A method and guidelines for the cooperation of ontologies and relational databases in Semantic Web applications

Loris Bozzato[1⋆], Stefano Braghin[2], and Alberto Trombetta[3]

[1] Data and Knowledge Management Unit, Fondazione Bruno Kessler, Trento, Italy
[2] School of Computer Engineering, Nanyang Technological University, Singapore
[3] Dip. di Scienze Teoriche e Applicate, Univ. degli Studi dell'Insubria, Varese, Italy

bozzato@fbk.eu, s.braghin@ntu.edu.sg, alberto.trombetta@uninsubria.it

Abstract. Ontologies are a well-affirmed way of representing complex structured information and they provide a sound conceptual foundation to Semantic Web technologies. On the other hand, a huge amount of information available on the web is stored in legacy relational databases. The issues raised by the collaboration between such worlds are well known and addressed by consolidated mapping languages. Nevertheless, to the best of our knowledge, a best practice for such cooperation is missing: in this work we thus present a method to guide the definition of cooperations between ontology-based and relational databases systems. Our method, mainly based on ideas from knowledge reuse and re-engineering, is aimed at the separation of data between database and ontology instances and at the definition of suitable mappings in both directions, taking advantage of the representation possibilities offered by both models. We present the steps of our method along with guidelines for their application. Finally, we propose an example of its deployment in the context of a large repository of bio-medical images we developed.

1 Introduction

Ontology-based knowledge representation systems are well known to be successful in representing complex and heterogeneous information. In particular, recently, Semantic Web tools and systems permit to build and reason over ontologies providing logically founded representations and even increasing possibilities for data size. Moreover, the growing interest and availability of Semantic Web ontologies opens the possibility to reuse known data sources and, above all, to share and integrate information between systems.

On the other hand, the vast majority of data is nowadays stored in relational databases, so tools and techniques bridging ontology-based repositories and relational databases are needed in order to effectively deploy the potential provided by ontology-based representations. Significant efforts have been

⋆ This work has been realized while the first and second author were working at Univ. degli Studi dell'Insubria, Varese, Italy.

made to make possible to provide translations between ontologies and relational schemas in order to easily *publish* readily available database data: however, there is no accepted way on how to use such tools to let cooperate an existing relational database system with a paired ontology based system. For example, to the best of our knowledge, there is no method supporting the decision on what to represent and how to map information in both directions by using already available mapping languages and tools (such as D2R [5,6], Virtuoso RDFview[4] and Sponger[5] just to name a few).

In this work we propose our experiences in the collaboration of an ontology-based knowledge base and a legacy relational database under a single application. In particular, believing in the fact that the problems of this setting can be quite common, we try to generalize the approach that we chose in our case to a general method for the integration between an ontology and a relational database schema, when deployed together in a Semantic Web-based application. We refer to the definition of *method* provided in the context of knowledge re-engineering [17]: a set of "orderly processes or procedures used in the engineering of a product or performing a service". More precisely, we define a sequence of steps that an application designer may follow in order to decide *how* and *what* to map between an ontology and a relational database schema.

We point out that we do not aim at defining a novel mapping language between ontologies and relational schemas. Rather we aim at a method for deciding what are, loosely speaking, the relationships occurring between the ontology and the relational database, e.g. to decide what data stored in the relational database may be fruitfully published as RDF or on what data apply the inference tools proper to ontologies, that is, how to distribute data between both repositories in order to take advantage of the capabilities of the two representations. We have deployed our method and guidelines during the implementation of a large image database currently in use at a veterinary institute (namely, *Istituto Zooprofilattico Sperimentale della Lombardia e dell'Emilia Romagna*, *IZSLER* for short[6]) serving a large user base distributed over more than fifteen sites in northern Italy. In the following we will briefly introduce the structure of the system that lead us to the formulation of this method: the *Imm@base system*, a repository of bio-medical images supporting advanced classification-based functionalities.

