4
The StdTrip Process

4.1 The "a priori" Approach

As discussed in section 2.8 the *a priori* approach emphasizes the reuse of widely adopted standards for database design as a means to secure future interoperability. The same philosophy is applicable to Linked Data, the Semantic Web standard used for the publication of Open Data. In the word of Bizer, Cyganiak and Heath:

> *"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"* [Bizer et al. 2007].

Unfortunately, that is not what happens in practice. Most teams prefer to create new vocabularies (as do the vast majority of triplification tools), rather than spending time and effort to search for adequate matches [Kinsella et al. 2008]. We believe this is mostly due to the distributed nature of the Web itself, i.e., there is no central authority one can consult when looking for a specific vocabulary. Semantic search engines, such as Watson, works as an approximation. Notwithstanding, there are numerous standards that designers can not ignore when specifying triple sets, and publishing their content. Section 2.3 presents a list of some of these. The term standard is used in a loose way, in that it encompasses vocabularies with different status (recommended, submitted, etc.) in regards with standard authorities.

Confident that good design, i.e. based on well known standards, will promote and facilitate future interoperability, we propose the StdTrip Process, detailed in this chapter.

4.2 The StdTrip Process

The StdTrip process aims at guiding users during the conceptual modeling stages of the triplification process, which in our scenario is defined as the translation of legacy data stored in relational databases into sets of RDF triples. Most triplifying tools today, such as Triplify [Auer et al. 2009], D2RQ [Bizer & Seaborne 2004] and Virtuoso RDF View [Erling & Mikhailov 2009], do that by mapping relational tables to RDF classes, and attributes to RDF properties, with little concern regarding the reuse of existing standard vocabularies. 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 [Breitman et al. 2006] [Casanova et al. 2009] [Leme et al. 2010]. 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. The StdTrip process architecture is represented in Figure 5, as follows.

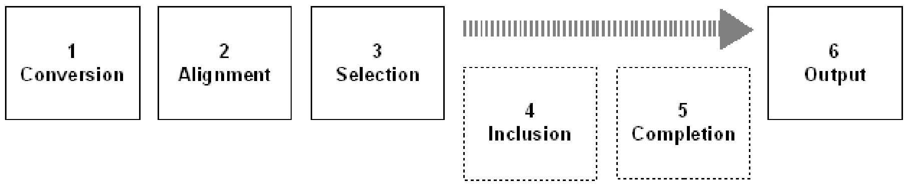

Figure 5: StdTrip Architecture.

The proposed process is comprised by the conversion, alignment, selection, inclusion, completion and output stages, which are detailed in the following sections. To illustrate our description we are going to use a relational database that represents an Author-Publication domain, throughout the next sections.

The relational database, depicted in Figure 6, stores publications from authors. A publication can be a conference paper or a journal article, while an author must work or study for an institution.

Before explaining each stage in more detail, it is important to note that we make implicit assumption that the input database is fully normalized. That is, we assume that the input to the conversion stage is in third normal form (3NF) and the user, who follows this approach, should know the application domain of the databases, such as the Data Administrator (DA).

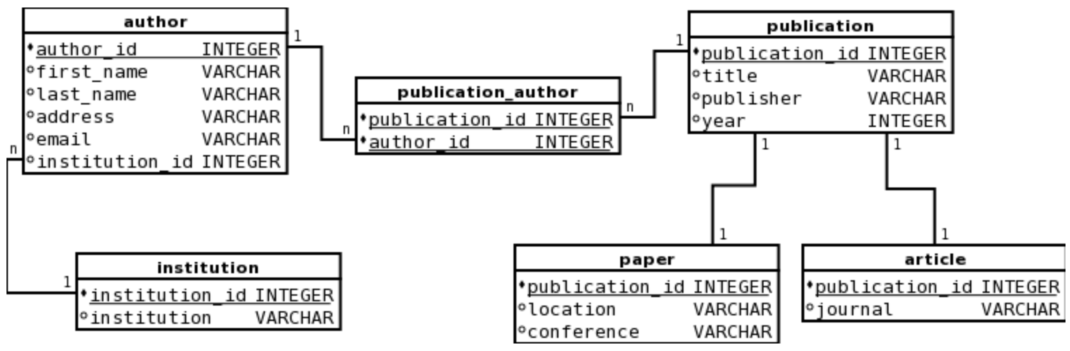

Figure 6: Author-Publication relational schema.

