StdTrip: Promoting the Reuse of Standard Vocabularies in Open Government Data

Percy Salas, José Viterbo, Karin Breitman and Marco Antonio Casanova

Abstract Linked Data is the standard generally adopted for publishing Open Government Data. This operation requires that a myriad of public information datasets be converted to a set of RDF triples. A major step in this process is deciding how to represent the database schema concepts in terms of RDF classes and properties. This is done by mapping database concepts to a vocabulary, which will be used as the base for generating the RDF representation. The construction of this vocabulary is extremely important, because it determines how the generated triples interlink the resulting dataset with other existing ones. However, most engines today provide support only to the mechanical process of transforming relational to RDF data. In this chapter, we discuss this process and present the StdTrip framework, a tool that supports the conceptual modeling stages of the production of RDF datasets, promoting the reuse of W3C recommended standard RDF vocabularies or suggesting the reuse of non-standard vocabularies already adopted by other RDF datasets.

1 Introduction

The focus of Open Government Data (OGD) lies on the publication of public data in a way that it can be shared, discovered, accessed and easily manipulated by those desiring such data [2]. The Semantic Web provides a common framework that allows data to be shared and reused across applications, enterprises, and community boundaries. Particularly, for representing open data, W3C recommends the Linked

Percy Salas, Karin Breitman and Marco Antonio Casanova
Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, R. Mq. de S. Vicente, 225, Rio de Janeiro/RJ, 22451-900, Brazil, e-mail: {psalas, karin, casanova}@inf.puc-rio.br

José Viterbo
Instituto de Computação, Universidade Federal Fluminense, R. Passo da Pátria, 156/Bloco E/3^o andar, Niterói/RJ, 24210-240, Brazil, e-mail: jviterbo@id.uff.br

Data standard [10], which is based on the representation of data in the form of sets of RDF triples. This approach requires the conversion of a myriad of public information datasets, stored in relational databases (RDB) and represented by database schemas and their instances, to RDF datasets. A key issue in this process is deciding how to represent database schema concepts in terms of RDF classes and properties. This is done by mapping database concepts to an RDF vocabulary, to be used as the base for generating the RDF triples. The construction of this vocabulary is extremely important, because the more one reuses well known standards, the easier it will be to interlink the result to other existing datasets [13]. This approach greatly improves the ability of third parties to use the information provided by governments in ways not previously available or planned, such as the creation of data mashups, i.e., the merge of data from different data sources, in order to produce comparative views of the combined information [1].

There are applications that provide support to the mechanical process of transforming relational data to RDF triples, such as Triplify [4], D2R Server [11] and OpenLink Virtuoso [26]. However, they offer very little support to users during the conceptual modeling stage. In this chapter, we present the StdTrip process, which aims at guiding the users in the task of converting relational data to RDF, providing support in the stage of creating a conceptual model of the RDF datasets. Based on an *a priori* design approach, StdTrip promotes the reuse of standard — W3C recommended — RDF vocabularies, when possible, suggesting the reuse of vocabularies already adopted by other RDF datasets, otherwise.

The rest of this chapter is organized as follows. In Section 2, we discuss the process of publishing relational databases as RDF triples and tools that support this operation. In Section 3, we discuss the interoperability problems and explain the *a priori* matching approach. In Section 4, we present the StdTrip process to be used in the conceptual modeling stages of the process. Finally, in Section 5, we conclude discussing some limitations of our approach and the challenges to be met in the future.

2 RDB-to-RDF Conversion Tools

The publication of relational databases as RDF is known as the RDB-to-RDF approach [37]. This operation takes as input a Relational Database (schema and data) and produces as output one or more RDF graphs [37]. This process may be divided in two independent tasks: the mapping and the conversion. The mapping is a fundamental step in the RDB-to-RDF process and consists in defining how to represent database schema concepts in terms of RDF classes and properties. This definition — represented in a mapping file using specific languages and formats — is used as the base for the conversion, which consists in the generation of the set of RDF triples containing each instance stored in the database. The consumer of the RDF Graph (virtual or materialized) can access the RDF data in three different ways [37]:

- **Query access.** The agent issues a SPARQL query against an endpoint exposed by the system, receives and processes the results (typically the result is a SPARQL result set in XML or JSON);
- **Entity-level access.** The agent performs an HTTP GET on a URI exposed by the system, and processes the result (typically the result is an RDF graph);
- **Dump access.** The agent performs an HTTP GET on dump of the entire RDF graph, for example in Extract, Transform, and Load (ETL) processes.

A survey on existing RDB-to-RDF approaches [40], points out that researchers and practitioners have provided different mechanisms with which to tackle the RDB-to-RDF conversion process. It is important to note, however, that the current RDB-to-RDF approaches provide different, proprietary, mapping languages for the mapping process. Due to this fact, there are initiatives towards establishing standards to govern this process. Such is the case of the W3C RDB2RDF Working Group[1], which is currently working on a standard language to express relational database to RDF mappings called R2RML [39]. A standard RDB to RDF mapping language will allow vendors to compete on functionality and features. The most relevant RDB-to-RDF approaches are summarized ahead.

- **Triplify.** Auer et al. [4] describe Triplify, a simplified approach based on mapping HTTP-URI requests onto relational database queries. Triplify motivates the need for a simple mapping solution through using SQL as mapping language, for transforming database query results into RDF triples and Linked Data. The mapping is done manually. It uses the table-to-class and column-to-predicate approach for transforming SQL queries results to the RDF data model. This transformation process can be performed on demand through HTTP or in advance (ETL). The approach promotes the reuse of mapping files, through a collection of configurations files for common relational schemata. It can be easily integrated and deployed with numerous popular Web applications such as WordPress, Gallery and Drupal. Triplify also includes a method for publishing update logs to enable incremental crawling of linked data sources. The approach was tested with 160 GB of geographical data from the OpenStreetMap project, showing high flexibility and scalability.
- **D2RQ.** Bizer et al. describe D2RQ [11], which generates the mapping files automatically, using the table-to-class and column-to-predicate approach. D2RQ uses a declarative language, implemented as Jena graph [14], to define the mapping file. The approach allows relational databases to offer their contents as virtual RDF graphs without replication of the RDB in RDF triples. The tool can also provide the RDF dump of the relational database if required. In the virtual access the mapping file is largely used for translating SPARQL to SQL queries. The mapping file may be customized by the user, thereby allowing the ontology reuse in the mapping process.
- **Virtuoso RDF View.** Erling et al. [26] describe the virtuoso RDF View, which uses the table-to-class approach for automatic generation of the mapping file. The

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

mapping file, also called RDF view, is composed by several declarations called "quad map patterns", which specify how the table column values are mapped to RDF triples. Similarity to D2RQ [11], Virtuoso RDF View allows to map arbitrary collections of relational tables, into "SPARQL accessible RDF" without having to convert the whole data into RDF triples. It is important to note that quad map patterns can be stored as triples, and are, therefore, queryable via SPARQL.

