comprehends two steps: “Step 1: Creation of the correspondence assertions” and “Step 2: Creation of an exported ontology to represent relational data in RDF”. This group receives as input a relational schema, the data source schema, and several target ontologies of the user’s choice, where each ontology is composed by a vocabulary and set of constraints. As output, it produces an exported ontology and the design rationale DR1 of Step 1 and DR2 of Step 2.

The second group enriches the exported ontology to facilitate interoperability. It also comprehends two steps: “Step 3: Generating annotations” and “Step 4: Generating linking recommendations”. This group receives as input the exported ontology and DR2. As output, it produces an enriched exported ontology and the corresponding design rationale (DR3 for Step 3 and DR4 for Step 4).

The last group generates SQL views according to the enriched exported ontology and the R2RML mappings. It comprehends two steps: "Step 5: Generating SQL views" and "Step 6: Generating R2RML mappings". This group receives as input the enriched exported ontology and DR4. As output, it produces: a set of relational views schemas; a set of R2RML mappings; and the final DR (DR5 for Step 5 and DR6 for Step 6. Figure 5.5 illustrates the complete R^{2}BA process with all new steps (Annotation and Matching) that will be detailed in the next Sections.

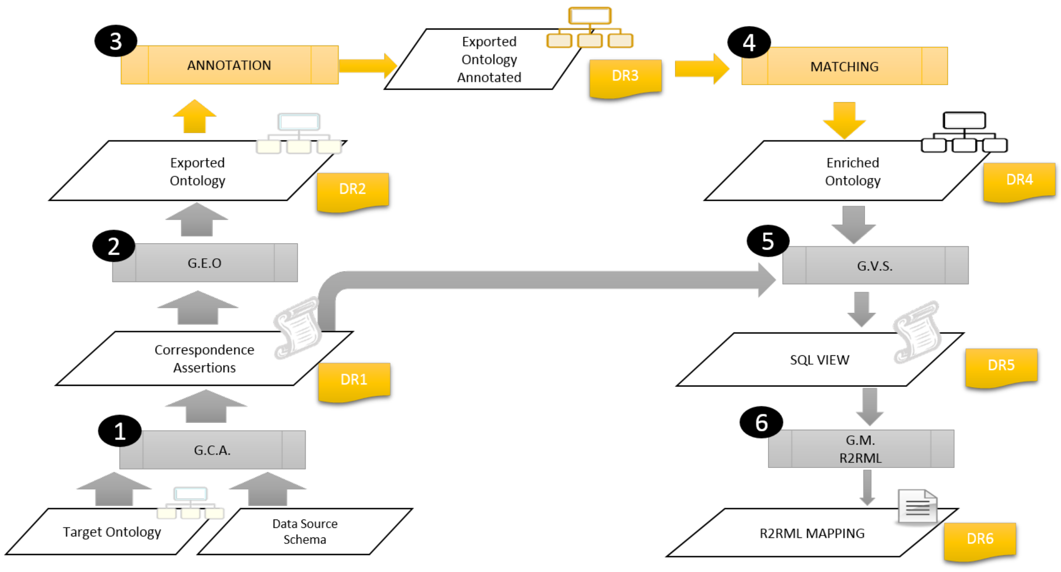


Figure 5.5 - R^{2}BA process

5.3 Group 1 - Creating the mappings and The Exported Ontology

5.3.1 Overview

The first group of steps of R^{2}BA has as goal the creation of the correspondence assertions and an exported ontology to represent relational data in RDF. It receives as input a relational schema, the *data source schema*, and several *target ontologies* of the user's choice, where each ontology is composed by a vocabulary and set of constraints. As output, it produces an *exported ontology* (EO) and the *design rationale* DR1 of step 1 and DR2 of step 2.

Step 1 – *Generating Correspondence Assertions.*

This step consists in a manual specification of a set of correspondence assertions (CA) between elements of the database relational schema and terms from vocabularies of user's choice. The design rationale captured in this step, referred to as DR1, records the original format of the data source schema elements and tracks which elements are not mapped.

To visualize the DR1 captured in this step, consider the *ISWC_REL* schema depicted in Figure 5.1. Observe the table *Persons* and its attributes *firstName*, *lastName*, *email* and *photo*. Their original formats are represented at DR1 in Figure 5.6 through the rectangular nodes with the same names.

The questions associated with DR1 are *Element* and *Map*. The *Element* question seeks to explicit the original format of the element, so it may be answered with relational database elements, such as *Table*, *Att*, *KeyAtt* or *FKAtt* for table, attribute, primary key attribute and foreign key attribute, respectively. For example the rectangular node *Persons* has the answer *Table* and the rectangular nodes *firstName* and *lastName* have *Att*. To represent a relationship between two tables, DR1 answers the question *Element* with *FKAtt* and creates a question *Ref* to be answered with the names of the table and attribute that is the reference of the *FKAtt*.

To record elements that are not mapped, R^2BA has a mechanism to compare the elements present in the CAs and the elements present in the original database schema. For example, attribute *photo* of table *Persons* has the question *Map* answered with *NOT*. At DR1, this is the only case where the question *Map* is asked.

Step 2 – *Generating the Exported Ontology*.

This step consists in using the set of correspondence assertions to automatically generate the exported ontology (EO). According to the RBA method, the list of CAs is consumed to generate the exported ontology, in the following order: all CCAs are first mapped to the EO; then all OCAs; and finally all DCAs. The design rationale captured in this step, referred to as DR2, records information parallel to each mapping created.

Together, DR1 and DR2 allow answering the following questions: (i) What is the original form of the data in the data source schema? ; (ii) Are all elements in the data source schema mapped? If not, which were and which were not mapped?; (iii) For those elements that were mapped, how they were mapped as ontology elements?

