Map-On: A web-based editor for visual ontology mapping

Álvaro Sicilia[a], German Nemirovski[b] and Andreas Nolle[b]
[a] *ARC Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Barcelona, Spain*
E-mail: *asicilia@salle.url.edu*
[b]*Business and Computer Science Albstadt-Sigmaringen-University of Applied Sciences Albstadt, Germany*
E-mail: *{nemirovskij, nolle}@hs-albsig.de*

Abstract. This paper presents Map-On, a web-based editor for visual ontology mapping developed at the Architecture, Representation and Computation research group of the La Salle at Ramon Llull University. The Map-On editor provides a graphical environment for ontology mapping creation based on the representation of mapping using an interactive graph layout. Thus, a point-and-click interface simplifies the mapping creation process. Based on the user inputs the editor automatically generates a R2RML document, particularly produces the IRI patterns and the SQL queries. It has been used in real scenarios alleviating the effort of coding R2RML statements which is one of the main barriers for adopting R2RML in research and in the industry communities.

Keywords: Ontology mapping editor, Semantic data integration, OBDA, Mapping visualization, R2RML

1. Introduction

In recent years, the interdisciplinary character of numerous projects and applications led to the increasing need for integration of data related to different knowledge domains and stored in different formats. In this context the community of experts and stakeholders working with heterogeneous data has grown considerably. These communities are usually composed by people with heterogeneous backgrounds skills and goals. On the other hand, thanks to widely spread initiatives such as the open data movement, the quantity and quantity of the available open data is steadily increasing.

The ontology-based data access (OBDA) paradigm can be useful in scenarios where people with different backgrounds and skills want to access heterogeneous data sources. In ODBA settings, the data sources are accessed using a high-level conceptual representation without the need to know how the data sources are organized [1, 2]. The main components of an OBDA system are the ontology, which represents the conceptualization of the data sources' domains, the mappings between the data sources and the ontology, and the query rewriter which receives queries in terms of the ontology to transform them in terms of the data sources. The queries can be processed with reasoning purposes on the ontology making possible to use hidden relations that are not explicitly defined in the data source.

In this context, the development of mappings between the ontology, which presents a generic conceptual view of the data, and the schemas of the integrated data sources is one of the key issues. Efforts for automation of the mapping tasks have been carried out in several approaches [3, 4]. However, due to the complexity in obtaining the proper semantics from the schemas, the manual mapping in OBDA systems is, nowadays, the widely adopted solution in academic and industry communities in spite of its high time consuming and high requirements of human expertise [5, 6]. The specification of mappings between the ontology and the data schemas requires from the experts knowledge of the data schemas to be integrated as well as of the ontology architecture.

Mappings from relational databases to a RDF dataset are mostly implemented using R2RML, a declarative language recommended by the W3C. R2RML is supported by the best practices and OBDA systems for data integration such as Maestro

Studio [7] and systems for data integration [8]. As stated above, creating R2RML mappings requires of advanced skills and expertise in ontology design and formal logic. But also domain experts and data owners, who usually are lacking the mentioned expertise, are involved in setting up OBDA scenarios. However, in these scenarios and for these kinds of users the main barrier is often the lack of a visual representation of the mappings which could help them to close the gaps in the expertise and complete the mapping task.

With this scenario in mind, we have developed Map-On, a graphical environment for ontology mapping which helps different kind of users -domain experts, data owners, and ontology engineers- to create and maintain mappings between a database and a domain ontology using R2RML recommendation. The ontology mapping environment provides visualizations of mappings based on a graph layout and supports automated generation of IRI patterns and SQL queries for the R2RML statements.

The paper is organized as follows. Section 2 gives an overview of existing mapping editors. We briefly introduce the R2RML language in section 3. Afterwards, we describe the main features of Map-On, including its visual representation of ontology mappings, point-and-click interface and automated generation of R2RML statements in Section 4. Section 5 gives a high level overview of the tool architecture. We describe the real-world deployments of Map-On in Section 6, and discuss current limitations and future plans in Section 7.

2. Related tools

Several tools have been developed to assist experts in specifying mappings between the data sources and ontologies. For instance, OntopPro extends Protégé by providing a mapping editor that integrates a SPARQL query interface with the Quest query engine [9]. In spite of that, the editor works with a proprietary mapping format, it can export and import R2RML files. Similar functionalities are included in the mapping editor presented by Sengupta et al. in [10], in this case with native support of R2RML. Generally, the users of these tools are technicians, experts in formal and especially description logic. These tools provide graphic visualization neither for the data source schemas, nor for the domain ontology nor for the mappings. The lack of these features makes these tools not useful for domain experts and data owners, who although being involved in the data integration process, do not have the background in ontology engineering.