4.3 Conversion Stage

This stage consists in transforming the structure of the relational database to an RDF ontology. It takes as input the relational database schema (Figure 6), which contains the metadata of the RDB. This transforming stage is comprised of two major operations. During the first operation we transform the relational database schema into an Entity Relationship (ER) model.

The second operation consists of transforming the Entity Relationship model, obtained as the result of the previous operation, into an OWL ontology. The reason for breaking down the conversion stage 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. The table *publication_author* (Figure 6), 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*.

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

4.3.1 Relational model to Entity Relationship

The relational data model, as originally conceived by Codd [Codd 1970], 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 remedy the lack of semantics, we convert the relational database schema into an entity relationship model, which provides a high-level conceptualization in which to describe databases.

This transformation operation is a combination of ideas and mapping rules proposed by [Casanova & De Sa 1984], [Heuser 2004], [Batini et al. 1991]. This process can be characterized as a reverse engineering process, since the input of this process is the implementation model, and the expected result is a conceptual model.

According to [Heuser 2004] 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 [Heuser 2004], [Casanova & De Sa 1984] are the ones we elected in our implementation of the RDB to ER mapping.

   – **Entity :** 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 6 the table *author* with a primary key *author_id*, is not a foreign keys. Thus the table *author* is an entity.
   – **Relationship :** The table which has a primary key composed by multiple foreign keys, represents a relationship element, between the tables referenced by these foreign keys, in the ER model. For instance, in the example the table *publication_author* with the column *publication_id* and *author_id* as primary key. Both columns are foreign keys, which reference the tables *author* and *publication*. Thus the table *publication_author* is an Relationship between the tables *author* and *publication*.
   – **Weak Entity or Specialized Entity :** The table which primary key intersects with the foreign key, represents a weak entity or a specialization of the entity referenced by this foreign key. For instance, in the table *person* with *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.

   The entity relationship diagram depicted in Figure 7 is the result of applying the above rules to the relational model depicted by Figure 6.

2. **Definition of relationship cardinality :** The cardinality of a relationship can be 1-n, 1-1 or n-n. Heusler in [Heuser 2004] states that in order to classify the cardinality for a given relationship we need to verify the

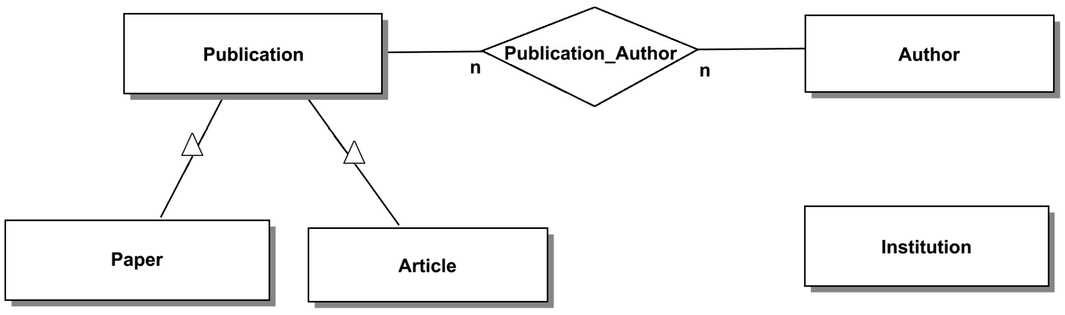

Figure 7: ER : Identification of ER elements for each table

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 illustrated one 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 :** Found in columns which serves as foreign keys, but are not part of the primary key. For instance, the column *institution_id* in the table *author* generates a new relationship with 1-n cardinality.

The corresponding entity relationship diagram that results from the application of the rules to the ER diagram depicted in Figure 7 is showed in Figure 8.