1.1 Motivating scenario

The definitions of the method and of the guidelines described in this paper have been carried out in the context of a project to satisfy the necessity of a major italian veterinary institute, the previously mentioned IZSLER. The requirement was to create a repository of biomedical pictures to be annotated with semantic information from well-known biomedical taxonomies, such as ICTVdb[7] and

[4] http://virtuoso.openlinksw.com/
[5] http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VirtSponger
[6] http://www.izsler.it
[7] http://www.ictvdb.org/

NCBI[8]. Moreover, the institute required a further classification of the pictures according to the medical cases they refer to. Such information is stored in a legacy RDB system, called DARWIN, which can not be modified for legal and pragmatic reasons. As an example of the firsts, the information stored in the database is used to quantify the refund which several farmers are entitled of in case of epidemics.

The architecture of the resulting application is shown in Fig. 1. According to the present architecture, the user interacts with the application through a web interface developed in PHP. Such interface provides, in a comprehensive way:

(i). a guided procedure to upload new pictures, properly annotated with metadata retrieved from the ontology database, which – we remind – contains both semantic data from the domain ontologies and a semantic technology-based representation of the data contained in the legacy DARWIN system used by the veterinary institute.
(ii). a web form for retrieving pictures and medical cases matching complex criteria defined by the user.

All the semantic data is retrieved via both ad-hoc and dynamically composed SPARQL queries used against a Joseki end-point. The domain ontologies data and the annotated pictures are stored in a PostgreSQL database while the DARWIN system uses of a SQLServer database. The first database has been created using the tools provided by the Jena API while the others are connected by means of D2RQ [5,6] mappings.

As it is easy to understand, the proposed architecture asks for the definition of a clear policy of cooperation between the semantic and relational repositories. After summarizing the related works on such cooperations, we introduce our solution, presented as a general method specification.

2 Related works

Several works address the issue of generating Semantic Web content from data stored in traditional databases. Such works can be classified, according to the chosen approach, in three categories:

(i). Annotation of the data extracted from databases with informations tracking how data have been obtained,
(ii). Mapping of the database model to an ontology,
(iii). Generation of an ontology related to the relational model of the DB.

The first approach works with the so-called DeepWeb [9] only and requires the database model to be public [8].

The second approach consists in mapping the database models to a given ontology by means of a mapping language in order to provide access to the content of the database as if it were a "semantic repository" [19]. Examples of

[8] `http://www.ncbi.nlm.nih.gov/`

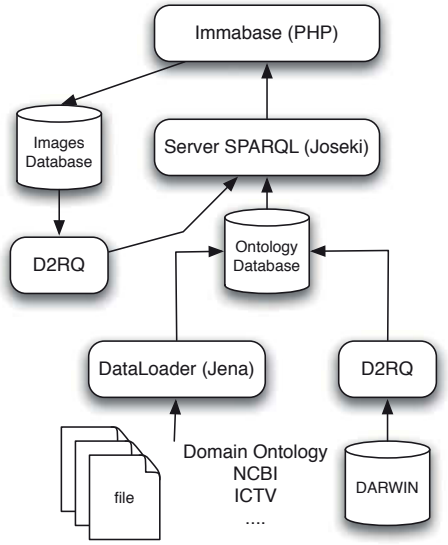

Fig. 1. The architecture of Imm@base web-application

such approach are D2RQ [5,6] and R2O [4]. The first one takes advantage of a proprietary mapping language, derived from the Jena assembler language, to allow the user to incorporate domain semantics in the mapping process. R2O, instead, is a XML based declarative language to express mappings between RDB elements and an ontology. The mappings realized with R2O can be used to "detect inconsistencies and ambiguities" in mapping definitions. A more detailed analysis of mapping languages and tools can be found in [14], where the authors also introduce interesting guidelines about how to further develop such mapping languages. In our work we take advantage of languages provided by works like [6] but proposing a more general methodology for the co-existence of databases and ontologies.

The third approach consists of the semi-automatic generation of an ontology from the database schema [15,16]. Such approach typically uses reverse-engineering techniques to generate the ontology from the database schema, like the ones we describe in Section 3.1, and to migrate the mapped data from the database creating ontological instances based on the tuples.

Moreover, several works have been presented with respect to the development of tools and algorithms to automatically match and merge ontology schemas, such as [1,3,13,18] (refer to [12] for a more detailed discussion). Such techniques may be used as tools for the identification of the common schema and for the definition of the mapping among the distinct repository which will be defined using the proposed method. Finally, [10] presents an ontology language, an example of formally defined mapping language and a query engine, all of which are based on the description logic *DL-Lite*.