In order to trace how the elements were mapped and record this information at DR2, we developed a DR interpretation for each kind of CA, as shown in Figure 5.6. Each interpretation expression is used to answer the questions asked during this step. The next sections detail and exemplify how to generate the DR2 for CCA, OCA and DCA.

5.3.2 Design Rationale for Class Correspondence Assertions (CCA)

Consider a class correspondence assertion of the form Ψ: C ≡ $R[A_1,...,A_n]$, that is, which does not use a filter (the first two examples in line 1 of Table 5.1). The DR interpretation for such CCAs is formalized as the expression *"Class type Table[URI]"*, read as *"a class that is mapped as type (rdf:type) from a table represented by R using attributes $A_1,...,A_n$ to build the URIs"*. For example, the assertion in the first line of Table 5.1 Ψ_1: *foaf:Person ≡ Persons[personID]* maps table *Persons* (see Figure 5.1) to class *foaf:Person* (see Figure 5.2) using attribute *personID* to build the URI of the class. The DR for this assertion therefore has *Class* as *foaf:Person, Table* as *Person*s and *URI* as the attribute *personID*. Figure 5.6 shows the representation of this example.

DR2 increments DR1 by adding a new circle node for *Class*, called *foaf:Person*, and connecting it to the corresponding node of *Table*, that is, the table *Persons*. The questions for the new node *Class* are *Element*, *URI* and *Map*. To answer the question *Element*, the *Class* component of the DR interpretation expression is used. To answer the *URI* question, the instantiation of the component *URI* of the DR interpretation expression is used (*personID_*in the example). Finally, the question *Map* is related to the connection arrow between the correspondent nodes *Class* and *Table*. To answer it, the component *type* of the DR interpretation expression is used.

Consider now a class correspondence assertion of the form Ψ: C ≡ $R[A_1,...,A_n]\sigma$, that is, which uses a filter σ (the third example in line 1 of Table 5.1). Such assertions are represented in DR2 with the help of the question

FILTER, answered by recording the filter used in the correspondence assertion (not shown in Figure 5.6 for simplicity).

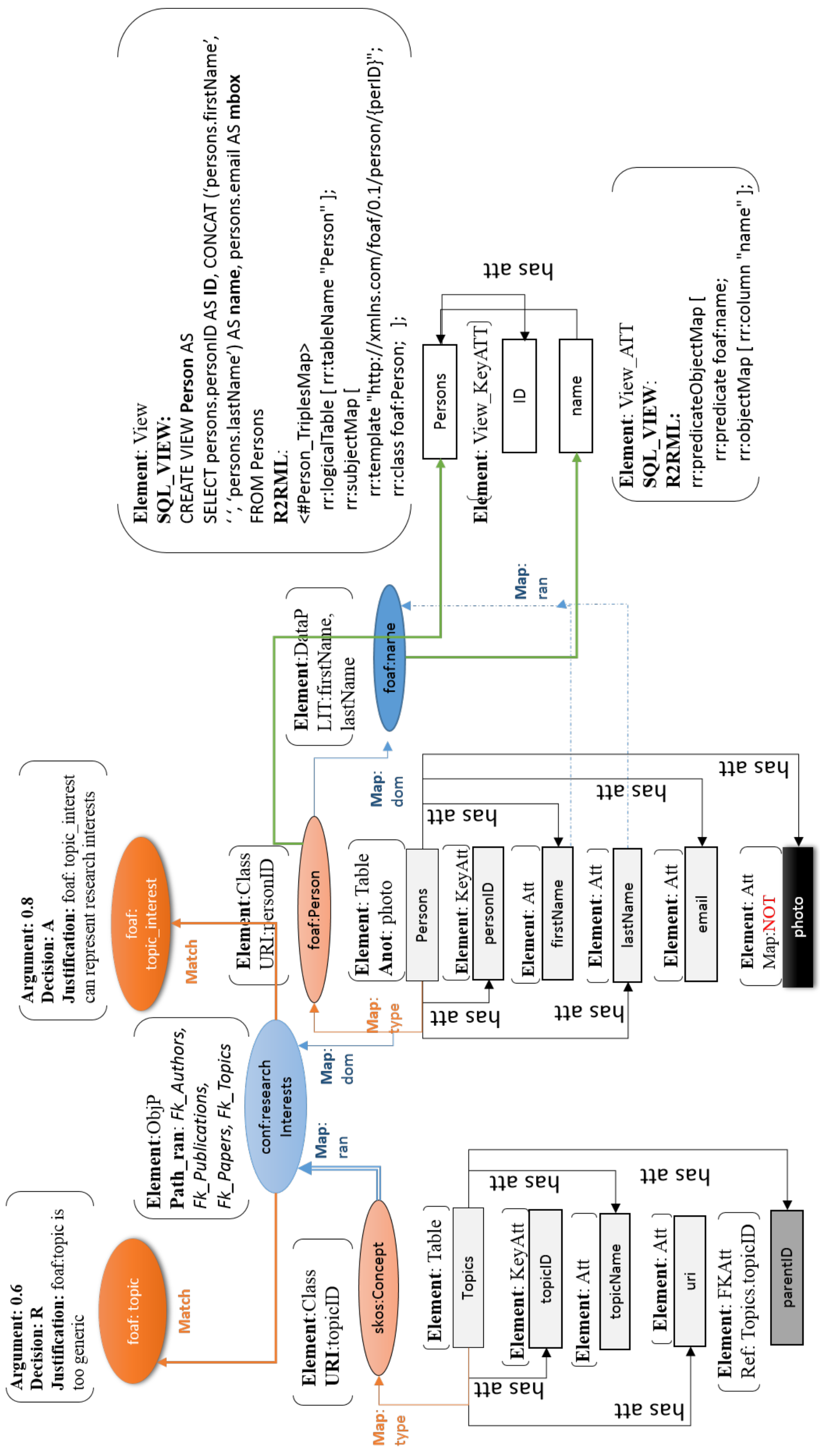
5.3.3 Design Rationale for Object Property Correspondence Assertions (OCA)

After recording the design rationale model for the class correspondence assertions, DR2 represents the design rationale for object property correspondence assertions (OCAs).