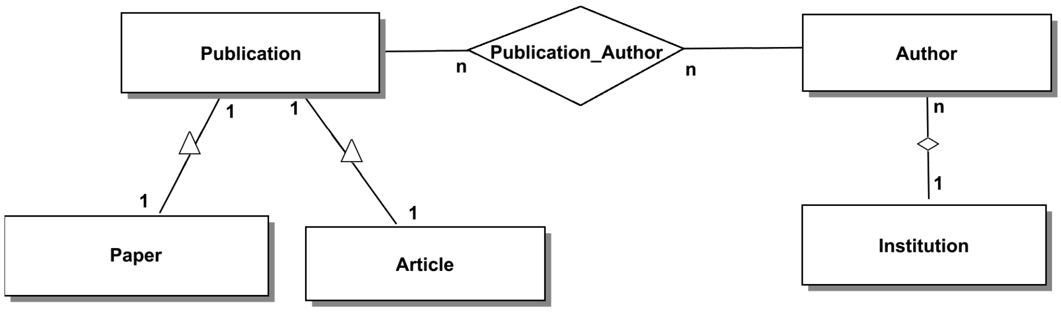

Figure 8: ER : Definition of relationship cardinality for the Author-Publication example

3. **Definition of attributes:** According to [Heuser 2004] in this step every column of the relation that is not a foreign key should be defined as an attribute of the entity or the relationship.

The result of applying this rule to the ER diagram depicted in Figure 8 is illustrated in Figure 9, as follows. Note that all attributes presented in the original database schema (Figure 6), are now included in the ER diagram.

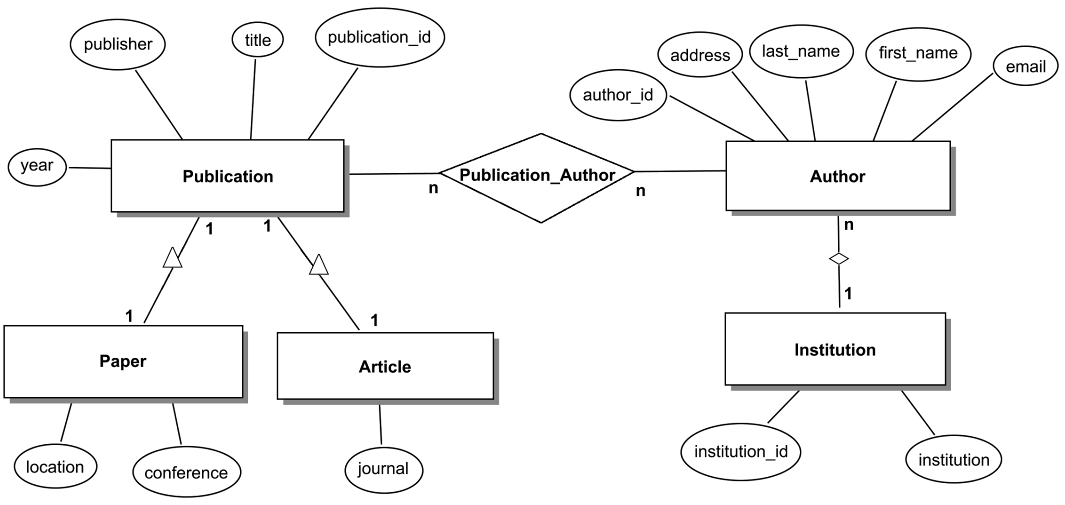

Figure 9: ER : Definition of attributes

4. **Definition of entities and relationships identifiers:** The final major step in the transformation process, deals with the entities and relationship identifiers. Heusler in [Heuser 2004] 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. Figure 10 illustrates the resulting ER diagram when this rule is applied. The attributes with the underlined name represent the entity or relationship identifiers. This ends the relational to ER mapping operation.

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 renamed to *hasAuthor*, that better describes this relationship between *Publication* and *Author*. Compliance to this recommendation will be very useful at later stages of the StdTrip process.