A significant help for non-experts in the ontology engineering can be provided by ontology visualization techniques. They help users to inspect, navigate, and verify the ontologies and mappings [11].

However, the first choices in this concern are the dedicated mapping editors like the Karma system. Karma is a mapping editor that provides a graphical user interface for visualizing and editing the mappings. Moreover, Karma can automatically suggest mappings and supports R2RML recommendation [4]. The mappings are displayed using a tree layout for the ontology and a table layout for the database schema. ODEMapster works in a similar way. It also provides with a graphic visualization of the database schema and with the domain ontology using a tree layout expressing the mappings in a proprietary mapping language called R2O [12]. Furthermore, the RBA (R2RML By Assertion) tool also uses a tree layout for displaying the mappings and supports R2RML [13].

The limitation of these tools consists in the exploitation of the tree layout for visualization. Such layout is unable to represent the complete structure of the database schema, ontology and mapping by itself.

Visualizing ontologies and mappings with a graph layout is probably the most natural and most common used. A prominent example is the mapping visualization model presented by Lembo et al. in [14]. In this case, the mappings are presented in a graph layout including three views focused on the mapping, the ontology, and the source. However the use of graph layout in ontology mapping editors is rare.

3. Preliminaries

A R2RML mapping is a set of triples maps that are composed of a logic table which can be defined as a base table, a view (i.e. the result set of a stored query), or a SQL query; a subject map which defines the subject of the RDF triples generated from the logic table; and a set of predicate and object maps which defines the predicates (i.e. roles) and objects (e.g. RDF objects) of the RDF triples. The subject and objects maps describe how the IRIs (Internationalized Resource Identifier) should be generated using the columns specified in the logic table and the elements of a target ontology (i.e. concepts, roles and attributes).

The manual creation of R2RML mappings requires of technical skills in SQL query design and in ontology engineering at the same time. The experts who create the mappings should understand the structure of the database schemas and the target ontology in order to find correspondences between the columns of the relational tables and the ontology entities. Moreover, users have to design SQL queries for the logic tables and the IRI patterns for the subject and object maps.

The triples map illustrated in Figure 1 uses a logic table based on a SQL query for the table *census*. The IRI of the subject map uses the *ID* column of the table and the concept *Building*. The object map is defined with the role *hasAddress* and the column *Address*.

```
<mapping1> a rr:TriplesMap;
 rr:logicalTable [
   rr:sqlQuery "SELECT ID, Address FROM census"
 ];
 rr:subjectMap [
   rr:template "http://example.com/building/{ID}";
   rr:class ex:Building
 ];
 rr:predicateObjectMap [
   rr:predicate ex:hasAdress
   rr:objectMap [rr:column "Address"]
```

Fig. 1. Example of a R2RML mapping.

The main limitation of R2RML languages is that in the research community and in the industry, there are certain problems with the adoption of R2RML. The manual creation of R2RML mappings is a time consuming process, the mappings are syntactically heavy in terms of the R2RML vocabulary (these processes imply to design valid SQL queries, to use the proper terms of the R2RML language, and to select the ontology concepts and properties), and for the experts using the language the steep learning curve is mainly caused by gaining expertise of the language [15].

The mapping editor presented in this paper, supports the creation of triple maps by providing visual representation of mappings and a point-and-click interface facilitating the user to map the columns of the tables to the ontology concepts, roles, and attributes. The IRI patterns and the logic tables are automatically generated by the tool.

4. Map-On features

The main goal of the Map-On tool is to support users to create mappings between a database schema and the existing domain ontology. The tool provides with visualizations of mappings based on a graph layout for the database schema and the ontology, an ontology mapping interface, and the automated generation of IRI patterns and SQL queries for the R2RML statements.

4.1. Mapping visualization

Database schemas and ontologies are usually represented using a graph layout, a recent prominent example is VOWL a visual language for visualizing ontologies as a force-directed graph layout [16]. In graph layout representations of database schemas and ontologies, the node elements (e.g. tables and concepts) are characterized as nodes and the relations (e.g. primary/foreign key constraints and roles) as edges. The mappings between a database schema and an ontology are a set of relations between their elements. That is when, for instance, a column of a relational table is used to define the IRI of a subject map and a concept of the ontology is utilized to define the type of the subject map. In this way, it is intuitive to represent the mappings graphically as edges between columns and concepts (Figure 2).