To the best of our knowledge there are no proposals for methods pointing out the rationale and the steps one should follow in order to let a DB and an ontology-based store cooperate under a single system. The most similar work can be found in [2], where the authors present some use cases of integration of ontologies and relational databases. The main difference with our work is that in [2] there is the limitation of accessing data contained in the database read only, while our approach allows for the modification of data. Thus, the proposed method aims to the cooperation of data from repositories of different nature in order to provide the final user fully fledge access to the data, instead of a read-only RDF-based view of data stored in RDBMSs.

3 Cooperation method

The method we propose is aimed at guiding the *separation of data* between relational database (RDB) objects and ontology instances and *defining a suitable mapping* between the two repositories, in order to let them cooperate consistently. To achieve this, we have to address several issues, namely:

- the treatment of consistent references between the two schemas,
- the integration in an existing repository of an external data source,
- the identification of static and changing data,
- the decision on where to store schema instances.

The method defines a *mapping table*, which specifies, for each conceptual object (entity or relationship) in both the re-engineered repositories, where to store the respective instances and whether and how to refer to them. The mapping table should be sufficient to define a formal mapping between the sources, either by modifying the representation of conceptual objects in both sides or defining mapping in both directions e.g. by using mapping languages as D2RQ [5]. As we discuss in the guidelines (see Section 3.2), the choices about separation of instances should be guided by the cost and feasibility of modifications to each of the two knowledge bases. Note that the method does not assume the existence of one or both of the sources: if the ontology or the RDB already exists, its underlying conceptual model is extracted, otherwise the model has to be defined from the system specifications and requirements. The method mainly operates over conceptual representations of the two repositories: intuitively, entities correspond to ontology classes and DB tables, while relationships correspond to ontology properties and attributes in DB tables. Our method assumes that the conceptual models are to be defined in a formalism suitable for representing relevant properties of both sides: we assumed to use graphical models defined following the notations presented in [7,11]. Note that in the case of the RDB, defining such schema roughly corresponds to the extraction of its relational schema.

3.1 Method specification

In this section we present the tasks of our method using the following schema, derived from the definitions in [17,20]: we divide our method in *activities* composed

by *tasks*. In Fig. 2 we show the outline of the activities and tasks of our method: we shortly describe each task and its required *input* and *output* documents.

The method is composed by two distinct activities: the first one is a reverse engineering phase on the available information about DB and ontology, while the second is a forward engineering phase for the definition of the mapping. In the first activity **A1** the method analyzes the available description of the database and ontology (either the conceptual schema, the requirements or directly the sources structure) in order to extract a conceptual representation of the entire system. In **T1** and **T2**, thus, the conceptual schemas of the two sources is retrieved or generated from the descriptions. The two are combined in **T3** by recognizing and merging (possibly automatically [12]) the entities shared by both schemas. This represents the conceptual schema of the integrated system and it is the starting point to the following activity **A2**, in which the decision on the instance separation is taken and the related mappings are defined. In **T4** the entities which instances have to be shared are recognized by the knowledge engineer. In **T5** the decision about the distribution to such instances can take place, thus also defining the direction of mapping for their representation in the other schema. The same is done in **T6** for relationships. The last task **T7** consists in the logical modelling of the mapping, defining the actual objects in both knowledge stores to be mapped with the technical solutions of choice.

This structure is coherent with the one presented in [17,20] for the non-ontological resource re-engineering process: however, our method does not aim at the development of a new ontology but to a re-engineering of both sources in the context of the development of a semantic technology-based application. Moreover, we remark that our method can be applied either when one of the sources is available (by extracting its conceptual schema), when only its conceptual schema is available or when we just have the information (e.g. requirements) to derive the conceptual schema of each part. We also remark that some of the tasks described in the method specification (e.g. **T1** and **T2**) are easily mechanized, in particular when the knowledge bases are already present.

The outputs of our method are two *mapping tables*: the *entity table* and the *properties table*. The *entity table* should describe, for each conceptual entity:

- *Ontology class, DB table:* where its instances are stored,
- *Mapping:* logical mapping on classes and DB tables,
- *ID:* property chosen as identifier,
- *Source:* original source.