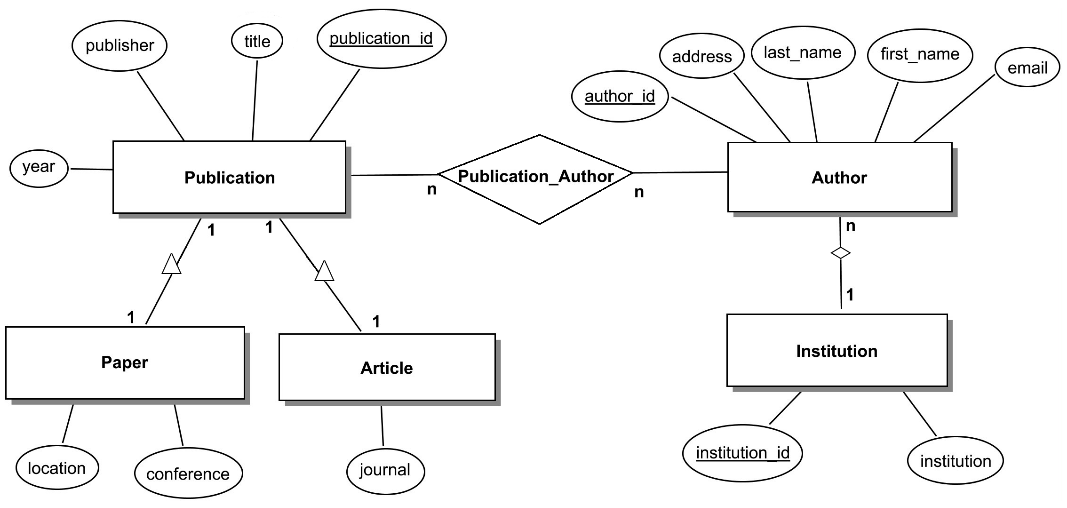

Figure 10: ER : Definition of entities and relationships identifiers

4.3.2 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, obtained as the result of the previous step.

The mapping rules to transform the entity-relationship model to OWL are straightforward, due to the fact that we already start from a conceptual, entity relationship model, with an adequate level of database semantics. The transformation rules listed below are a compendium from the work of [Fahad 2008] and [Myroshnichenko & Murphy 2009], adapted for our specific scenario.

– Map each **entity** in the ER into a Class in the OWL ontology. For instance, the entity *author* is maped to a *Author* Class.

– Map each **simple attribute of entity** in the ER into a functional datatype property. The domain of the datatype property is the entity, and range is the attribute datatype. For instance, the attribute *address* in the entity *author* is mapped to a datatype property *address* with *author* as domain and ***xsd:****String* as range.

– Map each **identifier attribute of entity** in the ER into a datatype property tagged with functional and inverse functional. For instance, the identifier attribute *author_id* in 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 into a Class and tagged with *subClassOf* indicating the owner Class. For instance, the entity *article*

ER component	OWL component
Entity	Class
Specialized entity	Class + subClassOf indicating the owner Class.
Simple entity attribute	Functional datatype property.
Identifier entity attribute	Datatype property + functional and inverse functional
Binary relationship without attributes	Two object properties between the relationship
Binary relationship with attributes	Class + Attribute Datatype property + Two pairs of inverse object property between new Class and the relationship entities.
Relationship cardinality	Max and Min cardinality restrictions.

Table 2: Correspondence between ER schema and OWL ontology components

is mapped to a *Article* Class and subclassOf property of the *Publication* Class.

- Map each **binary relationship without attributes** to two object properties between the relationship entities. The first should correspond to the relationship as represented in the ER, and the second to as an inverse property of the former. For instance, the relationship *publication_author* is mapped to a object property with the same name, *publication_author*, and to an inverse object property *isAuthorOf*.
- Map each **binary relationship with attributes** into a Class with its datatype corresponding to the relationship attribute, and two pairs of inverse object properties between the new Class and the relationship entities.
- Map the **relationship cardinality** into max and min cardinality restrictions.

The conversion output for the example entity relationship diagram (Figure 10) to OWL is illustrated by Figure 11, as follows. It is important to note that the OWL ontology, which is the result of the conversion stage, is a model that simply mirrors the schema of the input relational database depicted in Figure 6.