- **DB2OWL.** In [20] the authors present the DB2OWL tool, which maps a relational database to a single, local ontology. The DB2OWL mapping file uses the XML based language R2O [5] to describe relationships between database components and a local ontology. This mapping language is used to either execute the transformation in response to a query or to create an RDF dump, in batch mode. The DB2OWL tool adopts the table-to-class and column-to-predicate approach with some improvements, the more significant of which is the identification of object properties.
- **RDBtoOnto.** In [18], Cerbah proposes the RDBtoOnto tool and discusses how to take advantage of database data to obtain more accurate ontologies. The RDBtoOnto is a tool that guides the user through the design and implementation of methods for ontology acquisition using information stored in relational databases. It also supports the data transformation to populate the ontologies. The RDBtoOnto uses the table-to-class and column-to-predicate approach to create an initial ontology schema, which is then refined through identification of taxonomies hidden in the data.
- **Ultrawrap.** Sequeda et al. [43] present the automatic wrapping system called Ultrawrap, which provides SPARQL querying over relational databases. The Ultrawrap tool defines a triple representation as an SQL view in order to take advantage of the optimization techniques provided by the SQL infrastructure. The ontology, which is the basis for SPARQL queries, is generated following the table-to-class approach with First Order Logic, introduced by Tirmizi et al. in [45].
- **Automated Mapping Generation for Converting Databases into Linked Data.** Polfliet et al. [36] propose a method that automatically associates database elements with ontology entities in the mapping generation process. This method uses schema matching approaches, mainly string-based ones, to align RDB elements with ontology terms. D2RQ [11] is used to create the initial ontology schema. This approach provides a rudimentary method for linking data with other datasets, based on SPARQL queries and ***rdfs:label*** tags.

3 The Interoperability Problem

The RDB-to-RDF mapping operation results in the definition of a generic ontology that describes how the RDB schema concepts are represented in terms of RDF classes and properties. The sheer adoption of this ontology, however, is not sufficient to secure interoperability. In a distributed and open system, such as the Semantic

Web, different parties tend to use different ontologies to describe specific domains of interest, raising interoperability problems.

Ontology alignment techniques could be applied to solve heterogeneity problems. Such techniques are closely related to schema matching approaches, which consist of taking two schemata as input and producing a mapping between pairs of elements that are semantically equivalent [38]. Matching approaches may be classified as syntactic vs. semantic and, orthogonally, as *a priori* vs. *a posteriori* [15]. Both syntactic and semantic approaches work *a posteriori*, in the sense that they start with existing datasets, and try to identify links between the two. A third alternative — the *a priori* approach — is proposed in [15], where the author argues that,"when specifying databases that will interact with each other, the designer should first select an appropriate standard, if one exists, to guide design of the exported schemas. If none exists, the designer should publish a proposal for a common schema covering the application domain".

The same philosophy is applicable to Linked Data. In the words of Bizer, Cyganiak and Heath [9]: *"in order to make it as easy as possible for client applications to process your data, you should reuse terms from well-known vocabularies wherever possible. You should only define new terms yourself if you can not find required terms in existing vocabularies"*.

As defined by W3C, ontologies can serve as the global schema or standard for the *a priori* approach. The authors in [15] list the following steps to define a common schema for an application domain

- Select fragments of known, popular ontologies such as WordNet [2] that cover the concepts pertaining to the application domain;
- Align concepts from distinct fragments into unified concepts; and
- Publish the unified concepts as ontology, indicating which are mandatory and which are optional.

According to [30], it is considered a good practice to reuse terms from well-known RDF vocabularies whenever possible. If the adequate terms are found in existing vocabularies, these should be reused to describe data. Reuse of existing terms is highly desirable, as it maximizes the probability of the data being consumed by applications tuned to well-known vocabularies, without further processing or modifying the application. In the following list we enumerate some vocabularies that cover a widespread set of domains and are used by a very large community. As such, to ensure interoperability, these vocabularies should be reused whenever possible [9].

- **Dublin Core Metadata Initiative (DCMI)**[3]. Defines general metadata attributes such as title, creator, date and subject.
- **Friend-of-a-Friend (FOAF)**[4]. Defines terms for describing people, their activities and their relations to other people and objects.

[2] http://wordnet.princeton.edu/

[3] http://dublincore.org/documents/dcmi-terms/

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

- **Semantically-Interlinked Online Communities (SIOC)**[5]. Describes aspects of online community sites, such as users, posts and forums.
- **Description of a Project (DOAP)**[6]. Defines terms for describing software projects, particularly those that are Open Source.
- **Programmes Ontology**[7]. Defines terms for describing programmes such as TV and radio broadcasts.
- **Good Relations Ontology**[8]. Defines terms for describing products, services and other aspects relevant to e-commerce applications.
- **Creative Commons (CC)**[9]. Defines terms for describing copyright licenses in RDF.
- **Bibliographic Ontology (BIBO)**[10]. Provides concepts and properties for describing citations and bibliographic references (i.e., quotes, books, articles, etc.).
- **OAI Object Reuse and Exchange**[11]. Used by various library and publication data sources to represent resource aggregations such as different editions of a document or its internal structure.
- **Review Vocabulary**[12]. Provides a vocabulary for representing reviews and ratings, as are often applied to products and services.
- **Basic Geo (WGS84)**[13]. Defines terms such as latitude and longitude for describing geographically-located things.