The *properties table* should describe, for each relation:

- *Ontology property, DB column:* where its instances are stored,
- *Mapping:* mapping on ontology properties and DB columns,
- *Domain and range:* conceptual entities linked by the property.

We present an execution of our method and an example of the resulting documents in the following sections.

A1. Conceptual Modeling Activity
Reverse engineering on sources to extract complete conceptual schema of the system.
Input: DB and ontology, their conceptual models or requirements
Output: Total conceptual model

T1. DB schema extraction
Extract conceptual schema from DB.
Input: DB, requirements or original DB schema
Output: DB conceptual schema

T2. Ontology schema extraction
Extract conceptual schema from ontology.
Input: Ontology, requirements or original ontology schema
Output: Ontology conceptual schema

T3. Total schema definition
Merge previous schemas to obtain a complete system schema.
Input: DB and ontology schemas
Output: Total conceptual model

A2. Mapping Definition Activity
Forward engineering on extracted model for the definition of mapping tables.
Input: Total conceptual model
Output: Complete mapping tables

T4. Shared schema extraction
Identify conceptual objects to be shared between schemas.
Input: Total conceptual model
Output: Shared conceptual schema

T5. Instances distribution
For every entity, decide where to store its instances.
Input: Shared conceptual schema
Output: Instances table

T6. Relationships distribution
For every relationship, decide where to store its instances.
Input: Shared schema, instances table
Output: Properties table

T7. Logical modeling
Define actual classes and tables to be mapped.
Input: Shared schema, mapping tables
Output: Complete mapping tables

Fig. 2. Method specification

3.2 Guidelines

In the following we suggest some guidelines useful for the application of our method and the definition of the mapping between DB and ontology. First of all, the following guidelines may drive the decision on which of the two schemas refer when storing instances of a conceptual object.

- *Ontology instances:* data can be stored as instance of ontology classes and properties mostly because it is necessary to draw inferences from this data. This can be useful when arranging data in complex taxonomies or meronimies or when it is needed to verify correctness of the data with respect to the ontology logical constraints. Another scenario where this choice is necessary is when one needs to comply with an external (possibly standard) ontology. In general, ontology instances should be treated as fixed and non-changing data, mostly representing "metadata" of the application to be developed.
- *RDB instances:* on the other hand, DB instances should represent the "working data" of the application, that is the data that one expect to be updated

and changed the most. Other reasons to leave out such data from the ontology include the fact that only simple queries (and no inferences) over this data are needed, or the fact that they represent only "administrative" data that is uninteresting to map and publish over the ontology.

Note that this means that the actual *data* would be stored in the DB, while the *metadata* would be stored as ontology instance. Note also that, in both cases, the choice can be affected by where the original instances were stored, the modifiability of the sources or the impact that these modification can cover. Moreover, as it is clear from the method, not every conceptual object in both parts takes part in the mapping: however, by re-engineering the conceptual schema we can decide to move an object from a schema to the other.

Once the choice on where to store instances has been done, the following guidelines suggest how to map and identify these instances in the two directions, so that they are visible in the other schema.

- *Instances in DB, class in ontology:* this partly corresponds to the case treated by mapping tools. From DB to ontology, entity instances can be mapped to individuals of a class naming them, e.g. as `ClassName_ID`. To identify in the DB mapped instances once they are retrieved from the ontology, one can map the ID or primary key of the instance as a `hasID` datatype property value referred to the ontology instance.
- *Instances in ontology, values in DB:* we can suggest different solutions to access or refer to external ontology instances into the DB records. A solution consists in directly using the URI of the referred individual in the DB tuples: however, this solution can be non satisfactory in that one can not check the validity and consistency of the references and can not add information to such individuals in the DB. Another solution is to keep in the DB a table relating the DB tuples to their counterpart individual in the ontology: additional data for the objects (not available in the ontology) can be stored in other columns of such table. A similar solution, but more demanding in terms of updates to the DB schema, consists in using the URI (or a transformation of it) of the ontology instance as the ID of the tuple of their counterpart in the DB. A relaxation of the previous solution consists in defining a transformation from the URI (or other property value) of ontology individuals to the value used as ID in the DB.