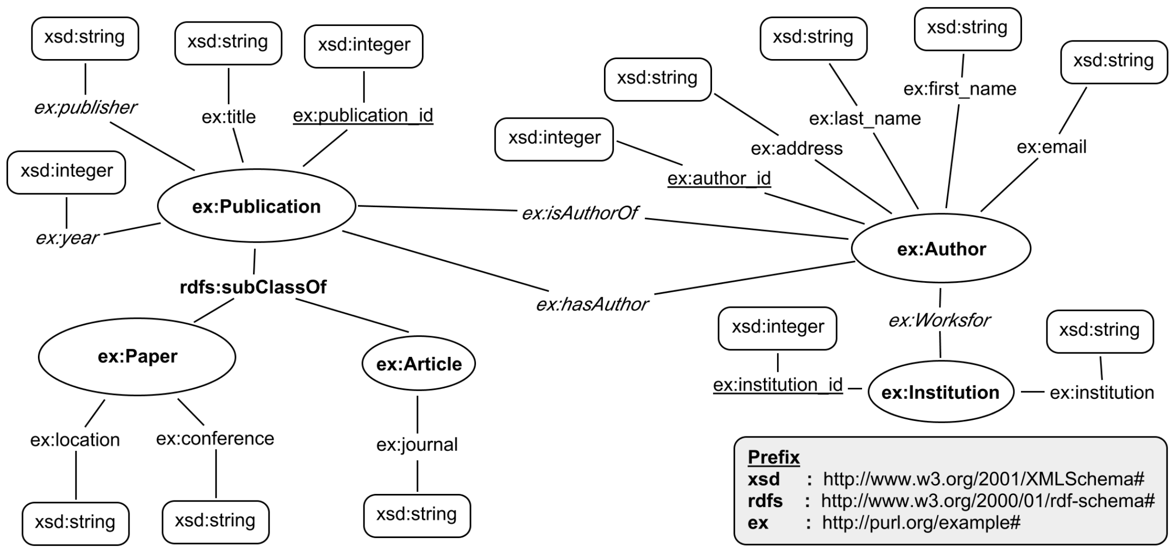

Figure 11: OWL ontology that resulted from applying the transformation process ER to OWL to the Author-Publication example

4.4 Alignment

The alignment stage is where the essence of our approach lies and, as the name suggests, it is during this step that we apply existing ontology alignment algorithms. We aim at finding correspondences between the ontology obtained in the previous stage and standard RDF vocabularies. Table 3 presents a list of some common, widely adopted RDF vocabularies. The alignment process is supported by the *K-Match* ontology alignment tool, which is described below.

4.4.1 K-Match

The *K-Match* tool takes as input two OWL ontologies and produces a mapping indicating which elements of the input ontology logically correspond to others in the second ontology. In the StdTrip context, the ontology obtained in the step 4.3 of the StdTrip process, also called “database ontology”, must be one of these input ontologies, while the second input ontology is one RDF vocabulary, also called “standard ontology” (such as the ones in Table3), alternated automatically during repeated executions of the tool. After each execution, a list of alignments is generated. The list is comprised of a set of mapping elements with similarity values that range from 0 to 1, depending on the similarity degree between elements pairs.

The alignment process consists of three steps: the first step comprises the execution of different matchers, the second step combines the results obtained in the previous step by applying aggregation strategies and the final step consists in applying a strategy to choose the best matches candidates for each

Ontology Name	Prefix	Namespace
Change Set	cs	http://purl.org/vocab/changeset/schema#
DBpedia Ontology	dbpedia	http://dbpedia.org/ontology/
Dcat: Data Catalog Vocabulary	dcat	http://www.w3.org/ns/dcat#
Dublin Core	dc	http://purl.org/dc/elements/1.1/
Dublin Core Terms	dcterms	http://purl.org/dc/terms/
FOAF: Friend Of A Friend	foaf	http://xmlns.com/foaf/0.1/
Geo: Geo Positioning	geo	http://www.w3.org/2003/01/geo/wgs84_pos#
GeoNames	gn	http://www.geonames.org/ontology#
MOAT: Meaning Of A Tag	moat	http://moat-project.org/ns#
Music Ontology	mo	http://purl.org/ontology/mo/
Programmes Ontology	po	http://purl.org/ontology/po/
SIOC: Semantically-Interlinked Online Communities	sioc	http://rdfs.org/sioc/ns#
SKOS: Simple Knowledge Organization System	skos	http://www.w3.org/2004/02/skos/core#
voiD: Vocabulary of Interlinked Datasets	void	http://rdfs.org/ns/void#