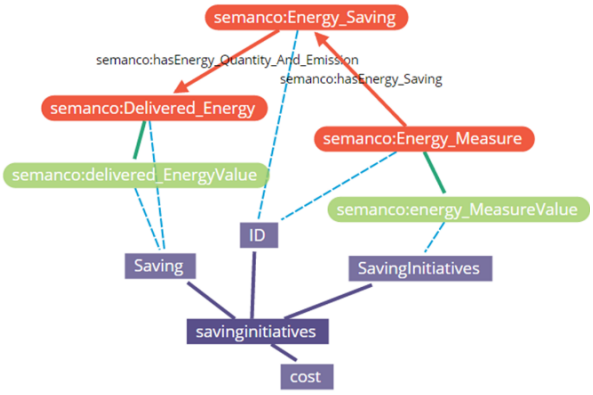

Fig. 2. Visual representation of a mapping.

The Map-On tool represents a table and their columns as purple rectangles connected with a solid purple line. The relationships between tables are displayed as a purple dashed line between foreign key and primary key constraints. The ontology concepts are represented as orange ellipses, the roles as directed solid orange lines, and the attributes as green ellipses. The relations between the elements of the database and the ontology are displayed as dashed blue lines. An example of the visual representation of mappings using a graph layout can be found in Figure 2.