4 Example

In the following section we present a simple example for the application of the previously proposed method. More precisely, we present a simplification of the actual integration of the DB schema and ontology mentioned in our motivating scenario. Note that the operations described in the following of the section have been performed manually because of the dimension of the problem. In order to deal with more complex scenarios it will be required to develop tools supporting the tasks described in Section 3.

In the case of our example, we assume to already have the ontology conceptual schema (since basically we are adding a new ontology to an existing DB based system) and to be able to extract the conceptual schema from DB tables. Moreover, we assume (by relaxing the situation of our motivating scenario) that we can freely modify both parts. In our example, ideally, the DB mostly contains the data pertaining to the actual files representing medical images, while the ontology stores the relations between the concepts represented in the subject of images.

Given these premises, in the following we proceed trough the tasks of our method, providing examples for the most relevant produced documents. For simplicity, we only represent the properties and relations of the main entities of our system.

After the first two tasks **T1** and **T2**, given the previous assumptions, we obtain the conceptual schema of the DB and of the ontology, which are shown in Fig. 3 and Fig. 4. In particular, note that their structure is slightly different: e.g. the entity *Origin* (representing the method of acquisition of an image) appears only in the DB schema, while in the ontology schema *Image* is specialized in *MacroImage* and *MicroImage* (actual photographs versus microscopy images) which need to be treated differently in our system. Most notably, only the ontology contains the relations between the entities representing subject properties as in the case of *hasPosition* for *Lesion*.

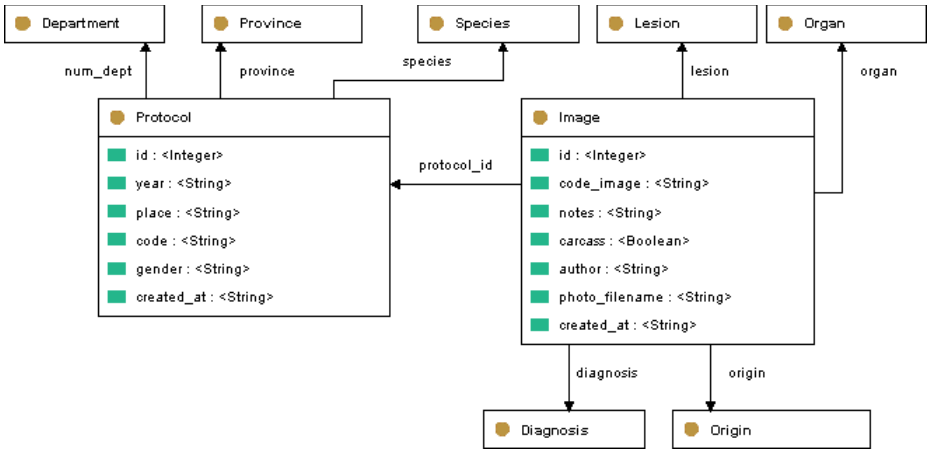

Fig. 3. DB Conceptual schema

In task **T3** we merge the two schemas in the total conceptual schema, considering the shared attributes: for example, note the case of the information about gender, in the DB represented as attribute and in the ontology as object property. We do not show this schema, for space and significance reasons. After obtaining the total schema, we can also begin to define the contents of the entity and property tables, mainly by filling in the names of entities and the properties with their specified domain and range.

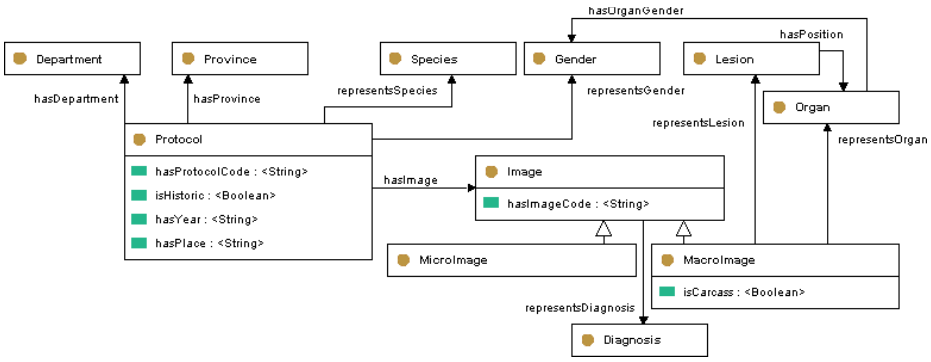

Fig. 4. Ontology Conceptual schema

We can now begin the forward engineering activity: the first task **T4** consists in identifying the shared schema in the total schema, which corresponds in picking out the instances that are not to be mapped, following the given guidelines. For example *Origin* and the attributes as *author* and *notes* only belongs to the DB while *MicroImage* is only to be contained in the ontology. The shared schema obtained in this task is shown in Fig. 5.