Table 3: RDF Standard Vocabularies

ontology term. These steps, are illustrated by the Figure 12.

In what follows we describe the steps of the alignment process in more detail.

1. **Matchers execution** As the name suggests, in this step we execute iteratively different matchers selected by the user. The result produced in this step with K matchers, N elements from the standard ontology and M elements from the database ontology is a KxNxM cube of similarity values that range from 0 to 1, depending on the similarity degree between elements pairs. This cube is called "The Similarity Cube".

   For instance, Table 4 presents the similarity values from a partial alignment between the Friend of a Friend (FOAF) vocabulary, now called standard ontology *O1*, and the ontology obtained in the step 4.3 of the StdTrip process (Figure 11), now called ontology *O2*.

2. **Combination strategies:** In this step we applies a combination strategy to combine the matching results stored in the *Similarity Cube*, in a unified *Similarity Matrix* with MxN result elements. In other words for each pair of ontology terms, from the ontology O1 and O2, we provide an *unified* similarity value.

   For instance the Table 5 presents the result of this combination step for the example of Table 4 using a strategy called *Average* strategy (see

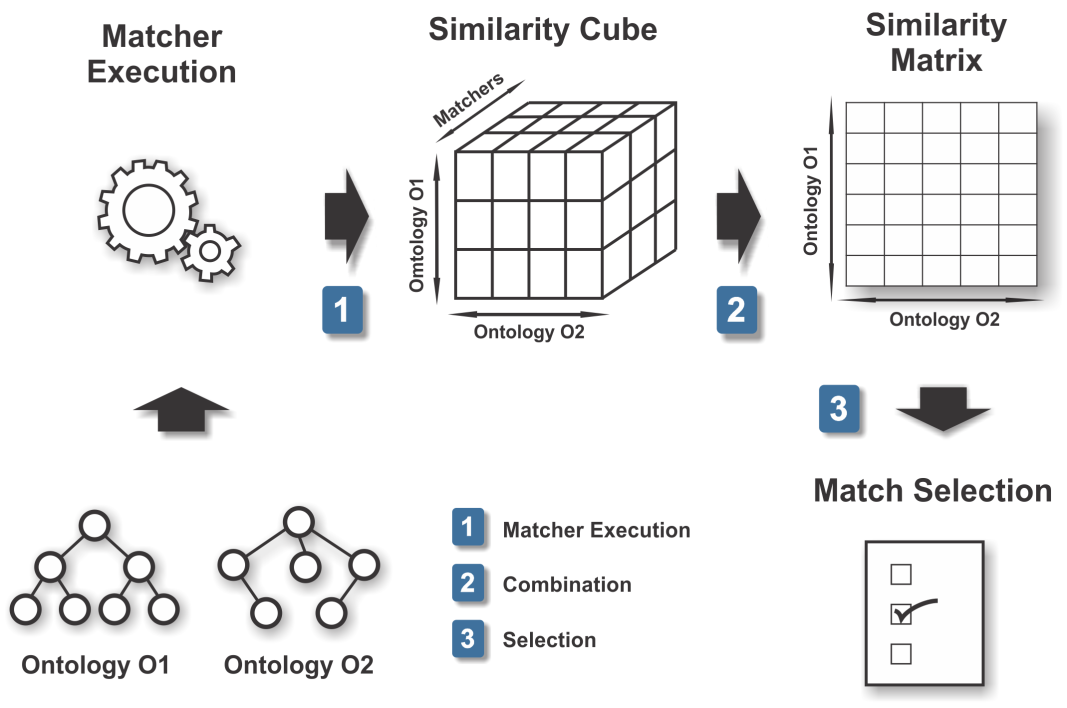

Figure 12: The *K-Match* overall matching process

details in Section 5.2).