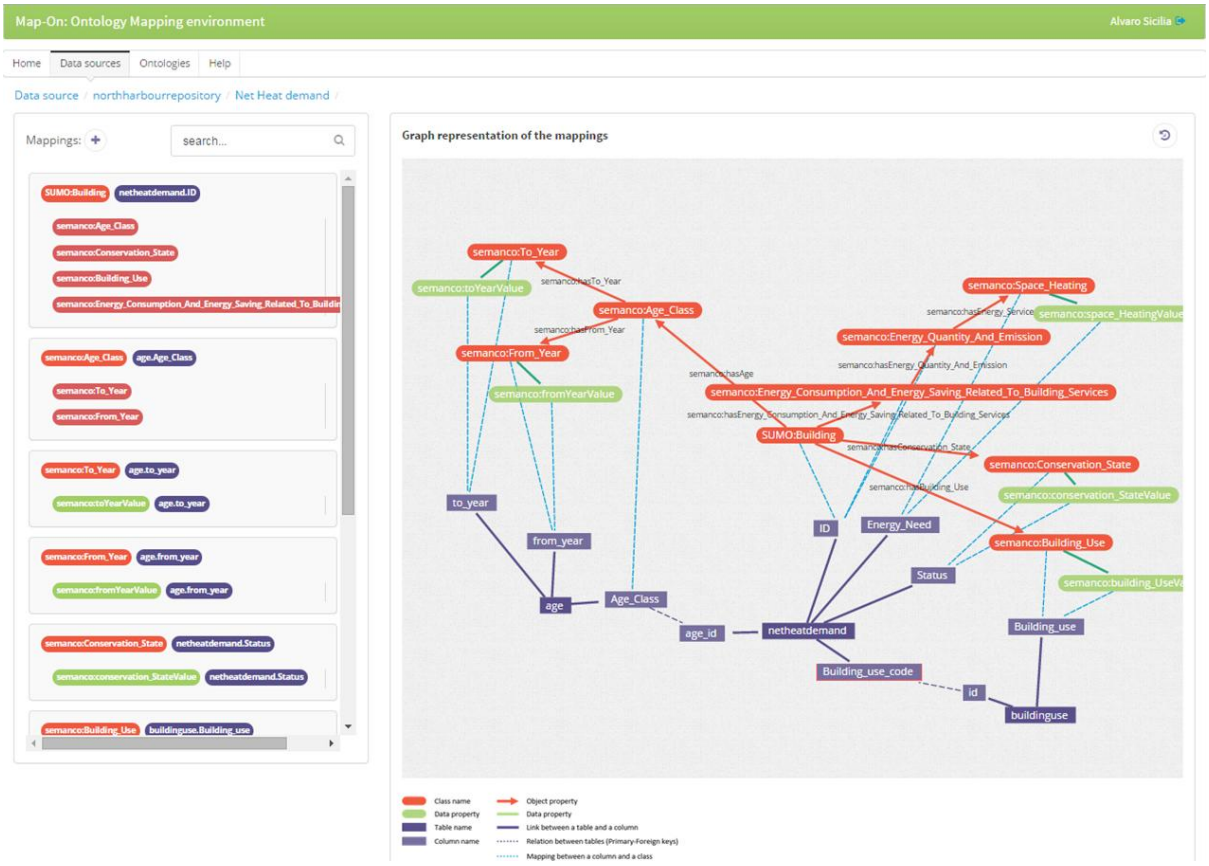

Fig. 3. Map-On main interface. On the left side, a list of mappings is presented available in the 'Net Heat demand' mapping space. On the right side: an interactive graphical visualization of the mappings.

After a mapping has been created, the corresponding layout is automatically generated by placing the nodes in a rectangular grid arrangement. As a next step, the user can modify the position of the nodes by dragging them to a desired position.

4.2. Ontology mapping interface

The Map-On editor provides the user with a high level of assistance in the ontology-database mapping by means of the four following features: point-and-click interface, ontology-driven mapping approach, top-down visualization, and mapping spaces.

The Map-On graphic user interface is based on point-and-click paradigm where most of the user actions are carried out with the cursor. The main benefits of this kind of interfaces are the high comfort and the low getting-started-barrier for those users who are lacking skills in mapping languages such as R2RML.

Furthermore, the interface provides easy access to the elements to be mapped and fosters productivity, as long as complex mapping tasks can be carried out with few user actions.

Map-On implements the ontology-driven approach for editing the mappings. Namely, the user starts with selecting concepts of the ontology and as a second step generates R2RML statements by defining the proper logic tables and IRI patterns. An alternative from the ontology driven approach is the database-driven approach which starts with selection of database elements followed by generation R2RML statements by selecting the proper target ontology elements. As stated in [15], none of these approaches (i.e. ontology-driven and database-driven) is better than the other; however users with a background in database can be more familiar with the ontology-driven approach.

The Map-On interface provides top-down mapping visualizations (Figure 3). In particular, the part of the ontology and database schema (i.e. tables, columns, concepts, roles and attributes) involved in the mapping are visualized in one single representation like a

global view which can be related to a set of triples maps in the final R2RML document. This approach helps the user to comprehend both, database and ontology structures at the same time, and therefore to reduce mapping errors and to simplify their maintenance. Furthermore, a list of mappings is provided in the left side of the interface using the same colour styles as the graph representation. When the cursor overs one mapping on the list, the node of the graph layout corresponding to that mapping is highlighted as well as their neighbourhood nodes. This feature works the other way round, when the cursor overs a node of the graph layout, the corresponding mapping of the list is highlighted.

Moreover, Map-On facilitates the user to define mapping spaces, these are partial views of an entire picture of mappings between an ontology and a database. Such spaces contain a limited set of ontology and database entities and serves to partitioning a complex mapping task into a set of less complex and smaller tasks. As long as the display space available for graph visualization is always limited, this feature is important in scenarios where there are a considerable number of elements involved in the mapping.

4.3. Automated generation of IRI patterns and SQL queries.

The Map-On editor automatically generates IRI patterns and logic tables (i.e. SQL queries) required by the R2RML statements based on the concepts and columns involved in the mappings created by the user. For example, in Figure 3, when a user maps the concept *SUMO:Building* to the column *ID* of the table *netheatdemand* the following IRI and SQL query are generated for defining the subject map:

```
IRI: <base_iri>/building/{\"netheatdemand.ID\"}

SQL: SELECT buildinguse.Building_use, netheat-
demand.Status, age.Age_Class, netheatdemand.ID
FROM buildinguse, age, netheatdemand WHERE
buildinguse.id = netheatde-
mand.Building_use_code AND age.Age_Class =
netheatdemand.age_id
```

The IRI is generated based on patterned URIs solution [17]. This pattern was selected considering that people are able to read it and that it is easily generated from a database where identifiers (i.e., primary keys) are always present. Furthermore, the name of the concept is added to the base IRI. In this way, the problem of generating different individuals with the same identifier but different concept is mitigated. In the above example, the *<base_iri>* variable is common for all the mappings, the *building* comes from the concept of the ontology and *netheatdemand.ID* is the column involved in the mapping.