Consider an OCA of the form Ψ*: P ≡ R /* φ (as in line 2 of Table 5.1). The DR interpretation of an OCA is formalized as the expression *"ObjP mapped Table_Domain / Ref_Att_URI_Range",* read as "an object property *P* is mapped using a *Table R* as its *domain* and its *range* is represented by an URI composed by a key attribute of a table, that is found by following the path φ in *Ref_Att_URI_Range*".

For example, the OCA in line 2 of Table 5.1 OCA: *conf:researchInterests ≡ Persons / [Fk_Authors, Fk_Publications, Fk_Papers, Fk_Topics]* maps the object property *conf:researchInterests* (in Figure 5.2) using the table *Persons* (in Figure 5.1) to represent the domain and the concept *skos:Concept* is found by following the path *[Fk_Authors, Fk_Publications, Fk_Papers, Fk_Topics]* (Figure 5.1 and Figure 5.2). The DR for this assertion therefore has *ObjP* as *conf:researchInterests*, *Table_Domain* as *Persons* and *Ref_Att_URI_Range* as *Fk_Authors, Fk_Publications, Fk_Papers, Fk_Topics*. Figure 5.6 shows the representation of this example.

DR2 increments DR1 by adding a new circle node for *ObjP* called *conf:researchInterests*. The questions for the new node *ObjP* are *Element* and *Map*. To answer the question *Element*, the *ObjP* component of the DR interpretation expression is used. As *ObjP* is an object property, the domain and range are URIs. When the range is an URI composed by following a path in the correspondence assertion, DR2 records a question *Path* and answers it with the path φ provided by the correspondence assertions.



Element: View
SQL_VIEW:

```
CREATE VIEW Person AS
SELECT persons.personID AS ID, CONCAT ('persons.firstName',
' ', 'persons.lastName') AS name, persons.email AS mbox
FROM Persons
```

R2RML:

```
<#Person_TriplesMap>
   rr:logicalTable [ rr:tableName "Person" ];
   rr:subjectMap [
      rr:template "http://xmlns.com/foaf/0.1/person/{perID}";
      rr:class foaf:Person;   ];
```

Element: View_ATT
SQL_VIEW:
R2RML:

```
rr:predicateObjectMap [
   rr:predicate foaf:name;
   rr:objectMap [ rr:column "name" ];
```

Figure 5.6 – DR graph for the motivating example.

Figure 5.6 shows this example for the range of the object property *conf:researchInterest*. The question *URI* is answered by the ID attribute used to compose the URI, which may be found by following a path in the correspondence assertion. To identify the arrows that are related to domain and range mappings, the DR model uses the question *Map* and the answers *dom* and *ran* for domain and range, respectively. Thus, the connection between the domain and range should be directed from the ontology element corresponding to the URIs. It is important to highlight that the OCA uses a table to represent the domain of the *ObjP*, but the true domain is the ontology element corresponding to this table (and likewise for the range representation).

For example, in the OCA of Table 5.1, the *ObjP* node is connected to the node labeled *foaf:Person*, which is the ontology element mapped from the table *Person.* The same happens with the connection between the *ObjP* node and its range. In the DR2 graph, a double arrow indicates a range defined by a path, while a single arrow indicates a range defined by a single attribute.

5.3.4 Design Rationale for Datatype Property Correspondence Assertions (DCA)

After having recorded the DR2 for CCAs and OCAs, finally the DR for DCAs is recorded. Consider first a DCA of the form $\Psi: P \equiv R / \{A_1,...,A_m\}$ (as in line 3 of Table 5.1), that is, which does not use a path. The DR interpretation of this correspondence assertion is formalized as the expression *"DataP mapped Table_Domain/ {Att_Lit_Range$_n$}"*, read as *"a datatype property P is mapped using table R as its domain and its range is a set of values generated using attributes $\{A_1,...,A_m\}$"*. Using this interpretation for the example in Table 1 *DCA:foaf:name ≡ Persons / {firstName, lastName}*, *DataP* is *foaf:name*, *Table_Domain* is *Persons* and *{Att_Lit_Range$_n$}* is *firstName,lastName*. This DCA maps the datatype property *foaf:name* (Figure 5.2) using the table *Persons* (Figure 5.1) as domain and the values of the attributes *firstName* and *lastName* (Figure 5.1) as range.

The representation of this example is shown in Figure 5.6 and is similar to the DR of an object property. The most important difference is that, in datatype properties, the range is not a class, but a XML data type defining a set of literals.

So, the literals are generated from attribute values, as indicated in the DCA. Thus, the connection in the DR graph is directed from the attributes. As this example uses a composition of two attributes, both are connected to the node represented by dashed lines. When the DCA specifies only one attribute, a single line is used.

For a DCA of the form Ψ*: P* ≡ *R / φ / {*A_1*,...,*A_m*}*, which uses a path, the DR follows likewise. Similarly to OCAs that uses a path to find ranges, a question *Path_ran* is created in the node associated with the data type property and answered with the path in the correspondence assertion. In this case, a double arrow in the DR graph indicates a range found by following a path in the correspondence assertion.

5.4 Group 2- Enriching the Exported Ontology

This group of steps enriches the exported ontology to facilitate interoperability. It receives as input the exported ontology (EO) and DR2. As output, it produces an enriched exported ontology (eEO) and the corresponding design rationale (DR3 for step 3 and DR4 for step 4).