Matching two schemata that were designed according to the *a priori* approach, is an easier process as there is a consensus on the semantics of terminology used, thus avoiding possible ambiguities. Unfortunately, this is not what happens in practice. Most teams prefer to create new vocabularies — as do the vast majority of tools that support this task —, rather than spending time and effort to search for adequate matches [32]. We believe that this fact is mainly due to the distributed nature of the Web itself, i.e., there is no central authority one can consult to look for a specific vocabulary. Semantic search engines, such as Watson, works as an approximation for such mechanism. Notwithstanding, there are numerous standards that designers can not ignore when specifying triple sets, and publishing their content.

Only if no well-known vocabulary provides the required terms, the data publishers should define new — data source-specific — terminology [9]. W3C provides a set of guidelines to help users in publishing new vocabularies [8], such as, "if new terminology is defined, it should be made self-describing by making the URIs that identify terms Web dereferenceable. This allows clients to retrieve RDF Schema or OWL definitions of the terms as well as mappings to other vocabularies".

[5] http://rdfs.org/sioc/spec/

[6] http://trac.usefulinc.com/doap

[7] http://purl.org/ontology/po/

[8] http://purl.org/goodrelations/

[9] http://creativecommons.org/ns#

[10] http://bibliontology.com/

[11] http://www.openarchives.org/ore/

[12] http://purl.org/stuff/rev#

[13] http://www.w3.org/2003/01/geo/

4 The StdTrip Process

The StdTrip process aims at guiding users during the conceptual modeling stages of the task of converting relational databases into RDF triples. Most tools that support this task do that by mapping relational tables to RDF classes, and attributes to RDF properties, with little concern regarding the reuse of existing standard vocabularies [4] [26]. Instead, these tools create new vocabularies using the internal database terminology, such as the table and attribute names. We believe that the use of standards in schema design is the only viable way for guaranteeing future interoperability [12] [16] [33]. The StdTrip process is anchored in this principle, and strives to promote the reuse of standards by implementing a guided process comprised by six stages: conversion, alignment, selection, inclusion, completion and output. These stages are detailed in the following subsections. To illustrate our description we are going to use the publication database depicted in Figure 1 throughout the next sections.

It is important to note that we make the implicit assumption that the input database is fully normalized. That is, we assume that the input data is a relational database in the third normal form (3NF). Furthermore, we assume that the user that follows this approach has some knowledge about the application domain of the databases.

4.1 Conversion

This stage consists in transforming the structure of the relational database in an RDF ontology. It takes as input the relational database schema (Figure 1), which contains the metadata of the RDB. This stage is comprised by two major operations. In the first operation, we transform the relational database schema into an Entity-Relationship (ER) model. In the second operation, we transform the Entity-Relationship model, resulting from the previous operation, into an OWL ontology.

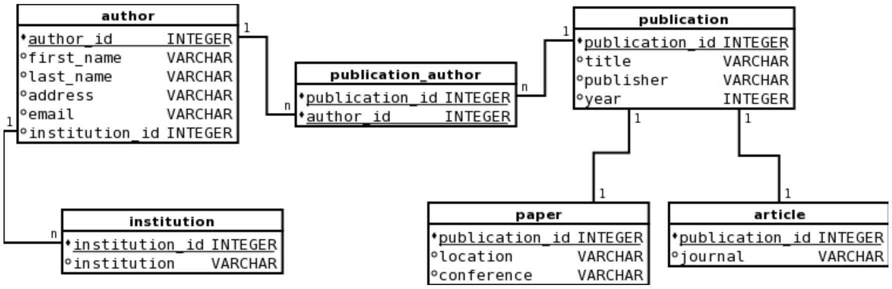

Fig. 1 Author-Publication relational schema.

The reason for splitting the conversion stage in two separate operations is that mapping the relational database model directly to OWL would not properly map some of the attributes, such as a binary relationship, to object properties. Using a direct RDB to OWL mapping approach, the table *publication_author* (Figure 1) would result in the Class *Publication_author* with *publication_id* and *author_id* as subjects, while the RDB to ER to OWL approach would correctly result in two object properties *publication_author* and the inverse property *has_publication_author*.

In the following subsections we describe each operation in more detail, starting with the mapping from relational database model to entity-relationship followed by the conversion process from entity-relationship to OWL.

4.1.1 Relational model to Entity-Relationship

The relational data model, as originally conceived by Codd [19], leaves practically all semantics to be expressed by integrity constraints. Therefore the use of relations as the sole data structure makes the model conceptually and semantically very simple. In order to solve this lack of semantics, we convert the relational database schema into an Entity-Relationship model, which provides a high-level conceptualization to describe the database.

This operation is a combination of ideas and mapping rules proposed by [6], [17], and [31]. This process can be characterized as a reverse engineering process, because the input of this process is the implementation model, and the expected result is a conceptual model. According to [31] the transformation process has the following major steps:

1. **Identification of ER elements for each table:** Each relation (table) in the relational model represents an entity, a relationship or a weak entity in the entity-relationship model. The following mapping rules, extracted from [17] and [31], are the ones we elected in our implementation of the RDB to ER mapping.

    - **Entity:** Corresponds to every primary key that is not composed by foreign keys. In other words, if a relation does not reference other relation schemes, the relation represents an Entity. For instance, in the example depicted in Figure 1, the table *author* with the primary key *author_id* has no foreign keys. Thus the table *author* is an *Entity*.
    - **Relationship:** A table that has a primary key composed by multiple foreign keys represents a relationship element between the tables referenced by these foreign keys. For instance, in the same example, the table *publication_author* has the columns *publication_id* and *author_id* composing a primary key. Both columns are foreign keys referencing the tables *author* and *publication*. Thus the table *publication_author* is a *Relationship* between the tables *author* and *publication*.
    - **Weak Entity or Specialized Entity:** The table whose primary key intersects with the foreign key represents a weak entity or a specialization of the entity

referenced by this foreign key. For instance, the table *article* has *publication_id* as primary key, which is also a foreign key to the table *publication*. Thus we can state that the table *article* is a *Weak Entity* that depends on — or is a specialization of — the *publication* entity.