3. **Selection of match candidates:** In the final step are selected possible matching candidates from the similarity matrix, obtained in the previous step. This is achieved applying a selection strategy to choose the match candidates for each ontology term. It is important to note that, in this step is apply an unidirectional match selection, as our goal is to find match candidates just for the database ontology terms.

   To illustrate this step, let us apply a strategy called *Threshold* strategy (see details in Section 5.2) to the results presented in Table 5, with ***t*** value set to **0.6**. The final result would be ***foaf:***_familyName_ and ***foaf:***_givenName_ as match candidates for the term ***ex:***_last_name._

4.5 Selection

During the selection stage a human user selects the term that he or she considers the best match for each of the database ontology terms. By best match we mean the term that best represent each of the database concepts. Ideally the user in this stage is a domain expert. He or she will have to choose the vocabulary element from a list of possibilities, listed in decreasing order of similarity value, obtained as the result of step 4.4.

Matcher *(K)*	**Standard Ontology** *(N)*	**Database Ontology** *(M)*	**Similarity Value**
Lily	**foaf:**first_name	**ex:**last_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**

Table 4: Similarity Cube : Similarity values from a partial alignment between *O1* and *O2* for the Author-Publication example

Ontology Source *(N)*	**Ontology Target** *(M)*	**Similarity Value**
foaf:first name	**ex:**last_name	**0.47**
foaf:familyName		**0.77**
foaf:givenName		**0.83**

Table 5: Similarity Matrix : Combined similarity values combined of Table 4 for the Author-Publication example

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*, (0.83 and 0.77 of similarity respectively.) Figure 13 shows the OWL ontology, that represent our example, after the execution of this stage. In cases where there were two or more choices of matching RDF vocabulary terms, we opted for the ones with higher similarity values.

4.6 Inclusion

There are cases where the selection stage does not yield any result (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. In such cases we provide a list of terms from other vocabularies in the Web that might provide a possible match. The choice of these vocabularies is domain-dependent, and the keyword-based search, is done with the aid of a Semantic Web searcher such as Watson [Sabou et al. 2007]. 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 vocab-