Step 3 – *Generating annotations.*
This step consists in generating annotations for those cases where the relational database is composed of a private parte (that is not published as RDF) and a public part (that is published). This is a new step in the RBA approach and aims at adding information about the private relational schema in the exported ontology.

DR3 increments the nodes in DR2 with annotations, according to a *neighboring mapping*, defined as: for each mapped element, look for a neighbor in the DR graph that has the question *Map* answered with *NOT*, which means it was not mapped to the exported ontology. If one such node exists, a new question is created *Anot* at the node that found an element that was not mapped. The question *Anot* is answered with the name of the unmapped node. This information is added to the exported ontology as a datatype property *rdfs:comment* whose value is a literal composed of the names of the unmapped nodes.

The search for unmapped neighbors follows an annotation strategy up to a certain depth in the DR graph. As an initial strategy and based on empirical observations, we considered a maximum of two levels. More specifically, we noticed that, for automatic annotations, including more than two levels becomes superfluous, as the additional levels are more likely to be out of context (Berardi et al., 2014). For instance, observe in Figure 5.6 that the attribute *photo* is marked with the question *Map:NOT* and is therefore annotated with the mapped node *Persons*.

The benefits of using the DR graph to generate annotations, instead of directly using the relational database, are: (i) Since the DR graph is created for provenance purposes, it can be accessed without having to create a new graph based on the relational schema to know what has to be annotated; (ii) Since the DR graph is created in all steps of R^{2}BA, it can be consumed whenever it is needed, without having to rerun the steps from the very beginning.

Step 4 – *Generating linking recommendations.*

This step consists in executing ontology matching algorithms using as input the eEO. The annotation helps ontology matching (OM) techniques to keep the context of the elements in the exported ontology. When only part of the schema is published, this part can lose information that can be useful for OM.

In order to promote interoperability, we seek to establish a link between similar classes or properties using *rdfs:equivalentClass* or *rdfs: equivalentProperty* properties. In this step the user interaction plays an essential role because the OM process gives a list of recommendations for the terms of the exported ontology. Ideally, the user should know the database domain so that he or she can accept or reject the recommendations. These links of the annotated exported ontology are part of the enriched exported ontology (eEO).

DR4 increments DR3 with two new nodes for the two largest recommendation similarity values. Then, the questions involved at this step are: *Argument*, answered with the similarity value output by the OM algorithms; *Decision*, with *A* or *R*, which represents the domain expert decision for accepting or rejecting the recommendation, respectively; and *Justification*, with the justification provided by the domain expert about her or his decision.

For instance, in Figure 5., the object property *conf:researchInterests* receives two recommendations for terms that seem to be equivalent to *foaf:topic* and *foaf:topic_interest*. The corresponding nodes of the recommendations are connected to the node of the term *conf:researchInterests*. This connection is labeled with *Match* to explicit that they are recommendations from the OM techniques.

Together, DR3 and DR4 allow answering the following questions: (i) Which elements received annotations and what are the annotations?"; (ii) Which recommendations each term received from the OM techniques? (iii) Which recommendations were accepted and why? (iv) Which recommendations were rejected and why?.

5.5 Group 3 - Generating SQL Views and R2RML Mappings

The last group of steps generates SQL views according to the enriched exported ontology (eEO) and the R2RML mappings. It receives as input the eEO and DR4. As output, it produces: a set of relational views schemas; a set of R2RML mappings; and the final DR (DR5 for step 5 and DR6 for step 6).

Step 5 – *Generating SQL views.*
This step consists in automatically generating a set of relational view schemas that is a direct transformation of the enriched exported ontology. In (Neto et al., 2013) an algorithm is presented to automatically generate the view schemas based on the exported ontology and the CAs.

Step 6 – *Generating R2RML mappings.*
This step consists in automatically generating R2RML mappings from the views to the enriched exported ontology, which is one-to-one. The DR5 and the final DR6 are captured in parallel and they allow to answer the following questions: (i) "Which SQL view is associated with each element of the enriched exported ontology?"; (ii) "Which R2RML mapping refers to each element of the enriched exported ontology?". The final DR6 makes it possible to trace all the

transformations that each element in the original relational schema suffered during the mapping process.

Table 5.2: Step 5 and 6 for classes.

```
1. Operation 1: For each class $C$ in $V_E$ where $K_1,...,K_n$ are the datatype properties of the key of $C$ do
2. Create a relational view also named $C$;
3. Create a new node that is an Element:View in DR4
4. Connect the Element View to the node correspondent to the class C
5. 1.2 Create $K_1, ... , K_n$ , the attributes of the primary key of view $C$;
6. Create a new node that is an Element:View_ Key_Att in DR 4
7. Connect the new Key_Att Element to the View Element correspondent and label it with “has_att”
8. 1.3 Create the subject map referring to view C using template T1.
9. In the Element View node, create a question “R2RML” and answer it with the triple map created in substep
```