2. **Definition of relationship cardinality:** The cardinality of a relationship can be 1-n, 1-1 or n-n. Heusler, in [31], states that in order to classify the cardinality of a given relationship we need to verify the data stored in the tables. With the purpose of systematizing this step, we adopted the following rules.
    - **Cardinality n-n:** Every relationship mapped directly from a table has the n:n cardinality. The table *publication_author*, in our example illustrates, such case.
    - **Cardinality 1-1:** This cardinality is found in relationships between an entity and its specialized entity. The tables *article* and *publication* are examples of 1-1 mappings.
    - **Cardinality 1-n:** This cardinality is frequently found in columns that represent foreign keys, but are not part of the primary key. For instance, the column *institution_id* from the table *author* generates a new relationship with 1-n cardinality.
3. **Definition of attributes:** According to [31], in this step every column of a table that is not a foreign key should be defined as an attribute of the entity or the relationship.
4. **Definition of entities and relationships identifiers:** The final major step in the transformation process deals with the entities and relationship identifiers. Heusler in [31] stated that every column that is part of the primary key, but is not a foreign key, represents an entity or a relationship identifier. The table *institution*, in our running example, with its column *institution_id* as primary key, functions as entity identifier for the *institution* entity.

Before starting the ER to OWL mapping operation, we recommend modifying the internal database nomenclature (codes and acronyms) to more meaningful names, i.e, names that better reflect the semantics of the ER objects in question. In our example, the *publication_author* relationship could by modified to *hasAuthor*, that better describes this relationship between *Publication* and *Author*. Compliance to this recommendation will be very useful in later stages of the StdTrip process.

4.1.2 Entity-Relationship to OWL mapping

In order to obtain an RDF representation of the database schema, we have to apply some mapping rules to convert the entity-relationship model, just obtained. The mapping rules used to transform the entity-relationship model are straightforward, due to the fact that we start from a conceptual, entity-relationship model, with the adequate level of database semantics. The transformation rules listed below are a compendium from the work of [29] and [34] adapted for our specific scenario.

- Map each **entity** in the ER to a class in the OWL ontology. For instance, the entity *author* is mapped to the class *Author*.
- Map each **simple attribute of entity** in the ER to a functional datatype property. Domain of the datatype property is the entity, and range is the attribute datatype. For instance, the attribute *address* of the entity *author* is mapped to the datatype property *address* with *author* as domain and *XSD:String* as range.
- Map each **identifier attribute of entity** in the ER to a datatype property tagged with functional and inverse functional. For instance, the identifier attribute *author_id* of the entity *author* is mapped to a functional datatype property *author_id* with *author* as domain and *XSD:Integer* as range.
- Map each **specialized entity** in the ER to a Class tagged with *subClassOf* indicating the owner Class. For instance, the entity *article* is mapped to the class *Article* and the property *subclassOf* related to the class *Publication*.
- Map each **binary relationship without attributes** into two object properties between the relationship entities. One corresponding to the relationship as represented in the ER, and the second as an inverse property of the former one. For instance, the relationship *publication_author* is mapped to a object property with the same name and an inverse object property *isAuthorOf*.
- Map each **binary relationship with attributes** to a class with datatype corresponding to the relationship attribute, and two pairs of inverse object property between the new class and the relationship entities.
- Map the **relationship cardinality** into max and min cardinality restrictions.

The output of the conversion stage corresponding to our example is illustrated in Figure 2. It is important to note that the resulting ontology is a model that simply mirrors the schema of the input relational database depicted in Figure 1.

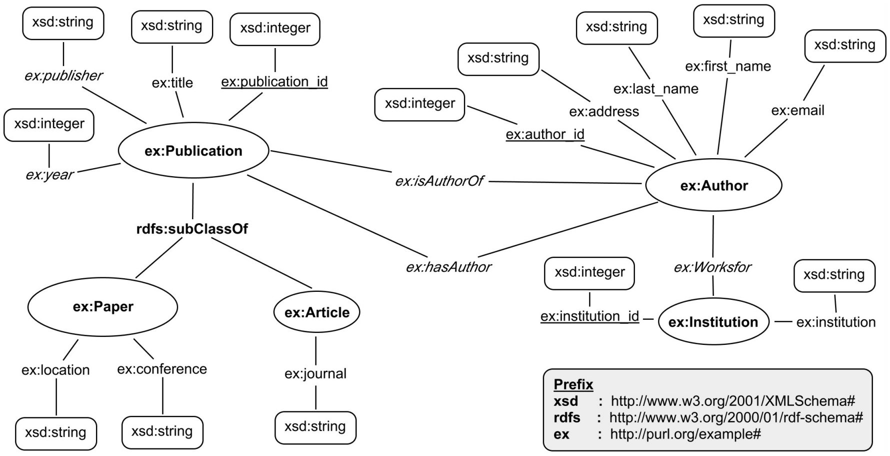

Fig. 2 Ontology representing the output of the conversion stage in the Author-Publication example

4.2 Alignment

The alignment stage is where lies the essence of our approach. As the name suggests, is in this stage that we apply existing ontology alignment algorithms. We aim at finding correspondences between a generical ontology — obtained in the previous stage — and standard well-known RDF vocabularies. The alignment operation is supported by the *K-match* ontology alignment tool, which is based on a collaborative approach to find matches between ontology terms. This tool was inspired by the work of Do Hong Hai [23], in which he presents a composite matching approach for schema matching.

Better than proposing "yet another ontology matching tool", *K-match* capitalizes from years of collaborative research and results obtained by the Semantic Web community, particularly during the OAEI contest [27] [28]. The collaborative aspect of the *K-match* tool comes from the characteristics of the Web 2.0 itself, as the tool is a mashup application that uses and combines functionalities from several alignment tools already available in the form of APIs[14] in the Web. Therefore, the *K-match* tool allows us to use different alignment applications (matchers) — with the option of including new ones — and applies different strategies to combine the results.

The *K-match* tool takes as input two OWL ontologies and produces a mapping indicating which elements of the inputs correspond to each other. In the StdTrip context, the ontology obtained in the previous stage is one of the input ontologies, while the second input ontology is one of a list of common RDF vocabularies (enumerated in Section 3), alternated automatically during repeated executions of the tool. After each execution, a set of matching results is generated. Each result is comprised of a set of mapping between elements, together with similarity values ranging from 0 to 1, eveluating the similarity degree between the mapped elements.