The logic tables of the triples maps statements are defined as a SQL queries which are automatically generated by the editor. The editor inspects the mappings created by the user for generating a valid SQL query and takes into account all the possible tables and columns involved in the mapping. In the above example, the SQL query has been generated for the mapping between the concepts *SUMO:Building* and the *ID* column of table *netheademand*. In the generation of the SQL query is taken into account the adjacent mappings, for instance the mapping between *semanco:Age_Class* and the column *Age_Class* of the table *age*. The query retrieves data from three different tables (i.e. *buildinguse*, *age* and *netheatdemand*) which are connected by constrains defined in the where clause. Moreover, the columns involved in the mapping are also included in the select clause.

5. Architecture of Map-On

This section presents the general architecture of Map-On including the important aspects of each modules (Figure 4).

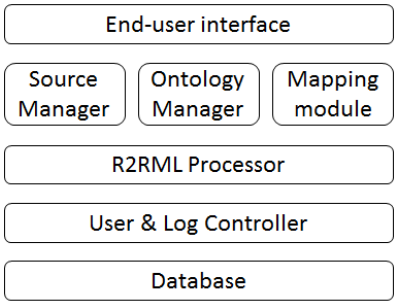

Fig. 4 Map-On basic architecture.

End-user interface. The visualization of the database and the domain ontology by means of a joint graph representation is one of the strong points of the interface together with the creation and modification of mappings using point-and-click paradigm. Thus, users can change the layout by dragging the graph nodes making the visualization more clear. The interface provides suggestion lists of possible concepts roles, and attributes to be used in the mappings. Moreover, the interface comprises pop-ups with tips as an integrated help.

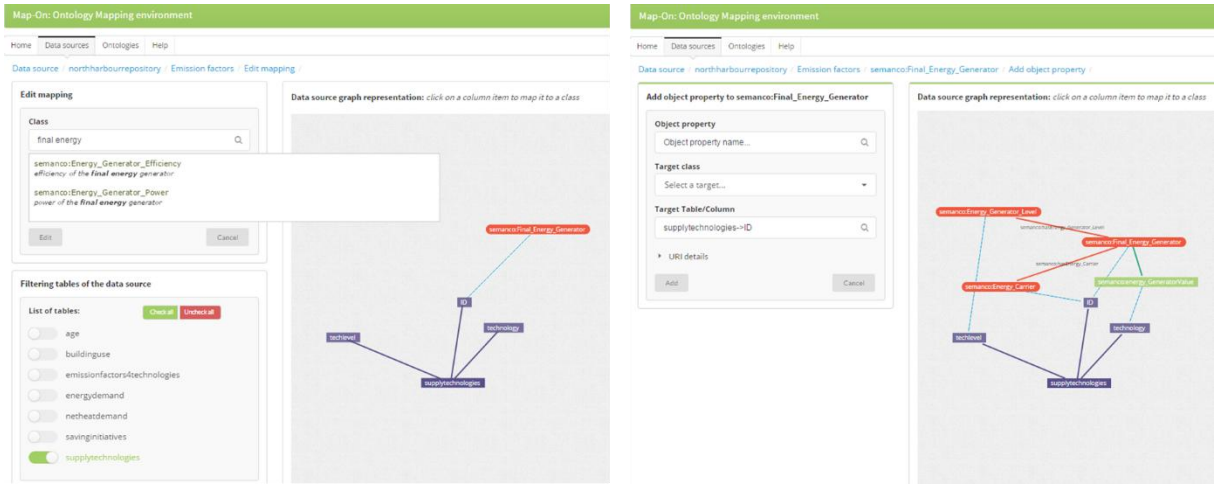

Fig. 5 Map-On interfaces. Left: new mapping creation interface. Right: Object property creation interface.

Source manager. This module provides the methods for loading schemas of the input databases. The schema is provided as an SQL file. Each database has its own mapping spaces with their mappings for producing a R2RML document.

Ontology manager. The domain ontologies are stored in the database of the Map-On tool for increasing the query response performance. The ontology manager provides functionalities for loading an OWL ontology in RDF/XML format and their related ontologies and import. The module also implements

R2RML processor. Based on the mappings created by the user, this module generates a compliant document with the R2RML recommendation. The user can provide custom mappings which are stored in the Map-On database. Furthermore, the custom mappings are attached to the final R2RML document generated by the processor.