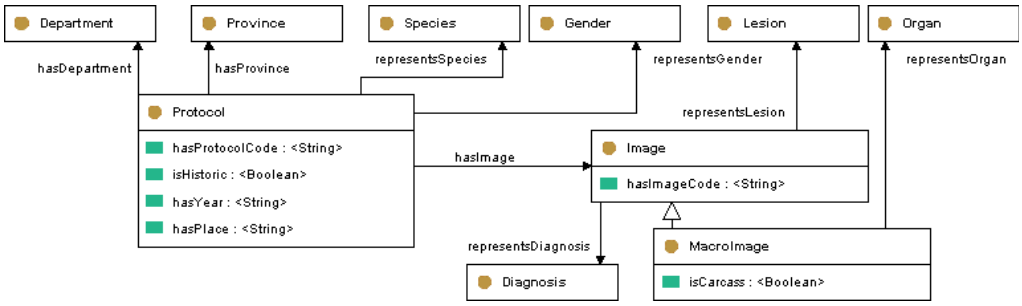

Fig. 5. Shared schema

The next two tasks **T5** and **T6** consist in separating the instances and relations of DB and ontology and thus defining the direction and the choices for the mapping, as suggested in our guidelines. For example, note that since *Protocol* and *Image* represent the data of our system, they are stored in the DB and mapped to their classes in the ontology. On the other hand, the objects actually detailed in the ontology have to be only referred in the DB instances.

In the last task **T7**, the mapping tables are completed with the actual DB tables and columns to be mapped. The final mapping tables for our example are shown in Table 1. Note that the proposed tables only contain relevant parts of the actual mapping tables for our schemas. We remark that the structure and notations used to present our method are simply suggestions for a manual execution of the method and can be replaced or hidden to the user in case of an implementation.

Entity	Mapping	Onto.Class	DB Table	ID	Source
Protocol	DB - O	Protocol	Protocol	Protocol.ID	O, DB
Image	DB - O	Image	Image	Image.ID	O, DB
MacroImage	DB - O	MacroImage	Image	Image.ID	O
MicroImage	O	MicroImage	-	URI	O
Species	O - DB	Species	-	URI	O, DB
Gender	O - DB	Gender	-	URI	O
Organ	O - DB	Organ	-	URI	O, DB
Origin	DB	-	Origin	Origin.ID	DB

Domain	Property	Range	Type	Mapping	DB column
Image (DB)	representsDiagnosis	Diagnosis (O)	object	DB - O	Image.diagnosis
	origin	Origin (DB)	object	DB	Image.origin
	hasImageCode	<string>	datatype	DB - O	Image.code_image
	notes	<string>	datatype	DB	Image.notes
	author	<string>	datatype	DB	Image.author
	...	...	...	...	...
MacroImage (DB)	representsOrgan	Organ (O)	object	DB - O	Image.organ
	representsLesion	Lesion (O)	object	DB - O	Image.lesion
	isCarcass	<boolean>	datatype	DB - O	Image.carcass
Lesion (O)	hasPosition	Organ (O)	object	O	-

Table 1. Entity and Property tables (excerpt)

5 Conclusions

In this paper we presented a method that allows a relational database and an ontology – as deployed in a Semantic Web application – to collaborate towards a fruitful distribution of data between them. We also provided guidelines in order to support the decisions to be taken in the deployment of our method. We defined the presented method motivated by the scenario of a system for the management of bio-medical images. In such project, semantic technologies are used to relate data from a relational database containing information about images to ontologies containing complex metadata classifying them.