Table 5.3: Step 5 and 6 for object properties.

```
1.     Operation 3: For each object property $P$ in $V_E$ do
2.     Let $D$ and $R$ be the views that match to the domain and range of $P$, respectively, let $K_{D1},...,K_{Dn}$ be the
       attributes of the primary key of $D$ and let $K_{R1},...,K_{Rn}$ be the attributes of the primary key of $R$; // views D
       and R were created in Step 1
3.     Case 3.2: $P$ has cardinality greater than 1.
3.2.1. Create relational view $D\_P$;
4.     Create a new node Element View
5.     Connect the new node Element View to the Element ObjP node correspondent
6.     3.2.2. Create attributes $K_{D1},...,K_{Dn}$ in $D\_P$ whose types are defined as in D;
7.     Create a new node Element View_FK_PK
8.     Connect the new node Element View_FK_PK to the Element View correspondent and label the connection
       with “has_att”
9.     3.2.3. Create foreign key $FK\_D\_P\_D(D\_P:\{K_{D1},...,K_{Dn}\}, D:\{K_{D1},...,K_{Dn}\});$
10.    Create the question “Ref” in the node Element View_FK_PK and answer it with $D:\{K_{D1},...,K_{Dn}\}$
11.    3.2.4. Create attributes $K_{R1},..., K_{Rn}$ in $D\_P$ whose types are defined as in $R$;
12.    Create a new node Element View_FK_PK
13.    Connect the new node Element View_FK_PK to the Element View node correspondent
14.    3.2.5. Create foreign key $FK\_D\_P\_R(D\_P:\{K_{R1},..., K_{Rn}\}, R:\{K_{R1},..., K_{Rn}\})$;
15.    Create the question Ref and answer it with D:{KD1,...,KDn })
16.    3.2.6. Create the subject map referring to view D_P and predicate object map for P using template T5.
17.    Create the question “R2RML” in the Element View node and answer it with the result of the step 3. 2.6
```

The algorithm that automatically generates the view schemas and the R2RML mappings has 3 main steps. Each one of these steps implements the SQL view and the R2RML mappings generation, respecting this order: classes first, then datatype properties and, finally, object properties. Tables 5.2 and 5.3 show the steps for classes and object properties mappings and DR composition respectively. We added new steps to the original algorithms presented in Tables 5.2 and 5.3 to record the DR, such as the lines 3, 4, 6, 7 and 9 at Table 5.2 and the lines 4, 5, 7, 8, 10, 12,13, 15 and 17 at Table 5.3.

The algorithm receives as input the enriched exported ontology and a set of templates for creating SQL views and R2RML mappings. The complete list of these templates can be found in (Vidal et al., 2014).

5.6 The DR resulted of Group 3

Let V_{eEO} be the vocabulary of the enriched exported ontology and $K_1,\ldots,K_n$ be the attributes of a view *C*. Table 5.2 shows the first part of the algorithm related to the class mapping. As an example of Step 1, consider the class *foaf:Person* of the exported ontology in Figure 5.3. Step 1 creates a view for this class called *Persons*, then DR5 generates a new rectangular node, also called *Person*, as shown in Figure 5..

At DR5, the questions involved are *Element* and *SQL_VIEW*. As DR5 is coupled with the view creation step, the algorithm is able to answer the question *Element* with *View*.

After having created the view, the algorithm also creates the ID attribute; DR4 then generates a new rectangular node. In this case, the question *Element* is answered with *View_KeyATT*. The question *SQL_VIEW* is answered as the view is in fact created. The new node *Persons* in DR5 represents the view *Persons*, which corresponds to the class *foaf:Person* in DR2, that consequently represents the Table *Persons* in DR1.

Following Step 1 of the algorithm, the last sub step is to generate the R2RML mapping. For that, the algorithm uses a list of templates according to each step. Space limitations do not permit to explore each template used, so we cover only one as an example.

For the class R2RML mapping, template T1 is used:

```
T1: <#C_TriplesMap>
  rr:logicalTable [rr:tableName "C"];
  rr:subjectMap [
  rr:template "namespaceOfC/{K1}/{K2}/…/{Kn}/…/";
  rr:class C; ];
```

DR5 is incremented with the template information recording, so that DR6 is built. The last sub step of Operation 1 is to finish DR6 with the question *R2RML* in the *Element:View* node and answering it with the instantiation of the template T1 used.

```
<#Person_TriplesMap>
rr:logicalTable [rr:tableName "Person"];
rr:subjectMap [
rr:template "http://xmlns.com/foaf/0.1/person/{personID}";
  rr:class foaf:Person; ];
```

Finally, Table 5.3 shows the part of the algorithm related to the object property mappings. Similarly to the datatype property mappings, this part of the algorithm implements different strategies for object properties with cardinality equal to 1 (Case 3.1) or greater than 1 (Case 3.2). In our example, we explore Case 3.2 using as example the object property *conf:researchInterests* of the exported ontology in Figure 5.3.

In Step 3, a new view is created, called *Person_ResearchInterests*, with two ID attributes *ID_Person* and *ID_Concept*. Both are also foreign keys to construct the domain and range of the object property. The DR 5 related with this step is the creation of a new node, called *Person_ResearchInterest*, that is a view element, with two new nodes, *ID_Person* and *ID_Concept*, which are primary keys and foreign keys, represented as *View_FK_PK*, to answer the question *Element*.

Following Step 3 (Case 3.2), the next sub step is to generate the R2RML mapping. In this case, the triple map for object property mapping is created using template T5:

```
T5: <#D_P_TriplesMap>
  rr:logicalTable [ rr:tableName "D_P " ];
  rr:subjectMap [
    rr:template "namespaceOfD/{K_D1}/{K_D2}/... /{K_Dn}/";
    rr:class D;   ];
  rr:predicateObjectMap [
  rr:predicate P;
  rr:objectMap [
    rr:parentTriplesMap <R_TriplesMap>;
      rr:joinCondition [
        rr:child "K_R1";
        rr:parent "K_R1";   ];
      …
      rr:joinCondition [
        rr:child "K_Rn";
        rr:parent "K_Rn";   ];   ];   ];
```

The composition of DR6 is created by answering the question *R2RML* at the rectangular node *Person_ResearchInterests* with the help of template T5:

```
T5: <#Person_ResearchInterests_TriplesMap>
  rr:logicalTable [rr:tableName “Person_ResearchInterests”];
  rr:subjectMap [
    rr:template  "http://xmlns.com/foaf/0.1/person/{personID}";
    rr:class foaf:Person;   ];
  rr:predicateObjectMap [
    rr:predicate conf:researchInterests;
    rr:objectMap [
      rr:parentTriplesMap <Concept_TriplesMap>;
      rr:joinCondition [
         rr:child “topicID”;
            rr:parent “topicID”;   ];   ];   ] .
```

5.7 Exploring the example of DR of R^2BA

In this Section, we use the example in Figure 5. to show how the design rationale can be captured during a customized approach. One motivation for capturing the DR of a process that uses R2RML is to help define new mappings. The example in Figure 5. shows that the DR allows to clearly understand that the published data about persons (*foaf:Person*) that has research interests (*foaf:topic_interest*) about some concepts (*skos:Concept*) are the result of a set of views. One of these views is Person that selects the first name, last name and email from Persons of the relational database.