User & Log controller. This module tracks the user actions and stores them in the database. The actions are tagged with an identifier depending on where the action takes place. For instance, when a mapping is created between a concept and a column of a table, the action is tagged with the identifier of the mapping space where mapping belongs to. Moreover, the actions are tagged with the identifier of the user who carries out the action.

Database. The different elements provided or created by the user are stored in the database, specifically the structure of the sources (i.e. table names, column names, column types, and foreign/primary keys), the ontology, and the mappings created. Thus, the graph layout configurations personalized by the user are also part of the database.

methods for query specific concepts, roles, and attributes based on a text provided by the user. Finally, the module takes care of the prefixes of the ontology needed for representing the ontology elements by QNames (Qualified names).

Mapping module. This module is responsible for creating and storing the mappings defined by the user. Thus, it manages the mapping spaces. The mapping module implements the methods for the automatic generation of IRI patterns and the SQL queries described in the previous section.

The modules have been developed in PHP using Code Igniter[1], an Open source PHP web application framework. The graph visualizations of the source, the ontology and the mappings have been implemented using the VivaGraphJS[2], a graph drawing library for JavaScript library. The EasyRDF library[3] has been used for parsing the ontology files and the Appmosphere RDF classes (ARC) library[4] for the ontology storage using MySQL engine. The R2RML visualization and text editor use the Codemirror JavaScript library[5] for the turtle syntax highlighting style.

[1] http://codeigniter.com
[2] https://github.com/anvaka/VivaGraphJS
[3] http://www.easyrdf.org
[4] https://github.com/semsol/arc2
[5] http://codemirror.net

6. Ontology mapping process with Map-On editor

Users of the Map-On[6] editor can carry out an ontology mapping process in a user-friendly interface without worrying about dealing with R2RML coding.

The first step of the process is to load the input database and the ontology to be mapped. The ontology and their imports are stored in the same place, this reduces the querying time. For loading the input database an SQL schema file has to be provided together with the base IRI to be used in the IRIs of the subject and object maps of the R2RML statements.

The second step is to define the mapping spaces. Users are free to create any number of mapping spaces, usually parts of ontology that have something in common are mapped in the same space.

The third step is to create the mappings by clicking the plus button on the top-left side of the interface (Figure 3). In the new mapping page, the graph layout contains an empty node representing an ontology concept and the nodes characterizing the structure of the input database. The user can search for an ontology concept by typing input box (Figure 5, left screen). The editor will provide a list of possible concepts based on the input given by the user. Once the user select one concept from the list, the node in orange takes the name of the concept. After that, the user can click a node of the graph representing a column and a mapping –a blue dashed line– will connect the concept and the column nodes. The tables of the database can be filtered in order to reduce the nodes in the graph visualization by clicking the checkboxes on the bottom-left side of the interface (Figure 5, left screen). After these actions the user has stablished a mapping between a concept and a column which corresponds to a subject map statement.

The next step is to further elaborate the mappings by linking concepts by roles and attributes to link values to some objects. The different interfaces are accessible through a dropdown menu which is shown when the cursor hovers the mapping list on the left-hand side of the interface. For an existing mapping, the user can create a role mapping which will corresponds to an object map statement in the R2RML document. The user can search the role by typing in the input box (Figure 5, right screen). The editor will provide a list of possible roles whose domain is the concept already mapped and that match the input text provided by the user. Once the role is selected the graph visualization is updated accordingly and a target concept (i.e. possible ranges of the role) can be selected from the list provided by the editor. Later, the user can click on a column node to map it to the target concept. The process of linking attributes is similar to the one described for the roles.