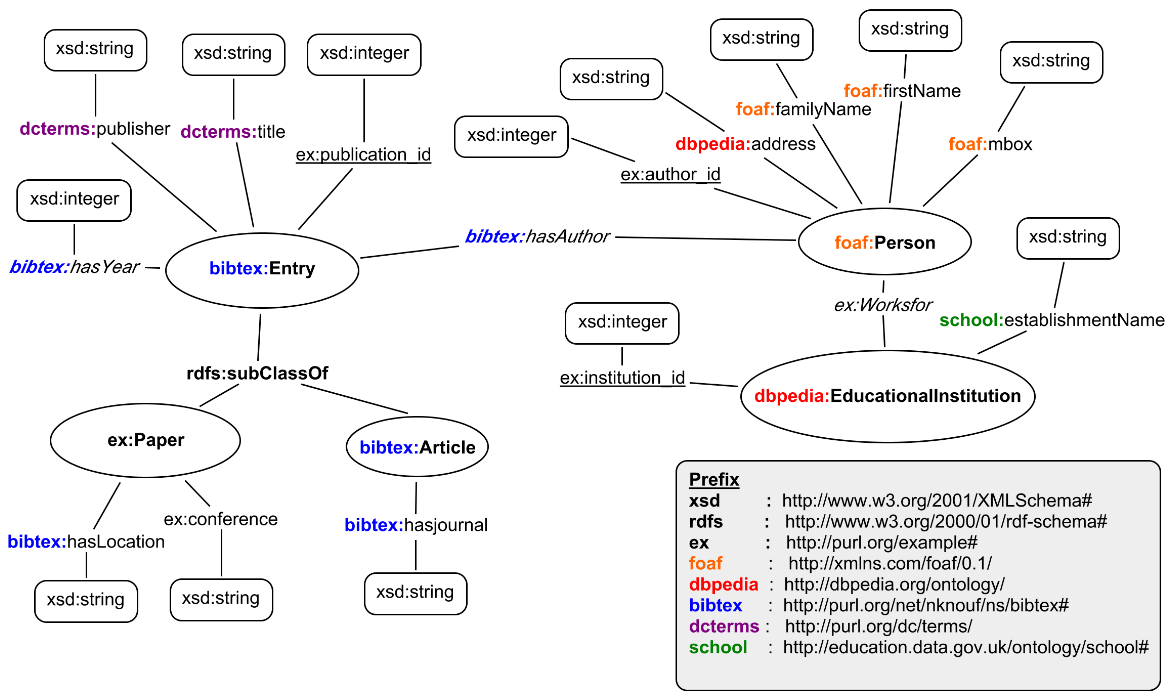

Figure 13: OWL ontology after the *Selection* stage

ulary in use, you will make your data more interoperable in the future, than by creating a brand new vocabulary."

To improve the quality of the results, it is crucial to follow some *"tuning"* and configuration guidelines to get the best out of this type of service. The following list is a compendium of the guidelines the StdTrip implements:

– Restrict the explore space, just searching in domain ontologies with direct relation to the database domain.

– Filter expected results to specific types of concepts (classes or properties), i.e. if we are searching for a specific class, every property will be excluded.

– If possible, extract the term description and apply a similarity algorithm. in order to reduce the ambiguity.

4.7 Completion

If none of the previous stages provided an appropriate RDF mapping for a given term, the user needs to define a new one. During this stage, we help users in the task of providing recommendations and best practices on how his or her vocabulary should be published on the Web. We also helps user choose an appropriate URI namespace, and its constituent elements (classes and properties).

The following list is the collection of best practices the StdTrip approach proposes, compiled from the following references [Berrueta & Phipps 2008], [Berners-Lee 1998c], [Sauermann & Cyganiak 2008], [Heath & Bizer 2011], [Allemang & Hendler 2008].

- **Do you own the domain name?** The URI namespace you choose for your vocabulary should be a URI to which you have write access. In order to minting 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?**
  - **For small vocabularies and stable sets of resources,** it may be most convenient to serve the entire vocabulary in a single Web access. Such a vocabulary would typically use a hash namespace, and a Web access. i.e *Good Relations*[24] is an example of a vocabulary that uses a hash namespace. For instance the following URI identified a Class in this vocabulary.

    http://purl.org/goodrelations/v1#**ProductOrServiceModel**
  - **For large vocabularies, to which additions are frequently,** should be arranged to easily extend the terms in the vocabulary. Thus, may be retrieved through multiple Web accesses. Such a vocabulary would typically use a slash namespace. i.e *Friend of a Friend (FOAF)*[25] is an example of a vocabulary that uses a slash namespace. For instance the following URI identified a Class in this vocabulary.

    *http://xmlns.com/foaf/0.1/***Person**
- **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. 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 with lowercase letters,** e.g. property names ***rdfs:****subClassOf* and ***owl:****inverseOf.*

[24]http://purl.org/goodrelations/v1
[25]http://xmlns.com/foaf/0.1/

– **Name classes with singular nouns,** e.g. classes names ***owl:****DatatypeProperty* and ***owl:****SymmetricProperty.*

The actual process of publication a new RDF vocabulary is outside 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.

4.8 Output

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

1. **The Triples Schema:** An ontology that contains the original database schema in the OWL format, with corresponding restrictions, and maximizing the reuse of standard vocabularies. The Triple Schema is intended to be published together with the triples, as a means to provide additional information to data consumers.

2. **A mapping specification file:** Which serves as the core parameterization for a RDB-to-RDF conversion tool. This specification 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 these are Triplify [Auer et al. 2009], Virtuoso RDF views [Erling & Mikhailov 2009], D2RQ [Bizer & Seaborne 2004] and the R2RML [Das et al. 2010]. By feeding the mapping specification to a RDB-to-RDF tool we are able to produce the triples set from the original database.