Another benefit is the justification provided by the domain expert that decides which terms better represent the database. In the example of Figure 5., we can see that the term *conf:researchInterest* received two recommendations for reusing terms from existing ontologies: “*foaf:topic*” and “*foaf:topic_interest*”. However, the accepted term (*Decision:A*) was the “*foaf:topic_interest*”, which is less generic that “*foaf:topic*”. However, for the context of the database, the semantic of having interest in a topic, instead of just describing a topic, is better represented with the term “foaf:topic_interest”. This information is available thanks to the recording of the DR; future mappings or recommendations can take into account this information.

5.8 Related Work

The notion of correspondence assertions was introduced in (Vidal et al., 2005) to define mappings between instances of source schema to instances of XML view schema. Then, the RBA tool was developed to simplify the R2RML mapping generation and the publication of relational database by using correspondence assertions (Neto et al., 2013).

The previous chapters described methods to capture design rationale for direct mapping processes (see also (Berardi et al., 2013 ; Berardi et al., 2014)). By contrast, R^2BA is the first method to capture design rationale for a customized mapping process as far as we know. It should be stressed that this method has not been described in (Vidal et al., 2014 ; Pequeno et al., 2014).

5.9 Conclusion

To improve the transparency of customized mappings using R2RML, we proposed to couple design rationale with correspondence assertions. With the help of a motivating example, we discussed how to represent this design rationale and how it can help answer several questions regarding the awareness of the possible transformations that the published data suffered. By consuming the final DR captured, it is possible to observe the transformation of the data from their original format in the database, until their final format as an exported ontology, SQL views and R2RML mappings.

We discussed how to use design rationale for transparency and maintenance purposes. Also, we argued that design rationale may help address interoperability issues by creating an enriched exported ontology. The design rationale captured may help new users use R2RML mappings by observing how the mapping process of the original data was implemented. He or she can learn different situations where R2RML is used in a convenient way.

The incorporation of DR in the RBA approach (R^2BA) was not implemented. However, it could be easily implemented either by adding new methods to the existing tool for the RBA approach (Vidal et al., 2005 and Vidal et al., 2014) to capture DR, or by just capturing some events of the existing approach and developing an additional module to represent the events.

6
Conclusions and Suggestions for Future Work

6.1
Conclusions

In this thesis, we focused on the development of mechanisms for a more conscious publication and consumption of relational databases exposed as RDF data.

We addressed the challenges we highlighted in "Section 1.1 Motivation" that are related to the publication and consumption perspectives:

1. *How to select vocabularies to publish the relational data as RDF, following the Linked Data principles?*

By recording the DR of an RDB-to-RDF process, it is possible to uncover the reasons that led a domain expert to choose a vocabulary previously used to represent a relational database. When a domain expert is mapping a new relational database to RDF, he or she can access previous DRs and analyze the justifications recorded about domain ontologies terms and have insights about their utilization. In Chapters 3 and 5, we showed how the DR is represented and how it can be consumed by a domain expert reading the DR graph. In addition, for the cases where the relational database is partially published, the DR can improve recommendations produced by ontology matching algorithms about reuse of terms from other domain ontologies. In Chapter 4, we demonstrated how the DR graph could be automatically consumed in order to generate annotations to enrich the published data with extra information and then improve the contextualization of the partial relational database.

2. *How to maintain the RDB-to-RDF mapping that was created by another designer in the past?*

The DR recorded during an RDB-to-RDF process allows a domain expert to visualize what is the original relational data format and see how it was mapped. Thus, it is easy to visualize a problem and identify what can be done to make a change in the ontology and foresee the impact of the change. In Chapters 3 and 5, we showed how the DR model is represented of and how the DR information can be accessed by a domain expert. In Chapter 4, we showed that the DR graph can be automatically consumed to improve the quality of vocabulary recommendations for reuse purposes, but it also represents a chance to implement another kind of search over the DR graph for maintenance purposes.

3. *How to use R2RML language to define RDB-to-RDF mappings?*

From the publishing perspective, a domain expert may find it difficult to define customized mappings using the R2RML language, which calls for the development of methods and tools to support the deployment of mappings using R2RML (Vidal et al., 2014). By accessing the DR information, the domain expert would be able to analyze the complete mapping process from the original format until the R2RML mappings. In Chapter 4, we showed how a customized approach that uses R2RML can have the respective DR collected and visualized.

4. *How to re-execute just part of the RDB-to-RDF process to accommodate changes in the relational schema?*

As the DR graph is incrementally recorded, it is possible to re-execute the steps of the RDB-to-RDF process one at-a-time. Indeed, as discussed in Chapter 4, the DR records the last status of the process, so it is easy to proceed from a specific step of the RDB-to-RDF process onwards.

5. *How to know whether a relational database lost information during the mapping process?*

As the DR graph records the original formal of the relational database schema and the final formal as RDF data, it is easy to automatic or manually detect any kind of information loss. In Chapter 4 and 5, we discussed how the DR model captured non-mapped elements of the relational database schema.