The method we proposed in these pages represents a first step towards the definition of a generally applicable re-engineering process: for its further development, it is certainly necessary to refine and evaluate the proposal with experiences on several real-world applications scenarios. Moreover, the proposed guidelines are not thought to constitute a complete best practice, but they want to draw the attention to some relevant aspects of the cooperation and possibly promote discussion about these issues. Another interesting direction for further developments is the study of the automatization possibilities and the effective implementation for the tasks of our method.

Acknowledgments

We would like to thank the IZSLER institute for the support and collaboration.

References

1. Alexe, B., Chiticariu, L., Miller, R.J., Tan, W.C.: Muse: Mapping understanding and design by example. In: ICDE. pp. 10–19. IEEE (2008)

2. Auer, S., Feigenbaum, L., Miranker, D., Fogarolli, A., Sequeda, J.: Use Cases and Requirements for Mapping Relational Databases to RDF. RDB2RDF XG Working Draft, W3C (Jun 2010), http://www.w3.org/TR/rdb2rdf-ucr/
3. Aumueller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and ontology matching with coma++. In: Özcan, F. (ed.) SIGMOD Conference. pp. 906–908. ACM (2005)
4. Barrasa, J., Corcho, O., Gómez-Pérez, A.: R2O, an Extensible and Semantically based Database-to-Ontology Mapping Language. In: Proceedings of SWDB2004. pp. 1069–1070. Springer (2004)
5. Bizer, C.: D2R MAP - A Database to RDF Mapping Language. In: Proceedings of WWW03 (Posters) (2003)
6. Bizer, C., Cyganiak, R.: D2RQ - Lessons Learned. W3C Workshop on RDF Access to Relational Databases (Oct 2007), http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/w3c-d2rq-positionpaper/
7. Brockmans, S., Haase, P.: A Metamodel and UML Profile for Networked Ontologies - A Complete Reference. Tech. rep., Institute AIFB, Universität Karlsruhe, Germany (2006)
8. Handschuh, S., Staab, S., Volz, R.: On deep annotation. In: Proceedings of WWW03. pp. 431–438 (2003)
9. He, B., Patel, M., Zhang, Z., Chang, K.C.C.: Accessing the deep web. Commun. ACM 50(5), 94–101 (2007)
10. Poggi, A., Lembo, D., Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. Data Semantics 10, 133–173 (2008)
11. Presutti, V.: D2.5.1. A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies. NeOn Project Deliverable D2.5.1/v1.2, NeOn (Feb 2008)
12. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. 10(4), 334–350 (2001)
13. Raunich, S., Rahm, E.: Atom: Automatic target-driven ontology merging. In: Abiteboul, S., Böhm, K., Koch, C., Tan, K.L. (eds.) ICDE. pp. 1276–1279. IEEE Computer Society (2011)
14. Sahoo, S., Halb, W., Hellmann, S., Idehen, K., Thibodeau, T., Auer, S., Sequeda, J., Ezzat, A.: A Survey of Current Approaches for Mapping of Relational Databases to RDF. RDB2RDF XG Report, W3C (Jan 2009), http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport.pdf
15. Stojanovic, L., Stojanovic, N., Volz, R.: Migrating data-intensive web sites into the semantic web. In: Proceedings of SAC. pp. 1100–1107. ACM (2002)
16. Stojanovic, N., Stojanovic, L., Volz, R.: A reverse engineering approach for migrating data-intensive web sites to the semantic web. In: Proceedings of IFIP. pp. 141–154 (2002)
17. Suárez-Figueroa, M.C.: D5.4.1. NeOn Methodology for Building Contextualized Ontology Networks. NeOn Project Deliverable D5.4.1/v1.0, NeOn (Feb 2008)
18. Suchanek, F.M., Abiteboul, S., Senellart, P.: Paris: Probabilistic alignment of relations, instances, and schema. CoRR abs/1111.7164 (2011)
19. Sugumaran, V., Storey, V.C.: The role of domain ontologies in database design: An ontology management and conceptual modeling environment. ACM Trans. Database Syst. 31(3), 1064–1094 (2006)
20. Villazón-Terrazas, B.: D2.2.2. Methods and Tools Supporting Re-engineering. NeOn Project Deliverable D2.2.2/v2.0, NeOn (Feb 2009)