The alignment process is comprised by three steps: the first step consists in the execution of different matchers, the second step combines the results of the previous step applying aggregation strategies, and the final step applies one of several selected strategies to choose the match candidates for each ontology term [24]. The steps of the *K-match* alignment process are depicted as follows.

1. **Matchers execution.** In this step we use three top ranked matchers of the OAEI 2009[15] contest, namely Lily [47], Aroma [22] and Anchor-Flood [42]. Most of them are syntactic matchers, mainly due to the lack of instances stored in standard vocabularies hosted in the Web, a fact that hinders the adoption of semantic, instance-based approaches[16]. The result of the execution of the matcher with *K* matchers, *N* elements from the source ontology and *M* elements from the target ontology is a *K* X *N* X *M* cube of similarity values. It is important to note that we applied a directional match, since the goal is to find all match candidates just for the ontology target. For instance, Table 1 presents the similarity values from a

[14] API : Application Programming Interface

[15] http://oaei.ontologymatching.org/2009/

[16] This is a limitation of existing tools, not of the *K-match* framework, which may be extended to include new matchers in the future.

partial alignment between the Friend of a Friend (FOAF) vocabulary, now called ontology source *O1*, with the single term *ex:last_name*, from the generic ontology obtained in the previous step of the StdTrip process (Figure 2), now called ontology target *O2*.

Table 1 Similarity Cube: Similarity values from a partial alignment between *O1* and *O2* for the term *ex:last_name*, from the Author-Publication example

Matcher *(K)*	**Ontology Source *(N)***	**Similarity Value**
Lily	**foaf:**first_name	0.5
	foaf:familyName	0.5
	foaf:givenName	0.5
Aroma	**foaf:**first_name	0.6
	foaf:familyName	0.8
	foaf:givenName	1.0
Aflood	**foaf:**first_name	0.3
	foaf:familyName	1.0
	foaf:givenName	1.0

2. **Combination strategies.** In this step we combine the matching results of the *K* matchers, executed in the former step and stored in the *similarity cube*, in a unified *similarity matrix* with *M* X *N* result elements. In other words, after applying the aggregation strategy, each pair of ontology terms gets a unified similarity value. For instance, Table 2 presents the combined similarity values obtained for the term *ex:last_name*. The following aggregation strategies are provided by the *K-match* tool to combine individual similarity values for pairs of terms into a unified value:

   - **Max.** This strategy returns the maximal similarity value of any matcher. It is optimistic, in particular in case of contradictory similarity values.
   - **Weighted.** This strategy determines a weighted sum of the similarity values and needs relative weights for each matcher, which should correspond to the expected matchers importance.
   - **Average.** This strategy represents a special case of the *Weighted* strategy and returns the average similarity over all matchers, i.e., considering all them equally important.
   - **Min.** This strategy uses the lowest similarity value of any matcher. As opposed to *Max*, it is pessimistic.
   - **Harmonic mean.** This strategy returns the harmonic mean over the matchers, with values greater than zero.

3. **Selection of match candidates.** The final step is to select possible matching candidates from the similarity matrix obtained in the previous step. This is achieved

Table 2 Similarity Matrix: Similarity values combined from Table 1 for the term *ex:last_name*, from the Author-Publication example

Ontology Source *(N)*	**Similarity Value**
foaf:first_name	0.47
foaf:familyName	0.77
foaf:givenName	0.83

applying a selection strategy to choose the match candidates for each ontology term. For selecting match candidates, the following strategies are available:

- **MaxN.** The ***n*** elements from the source ontology with maximal similarity are selected as match candidates. Having n=1, i.e., Max1, represents the natural choice for 1:1 correspondences. Generally, n>1 is useful in interactive mode to allow the user to select among several match candidates.
- **MaxDelta.** The element from the source ontology with maximal similarity is selected as match candidate, together with all elements with a similarity degree differing by at most a given tolerance value ***d***, which can be specified either as an absolute or relative value. The value ***d*** is set by the user. The idea is to return multiple match candidates when there are several source ontology elements with the same or almost the same similarity value.
- **Threshold.** All elements from the source ontology with a similarity value higher than a given threshold ***t*** are returned. The value ***t*** is set by the user.
- **Max-Threshold.** This strategy represents a special case of the previously strategy, and returns the source ontology element with the maximal similarity above a given threshold ***t***. Again, value ***t*** is fixed by the user.

To illustrate this step, we applied the *Threshold* strategy in the partial result depicted in Table 2, with $t = 0.6$. The result was the choice of **foaf:***familyName* and **foaf:***givenName* as match candidates for **ex:***last_name*.

4.3 Selection

In this stage, human interaction plays an essential role. Ideally, the user should know well the application domain, because he or she will have to choose the vocabulary elements that best represent each concept in the database. The user will select each vocabulary element from a list of possibilities, listed in decreasing order of similarity value obtained as the result of the previus stage.

For instance, in the case of the term *ex:last_name* the user will have to decide between the terms *foaf:givenName* and *foaf:lastName* with 0.83 and 0.77 of similarity value respectively. Figure 3 shows the OWL ontology after the execution of this stage. In cases where there were two or more choices of matching RDF vocabulary terms, we opted always for the ones with higher similarity values.