6. *How to know whether the original relational database suffered changes during the mapping process that impact its quality?*

A user consuming RDF data can access the DR graph, published together the data, and evaluate the changes the data suffered during the RDB-to-RDF process. Based on that, he or she can decide if the data is adequate to consume or not.

7. *How to know why is the chosen ontology the most appropriate to represent the original relational database?*

Since the decisions and justifications are recorded in the DR graph, a user consuming RDF data can access the DR related to the vocabulary construction and analyze which recommended terms were accepted or rejected. Indeed, by analyzing the elements of the DR graph related to the original format of the relational data, the user may better understand the context of the data and the ontology chosen to represent the data.

Finally, in general, the results in this thesis contributed to increase the level of awareness about the complete process of publishing a relational database as RDF data. In particular, the results in this thesis also contributed to a better understanding of how to use R2RML mappings.

6.2 Suggestions for Future Work

As for future research, we suggest:

- Defining queries over Kuaba+W to automatically consume the DR graph with the objective of answering questions that a consumer might have.
- Providing a more compact visualization of the captured DR, allowing a detailed visualization just when required by the triple set consumer.
- Analyzing the scalability of the DR capture process to avoid overloading the RDF triple set with DR data.
- Improving the quality of vocabulary recommendations by taking advantage of semantic ontology matching techniques; by doing so, we would also achieve a smaller number of recommendations and output better contextualized terms.
- Analyzing the advantages of capturing the DR of the preparation step when the RDB-to-RDF process includes a step that covers the conversion of the relational schema to an entity-relationship schema.
- Implementing a mechanism to automatically consume the decisions and justifications contained in the DR graph and use this information to add a recommendation functionality to the step of vocabulary reuse, in similar domains, of the RDB-to-RDF process.

- Extending the method proposed in Chapter 5 to capture the DR of complex correspondence assertions (Complex CAs) (Pequeno et al., 2014).
- Simplifying the DR model, by making it closer to the syntax of the complex CAs.
- Creating and testing mechanisms of automatic comparison between different models of DR to make operations like Union, for example.
- Incorporating the DR model into other RDB-to-RDF strategies, such as the publication of datacubes. To apply a DR capture, Kuaba+W must be reviewed to evaluate whether the model is able to represent the complexity observed in datacubes. After reviewing the model, it is important to map each step of datacube publication and then identify, in each step, what are the questions to be posed and how each one will be answered in this process.
- Regarding the implementation, the DR capture is not yet fully integrated to the tools of all processes addressed in this thesis. In the Chapter 4 we have implemented the capture of the events associated with the DR. Then, the usage of this information, such as annotation and matching were executed independently of the tools. So, as a future work we can fully incorporate the DR capture in the tools without using individual modules.

7
References

Auer S, Dietzold S, Lehmann J, Hellmann S and Aumueller D (2009) **Triplify: light-weight linked data publication from relational databases**. In: WWW ’09: Proceedings of the 18th international conference on World Wide Web, pp. 621–630. ACM, New York, NY, USA. DOI http://doi.acm.org/10.1145/1526709.1526793

Batini C, Ceri S and Navathe SB. (ed) (1991) *Conceptual database design: an Entity-relationship approach*. Addison-Wesley. 470 p.

Bellahsene, D.N.Z. and Todorov, K. (2013) Opening the Black Box of Ontology Matching. In: ESWC 2013

Berardi, R., Breitman, K., Casanova, M. A., Lopes, G. R., and de Medeiros, A. P. (2013) **StdTrip+K: Design Rationale in the RDB-to-RDF Process**. In: Database and Expert Systems Applications, Prague, Czech Republic

Berardi, R., Schiessl, M., Thimm, M., Casanova, M.A. (2014) **The Role of Design Rationale in the Ontology Matching Step during the Triplification of Relational Databases**. Proc. 25th International Conference on Database and Expert Systems Applications, Munich, Germany (Sept. 1-5, 2014)

Berardi, R., Vidal, V., Casanova, M.A. (2015) **R^{2}BA: Rationalizing R2RML mapping by assertion**. Proc. 17th International Conference on Enterprise Information Systems, Barcelona, Spain (April 27-30, 2015).

Bizer C and Seaborne A (2004) **D2RQ-treating non-RDF databases as virtual RDF graphs**. In Proceedings of the 3rd International Semantic Web Conference (ISWC2004)

Bizer C, Heath T, Berners-Lee T (2009) **Linked Data - The Story So Far**. International Journal on Semantic Web and Information Systems (IJSWIS), Vol. 5, No. 3., pp. 1-22, doi:10.4018/jswis.2009081901

Breitman K, Casanova M A and Truszkowski W (2006) *Semantic Web: Concepts, Technologies and Applications*. Londres: Springer, v. 1. 337

Breslin J, Passant A and Decker S (2009) **The Social Semantic Web**. Springer Publishing Company, Incorporated

Casanova M A and Sá J, (1984) **Mapping uninterpreted schemes into entity-relationship diagrams: two appllications to conceptual schema design**. IBM Journal of Research and Development 28. 82-94. DOI http://dx.doi.org/10.1147/rd.281.0082

Cerbah F (2008) **Learning highly structured semantic repositories from relational databases**. The Semantic Web: Research and Applications pp. 777–781

Cheatham, M. and Hitzler, P. (2013) **String Similarity Metrics for Ontology**. In: ISWC 2013

Cordeiro K, Faria F, Pereira B, Freitas A, Freitas J, Bringuente A, Arantes L, Calhau R (2011) **An approach for managing and semantically enriching the publication of Linked Open Government Data**. In Proceedings of the 3rd Workshop in Applied Computing for Electronic Government (WCGE), SBBD 2011.