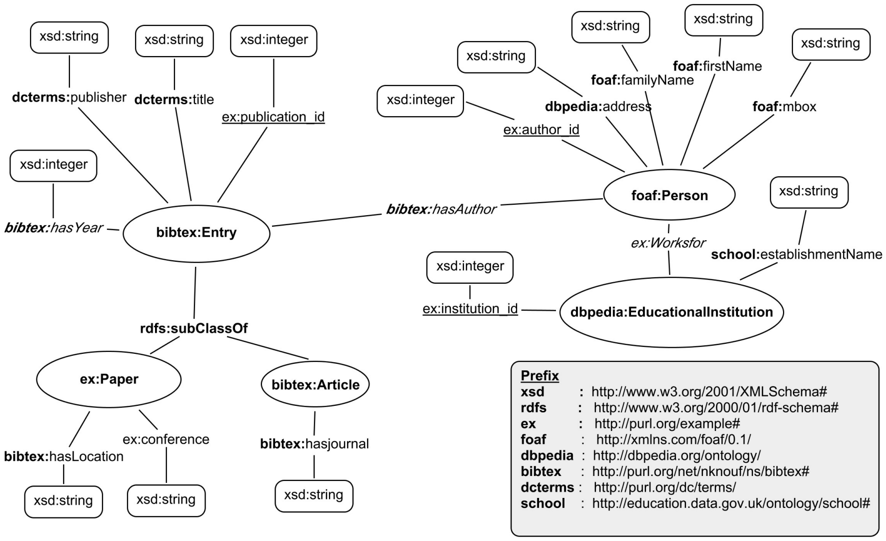

Fig. 3 OWL ontology after the *Selection* stage

4.4 Inclusion

There are cases where the selection stage does not yield any result, i.e., either there is no element in the known vocabularies that matches the concept in the database, or none of suggestions in the list is considered adequate by the user. For such cases we provide a list of terms from other, vocabularies in the web that might be possible matches. The choice of these vocabularies is domain-dependent, and the search, based on keywords, is done using a semantic web searching tools. The rationale is the following, "if your concept is not covered by any of the known standards, look around and see how others dealt with it. By choosing a vocabulary in use, you will make your data more interoperable in the future, than by creating a brand new vocabulary."

This stage is accomplished with the aid of existing mechanisms for searching semantic documents offered by Semantic Web searchers, namely Watson[17] [21], which executes a keywords based search. In order to improve the quality of the results, it is crucial to follow some *"tuning"* and configuration guidelines to get the

[17] http://kmi-web05.open.ac.uk/WatsonWUI/

best out of this type of service. The following list is a compendium of the guidelines that we adopt:

- Restrict the exploration space, searching just for terms belonging to domain ontologies directly related to the database domain.
- Filter expected results specifying the type of term — classes or properties —, i.e., if we are searching for a class, every property will be excluded.
- Extract the term description, if available, and apply similarity algorithms to reduce ambiguity as much as possible.

4.5 Completion

If, for some terms, none of the previous stages was able to provide appropriate RDF mapping, the user will have to define a new vocabulary. During this stage, we help users providing recommendations and best practices on how his or her vocabulary should be published on the Web, how choose an appropriate URI namespace, and its constituent elements (classes and properties). The following list is a collection of best practices for this specific scenario, compiled from [8], [7], [41], [30] and [3].

- **Do you own the domain name?** The URI namespace you choose for your vocabulary should be a URI to which you have write access, so that you will be able to mint URIs in this namespace.
- **Keep implementation-specific out of your URIs:** URIs should not reflect implementation details that may need to change at some point in the future.
- **How big you expect your vocabulary to become?**
  - **Small vocabularies and stable sets of resources,** may be more conveniently served if the entire vocabulary is retrieved in a single Web access. Such a vocabulary would typically use a hash namespace. *Good Relations*[18] is an example of a vocabulary that uses a hash namespace. For instance, the following URI identifies a Class in this vocabulary.

    http://purl.org/goodrelations/v1#**ProductOrServiceModel**

  - **Large vocabularies, to which additions are frequently,** should be arranged to ease the extension of terms in the vocabulary. Therefore, terms must be retrieved through multiple Web accesses. Such a vocabulary would typically use a slash namespace. *Friend of a Friend (FOAF)*[19] is an example of a vocabulary that uses a slash namespace. For instance, the following URI identifies a class in this vocabulary.

    http://xmlns.com/foaf/0.1/**Person**

[18] http://purl.org/goodrelations/v1

[19] http://xmlns.com/foaf/0.1/

- **Name resources in CamelCase:** CamelCase is the name given to the style of naming in which multiword names are written without any spaces but with each word written in uppercase, e.g., resource names like *rdfs:subClassOf* and *owl:InverseFunctionalProperty.*
  - **Start class names with capital letters,** e.g., class names *owl:Restriction* and *owl:Class.*
  - **Start property names in lowercase,** e.g., property names *rdfs:subClassOf* and *owl:inverseOf.*
- **Name resources with singular nouns,** e.g. classes names *owl:DatatypeProperty* and *owl:SymmetricProperty.*

The actual process of publishing a new RDF vocabulary is outside of the scope of the StdTrip process. By providing these guidelines we hope that users understand the value of making the semantics of their data explicit and, more importantly, reusable.

4.6 Output

This is not properly a stage, rather the output of the StdTrip process, which produces two artifacts.

1. **A mapping specification file.** This artifact serves as the core parameterization for a RDB-to-RDF conversion tool. The specification file format can be easily customized for several approaches and tools that provide support to the mechanical process of transforming RDB into a set of RDF. Among them, there are the formats used by Triplify [4], Virtuoso RDF views [26] and D2RQ [11], and also R2RML [39], the new standardized language to map relational data to RDF. For instance, the code below shows a fragment of the mapping specification file for the Triplify tool [4] corresponding to the Author-Publication example.

```
$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" => "foaf:Person",
...);

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

2. **"Triples Schema".** The second artifact is an ontology representing the original database schema, with the corresponding restrictions, and maximizing the reuse of standard vocabularies. The code below is a "Triples Schema" fragment of the Author-Publication, running example.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        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>
...
```

5 Conclusion

In this chapter, we introduced StdTrip, a tool that emphasizes the use of standard-based, *a priori*, design of triples, in order to promote interoperability, the reuse of vocabularies and to facilitate the integration with other datasets in the Linked Data cloud. StdTrip was initially conceived to serve as an aid in a training course on Publishing Open Government Data in Brazil. The course was an initiative of W3C Brasil to promote the adoption of the Linked Data technology by Brazilian government agencies. Target audiences were assumed to have no familiarity with Semantic Web techniques in general, nor with RDF vocabularies, in particular. To promote vocabulary and standards reuse, we designed a tool that "has it all in one place", i.e., supports all the operations needed to create conceptual model. The StdTrip approach served as an educational tool by “reminding” — or by introducing new — RDF vocabulary concepts to users.

We believe our approach can be further improved as follows. First of all, as we discussed in Section 4.1, typically the terminology used to describe the relational database, including table and column names, is inappropriate to be externalized. To examplify this, we could think of a relationship element named *country_id* that relates *City* and *Country*, an acronym *tb_cust* that could represent a table *Customer* or, even worst, an attribute *Ir675F* representing an ISBN code. In such cases, the StdTrip process tackles this lack of semantics with the following techniques:

- A domain expert (e.g. database administrator) first defines an external vocabulary, i.e., a set of terms that will be used to identify the data materialized to Web users. Artificially generated primary keys, the foreign keys that refer to such primary keys and attributes with domains that encode classifications or similar artifacts, when selected for the StdTrip process, should have their internal names replaced by more meaningful names, best suited for external use.
- A common user could replace the inappropriate terminology by consulting documents that fully describe the data represented in the database (e.g. glossary, data dictionary).

It is important to note that, currently, none of these techniques is supported by an automatic or even semi-automatic mechanism during the StdTrip process, making this operation practically unfeasible in the absence of a domain expert or a document that fully describes the database domain. In future work, we plan to add semi-automatic techniques in order to help the user to decide the most adequate terms that characterize the nature of the data itself in the following ways:

- We can take advantage of instance based approaches, such as the one proposed by [46], to suggest suitable names based on the data stored in the dataset. For example, an attribute named *Ir675F*, in the format XXX-XXXXXXXXXX (where Xs are numbers) may be easily identified as ISBN numbers automatically.
- Taking into consideration that the relationships in the ER model — derived from the relational model — often lack proper names, we can use the semantics of the elements related by these relationships and apply Natural Language Processing algorithms to suggest terms that better describe such relationships. For example, a relationship attribute named *country_id*, which relates the entities *City* and *Country*, can be replaced by *isPartOf*, in order to obtain an statement *City isPartOf Country*.
- Following the work of [44], we plan to use Wordnet extensions to expand and normalize the meaning of database comments, using such comments as a source for additional semantics.

Secondly, as we mentioned in Section 4, we assume that the input of the StdTrip is a relational database in third normal form (3NF). This assumption has some drawbacks in practice, as many databases might not be well normalized. Without support for database normalization, users might be tempted to directly take the databases as input even if badly designed. We plan to resolve this drawback in the following ways:

- Following the approach of [25] and [48], we plan to automate the process of finding functional dependencies within data, in order to eliminate data duplication in the source tables, and to algorithmically transform a relational schema to third normal form. Please note that there are cases where the third normal form is not possible, e.g. the U.S. Environmental Protection Agency's Facilities Registry System dataset available as CSV on data.gov that model much meaningful data as plain literal strings.
- We also plan to offer more input options, such as W-Ray [35], in which a set of database views, capturing the data that should be published, is manually defined. In this sense, another interesting and helpful input option could be a valid SQL query against the input database.
- We noticed that many relational databases use autonumbered columns to set tables identifiers (primary key). This autonumber does not work properly as identifier for well-known entities such as people, institutions or organizations. Therefore we plan to include the option of replacing the table primary key, whenever possible, for a more suitable column that better identifies what is represented in the table. For example, a table *Person* that uses as primary key an autonumber column *person_id*, could have this key replaced by a column *SSN* (security social number), which would better identify the data stored in the table *Person*.

Finally, as users are likely to be confronted with more than one choice during the StdTrip process, e.g.,***foaf:****Person* or***foaf:****Agent*, we plan to include a mechanism for capturing rationale to register design decisions during the modeling process (stages discussed in Sections 4.3 and 4.4). A what-who-why memory would be a beneficial asset, allowing the reuse of previous mapping files that could be rapidly updated to adapt to future modifications, improvements and redesign of the dataset.

Acknowledgements The authors would like to thank the reviewers for their contributions. This research was made possible by grants E-26/170028/2008 from FAPERJ and 557.128/2009-9 from CNPq, at the Brazilian Web Science Institute.

References

1. Improving access to government through better use of the web (2009). URL http://www.w3.org/TR/egov-improving/
2. Publishing open government data (2009). URL http://www.w3.org/TR/gov-data/
3. Allemang, D., Hendler, J.: Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL. Morgan Kaufmann (2008)
4. Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., Aumueller, D.: 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 (2009). DOI http://doi.acm.org/10.1145/1526709.1526793