Cullot N, Ghawi R and Yétongnon K (2007) **DB2OWL: A Tool for Automatic Database-to-Ontology Mapping**, SEBD pp. 491–494

Das S, Sundara S and Cyganiak R (2012) **Rdb to rdf mapping language**. W3c rdb2rdf working group. Retrieved November 16, 2012, from http://www.w3.org/TR/r2rml/

David, J.: **AROMA results for OAEI** 2009 (2009)

Dividino R, Schenk S, Sizov and Staab S (2009) **Provenance, Trust, Explanations – and all that other Meta Knowledge**. Künstliche Intelligenz. KI 23 (2):24-30

Do HH (2006) **Schema matching and mapping-based data integration**. Retrieved in November 28 2012 from http://lips.informatik.uni-leipzig.de/?q=node/211

Donald S (1983) *The Reflective Practitioner : How Professionals Think in Action*. New York: Basic Books

Erling O and Mikhailov I (2009) **Rdf support in the virtuoso dbms**. Networked Knowledge-Networked Media pp. 7–24

Euzenat J and Shvaiko P (2007) *Ontology matching*. Springer-Verlag. Heidelberg (DE).

Euzenat J, Ferrara A and Hollink L (2009) **Results of the ontology alignment evaluation initative 2009**. In: Proc. 4th of ISWC Workshop on Ontology matching (OM).

Herbert S (1981) *The Sciences of Artificial*, 2nd ed. Cambridge, MA: MIT Press

Heuser C (2004) *Projeto de banco de dados*. Sagra Luzzatto.

Kunz W and Rittel HW (1970) **Issues as Elements of Information Systems. Institute of Urban and Regional Development**. Working Paper 131, Univ of California, Berkeley, CA

Lee J (1997) **Design Rationale Systems: Understanding the Issues**. IEEE Expert Volume 12, No. 13 pp 78-85

Lee J, Lai K (1991) **What's in Design Rationale**. Human-Comput. Interaction, No. 6 (3-4) pp 251-280

Maclean A, Young R, Bellotti V, Moran T (1991) **Questions, Options, and Criteria: Elements of Design Space Analysis**. Human-Comput. Interaction, No. 6 (3-4) pp 201-250

McCall Raymond M (1991) **PHI: A conceptual foundation for design hypermedia**. Design Studies, No.12 (1) pp 30-41

McGuinness D and Harmelen F (2004) **OWL web ontology language – W3C Recommendation**. Retrieved Feb 2013. http://www.w3.org/TR/owl-features/

Medeiros AP, Schwabe D (2008) **Kuaba approach: Integrating formal semantics and design rationale representation to support design reuse**. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, v. 22, p. 399-419

Neto, L., Vidal, V., Casanova, M., Monteiro J. (2013) **R2RML by Assertion: A Semi-Automatic Tool for Generating Customized R2RML Mappings**. ESWC 2013.

Pequeno, V.M., Vidal, V.M.P., Casanova, M.A., Neto, L.F.T., Galhardas, H. (2014) **Specifying Complex Correspondences between Relational Schemas and RDF models for generating customized R2RML mappings**. Proc. 18th International Database Engineering & Applications Symposium, Porto, Portugal (July 7-9, 2014), pp. 96-104

Prud'hommeaux, E, Hausenblas, M (2010) **Use cases and requirements for mapping relationaldatabases to rdf**. Retrieved November, 27, 2012 from www.w3.org/TR/rdb2rdf-ucr/

Salas P, Viterbo J, Breitman K, Casanova M A (2010a) **StdTrip: Promoting the Reuse of Standard Vocabularies in Open Government Data**. Linking Government Data. 1ed. Heidelberg. Springer, v. 1, p. 113-133.

Salas P, Breitman K, Casanova MA, Viterbo J (2010b) **Interoperability by Design Using the StdTrip Tool: An a priori approach**. In: I-SEMANTICS, 2010, Graz, Austria. Proceedings of the 6th International Conference on Semantic Systems - I-SEMANTICS. New York: ACM Press, 2010. v. 1. p. 97-105.

Seddiqui, M.H., Aono, M.: **Anchor-Flood: Results for OAEI 2009** (2009)

Sequeda J, Depena R, Miranker (2009) **Ultrawrap: Using sql views for rdb2rdf**. In Proceedings of International Semantic Web Conference. ISWC 2009.

Sequeda, J., Tirmizi, S., Corcho, O., Miranker, D. (2011) **Survey of directly mapping SQL databases to the Semantic Web (2011)**. Knowledge Engineering Review, 26, pp. 445-486. 32

Shvaiko, P. and Euzenat, J. (2013) **Ontology Matching: State of the Art and Future Challenges**. In: IEEE TKDE, Los Alamitos, CA, USA

Terry W (1996) *Bringing Design to Software*, Reading, MA: Addison-Wesley.

Vidal, V. M., Araujo, V. S., Casanova, M. A. (2005) **Towards Automatic Generation of Rules for Incremental Maintenance of XML Views of Relational Data**. Proc. Web Information Systems Engineering (WISE 2005), pp. 189-202

Vidal, V.M., Casanova, M.A., Neto, L.E., Monteiro, J.M. A. (2014) **Semi-Automatic Approach for Generating Customized R2RML Mappings**. 29th ACM Symposium On Applied Computing, Gyeongju, Korea - March 24 - 28, 2014

Vinod G, Peter P (1989) **Motivating the notion of generic design within information processing theory: the design problem space**. Al Magazine 10, 19-36

Vladimir H, Ernst E (1996) *Design science: Introduction to Needs, Scope and Organization of Engineering Knowledge*, 2nd ed. London:Springer-Verlag London Limited.

Wang, P., Xu, B.: Lily: **Ontology alignment results for OAEI 2009** (2009)