5. Barrasa, J., Corcho, O., Gómez-Pérez, A.: R2O, an extensible and semantically based database-to-ontology mapping language, vol. 3372 (2004)
6. Batini, C., Ceri, S., Navathe, S.B.: Conceptual database design: an Entity-relationship approach (1991)
7. Berners-Lee, T.: Cool uris don't change. Retrieved January 10, 2010, from http://www.w3.org/Provider/Style/URI (1998)
8. Berrueta, D., Phipps, J.: Best practice recipes for publishing rdf vocabularies – w3c working group note. Retrieved December 14, 2010, from http://www.w3.org/TR/swbp-vocab-pub/ (2008)
9. Bizer, C., Cyganiak, R., Heath, T.: How to publish linked data on the web. Retrieved December 14, 2010, from http://www4.wiwiss.fuberlin.de/bizer/pub/LinkedDataTutorial/ (2007)
10. Bizer, C., Heath, T., Ayers, D., Raimond, Y.: Interlinking Open Data on the Web (Poster). In: In Demonstrations Track, 4th European Semantic Web Conference (ESWC2007) (2007)
11. Bizer, C., Seaborne, A.: D2RQ-treating non-RDF databases as virtual RDF graphs (2004)
12. Breitman, K., Casanova, M.A., Truszkowski, W.: Semantic Web: Concepts, Technologies and Applications (NASA Monographs in Systems and Software Engineering). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
13. Breslin, J., Passant, A., Decker, S.: The Social Semantic Web. Springer Publishing Company, Incorporated (2009)
14. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: implementing the semantic web recommendations, pp. 74–83 (2004)
15. Casanova, M.A., Breitman, K., Brauner, D., Marins, A.: Database conceptual schema matching. IEEE Computer **40**(10), 102–104 (2007)
16. Casanova, M.A., Lauschner, T., Leme, L.A.P., Breitman, K., Furtado, A.L., Vidal, V.: A strategy to revise the constraints of the mediated schema. In: Proc. of the 28th Int'l. Conf. on Conceptual Modeling, *Lecture Notes in Computer Science*, vol. 5829, pp. 265–279. Springer (2009). DOI 10.1007978-3-642-04840-1_21
17. Casanova, M.A., de Sá, J.E.A.: Mapping uninterpreted schemes into entity-relationship diagrams: two applications to conceptual schema design. IBM Journal of Research and Development **28**, 82–94 (1984). DOI http://dx.doi.org/10.1147/rd.281.0082
18. Cerbah, F.: Learning highly structured semantic repositories from relational databases. The Semantic Web: Research and Applications pp. 777–781 (2008)
19. Codd, E.F.: A relational model of data for large shared data banks. Communications of the ACM **13**, 377–387 (1970). ACM ID: 362685
20. Cullot, N., Ghawi, R., Yétongnon, K.: DB2OWL: A Tool for Automatic Database-to-Ontology Mapping, pp. 491–494 (2007)
21. d'Aquin, M., Sabou, M., Dzbor, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Motta, E.: Watson: A gateway for the semantic web (2007)
22. David, J.: AROMA results for OAEI 2009 (2009)
23. Do, H.H.: Schema matching and mapping-based data integration (2006). URL http://lips.informatik.uni-leipzig.de/?q=node/211
24. Do, H.H., Rahm, E.: COMA: a system for flexible combination of schema matching approaches, pp. 610–621. VLDB '02. VLDB Endowment (2002). URL http://portal.acm.org/citation.cfm?id=1287369.1287422. ACM ID: 1287422
25. Du, H., Wery, L.: Micro: A normalization tool for relational database designers. Journal of Network and Computer Applications **22**(4), 215–232 (1999). DOI 10.1006/jnca.1999.0096
26. Erling, O., Mikhailov, I.: Rdf support in the virtuoso dbms. Networked Knowledge-Networked Media pp. 7–24 (2009)
27. Euzenat, J., Ferrara, A., Hollink, L., et al.: Results of the ontology alignment evaluation initiative 2009. In: Proc. 4th of ISWC Workshop on Ontology Matching (OM) (2009)
28. Euzenat, J., Shvaiko, P.: Ontology matching. Springer-Verlag, Heidelberg (DE) (2007)
29. Fahad, M.: Er2owl: Generating owl ontology from er diagram. Intelligent Information Processing IV pp. 28–37 (2008)
30. Heath, T., Bizer, C.: Linked Data. Morgan & Claypool Publishers (2011)

31. Heuser, C.A.: Projeto de banco de dados. Sagra Luzzatto (2004)
32. Kinsella, S., Bojars, U., Harth, A., Breslin, J.G., Decker, S.: An interactive map of semantic web ontology usage. In: IV '08: Proceedings of the 2008 12th International Conference Information Visualisation, pp. 179–184. IEEE Computer Society, Washington, DC, USA (2008). DOI http://dx.doi.org/10.1109/IV.2008.60
33. Leme, L.A.P., Casanova, M.A., Breitman, K., Furtado, A.L.: Owl schema matching. Journal of the Brazilian Computer Society **16**(1), 21–34 (2010). DOI 10.1007/s13173-010-0005-3
34. Myroshnichenko, I., Murphy, M.C.: Mapping ER Schemas to OWL Ontologies, vol. 0, pp. 324–329. IEEE Computer Society (2009). DOI http://doi.ieeecomputersociety.org/10.1109/ICSC.2009.61
35. Piccinini, H., Lemos, M., Casanova, M.A., Furtado, A.: W-Ray: A Strategy to Publish Deep Web Geographic Data. In: Proceedings of the 4th International Workshop on Semantic and Conceptual Issues in GIS (SeCoGIS 2010), to appear (2010)
36. Polfliet, S., Ichise, R.: Automated mapping generation for converting databases into linked data. Proc. of ISWC2010
37. Prud'hommeaux, E., Hausenblas, M.: Use cases and requirements for mapping relational databases to rdf. Retrieved December 18, 2010, from http://www.w3.org/TR/rdb2rdf-ucr/ (2010)
38. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. The VLDB Journal **10**(4), 334–350 (2001). DOI http://dx.doi.org/10.1007/s007780100057
39. S., D., S., S., R., C.: R2rml: Rdb to rdf mapping language. w3c rdb2rdf working group. Retrieved December 15, 2010, from http://www.w3.org/TR/r2rml/ (2010)
40. Sahoo, S.S., Halb, W., Hellmann, S., Idehen, K., Thibodeau Jr, T., Auer, S., Sequeda, J., Ezzat, A.: A survey of current approaches for mapping of relational databases to rdf. W3C RDB2RDF Incubator Group report (2009)
41. Sauermann, L., Cyganiak, R.: Cool uris for the semantic web. Retrieved January 18, 2010, from http://www.w3.org/TR/cooluris/ (2008)
42. Seddiqui, M.H., Aono, M.: Anchor-Flood: Results for OAEI-2009
43. Sequeda, J.F., Depena, R., Miranker, D.P.: Ultrawrap: Using sql views for rdb2rdf. Proc. of ISWC2009
44. Sorrentino, S., Bergamaschi, S., Gawinecki, M., Po, L.: Schema normalization for improving schema matching. In: Proc. of the 28th International Conference on Conceptual Modeling (ER '09), pp. 280–293. Springer-Verlag, Berlin, Heidelberg (2009). DOI http://dx.doi.org/10.1007/978-3-642-04840-1_22
45. Tirmizi, S., Sequeda, J., Miranker, D.: Translating sql applications to the semantic web, pp. 450–464 (2008)
46. Wang, J., Wen, J.R., Lochovsky, F., Ma, W.Y.: Instance-based schema matching for web databases by domain-specific query probing. In: Proc. of the 13th international conference on Very large data bases (VLDB '04), pp. 408–419. VLDB Endowment (2004)
47. Wang, P., Xu, B.: Lily: Ontology alignment results for OAEI 2009 (2009)
48. Wang, S.L., Shen, J.W., Hong, T.P.: Mining fuzzy functional dependencies from quantitative data, vol. 5, pp. 3600–3605 vol.5 (2000). DOI 10.1109/ICSMC.2000.886568