Finally, when the mappings are created the R2RML document can be visualized using a turtle syntax highlighting library (Figure 6). Furthermore, users can write their own mappings. The R2RML document can be downloaded as a text file.

```
##############################################
# TripleMap for 133: http://semanco-tools.eu/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#Final_Energy_Generator

<mapping1_133> a rr:TriplesMap;
    rr:logicalTable [ rr:sqlQuery "SELECT supplytechnologies.technology, supplytechnologies.techlevel, supplytechnologies.ID FROM supplytechnologies" ];
    rr:subjectMap [ rr:template "http://www.semanco-project.eu/copenhagen/final_energy_generator/{\"supplytechnologies.ID\"}";
            rr:class <http://semanco-tools.eu/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#Final_Energy_Generator>
    ];

    rr:predicateObjectMap [
        rr:predicate    <http://semanco-tools.eu/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#energy_GeneratorValue> ;
        rr:objectMap [ rr:column "supplytechnologies.technology" ]
    ];
    rr:predicateObjectMap [
        rr:predicate    <http://semanco-tools.eu/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#hasEnergy_Generator_Level> ;
        rr:objectMap [ rr:template "http://www.semanco-project.eu/copenhagen/energy_generator_level/{\"supplytechnologies.techlevel\"}" ]
    ];
    rr:predicateObjectMap [
        rr:predicate    <http://semanco-tools.eu/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#hasEnergy_Carrier> ;
        rr:objectMap [ rr:template "http://www.semanco-project.eu/copenhagen/energy_carrier/{\"supplytechnologies.ID\"}" ]
    ];
    .


##############################################
# TripleMap for 134: http://semanco-tools.eu/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#Energy_Generator_Level

<mapping1_134> a rr:TriplesMap;
    rr:logicalTable [ rr:sqlQuery "SELECT supplytechnologies.techlevel FROM supplytechnologies" ];
    rr:subjectMap [ rr:template "http://www.semanco-project.eu/copenhagen/energy_generator_level/{\"supplytechnologies.techlevel\"}";
            rr:class <http://semanco-tools.eu/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#Energy_Generator_Level>
    ];

    rr:predicateObjectMap [
        rr:predicate    <http://semanco-tools.eu/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#energy_Generator_LevelValue> ;
        rr:objectMap [ rr:column "supplytechnologies.techlevel" ]
    ];
    .
```

Fig. 6. An excerpt of the R2RML document generated by Map-On

[6] http://semanco-tools.eu/map-on

7. Experience on a real scenario

In order to demonstrate the usefulness and the feasibility of the Map-On editor, we outline a real scenario in which the editor has been used for creating R2RML documents.

The SEMANCO Integrated Platform[7] provides access to energy related data obtained from multiple sources and domains and generated by different users and applications. In this platform, assessment, simulation and analysis tools interact with the semantically modelled data. The information is visualized in 3D models, diagrams and tables to make it understandable to different user profiles. The platform has been developed within the FP7 SEMANCO[8] project and has been tested in three case studies: Newcastle in UK, Copenhagen in Denmark, and Manresa in Spain.

Each city has different data sources which comprise two kinds of data: a) data regarding building typologies which provides statistical data about their physical properties and energy performance characteristics; and b) urban data which provides contextual information of the areas at city and neighbourhood scale.

The platform makes interoperable the data sources and tools thanks to the semantic energy information system (SEIF) whose main component is the federated query engine ELITE [18] built on top of Quest reasoner [19]. In the core of SEIF lies a global ontology coded in *OWL 2 QL*[9], which is based on the DL-Lite family (we focus here especially on the variant *DL-Lite*$_A$) and is designed for efficient reasoning and query answering tasks in context of OBDA [20].

The Map-On is used in this scenario for mapping the three databases to the global ontology. The R2RML documents generated through the editor are used for accessing the contents of the database thought the SEMANCO ontology using Sesame/Quest instance [19]. Table 1 comprises the figures of the mappings generated for these databases. Specifically, a total number of 221 triples maps are generated with a total of 307 object maps. The Map-On editor is also being used for updating and enhancing mappings during the life-cycle of the platform.

Thanks to the Map-On editor it is possible to create all of those mappings with a reduced effort since no SQL query have to be designed manually. The graphical visualization of mappings supports/helps to understand, evaluate, and correct the mappings.

Table 1

Overview of the use case features

Number of:	Newcastle	Copenhagen	Manresa
Tables	3	7	23
Columns	33	28	118
Constraints	0	5	7
Subject maps	26	30	165
Object maps	38	45	224

8. Conclusions

In this paper we presented Map-On, a graphical environment for ontology mapping which supports different kinds of users to manually establish relations between the elements of a database and a domain ontology in context of an OBDA scenario. The ontology mapping editor has four distinctive features:

- Point-and-click interface for reducing the mapping activities effort,
- Ontology-driven mapping approach, where the mapping process starts from the ontology instead of working with the database.
- Top-down visualization, for representing the whole database schema, ontology structure, and mappings using an interactive and joint graph layout.
- Mapping spaces, where the mappings are freely grouped by the users in order to keep mappings tidy.
- R2RML mappings automatically generated from the actions carried out by the users in the graphical interface.

Map-On has been validated through their application in the data integration process of the SEMANCO project. However, the tool is generic enough to be applied to other OBDA scenarios.

We plan to develop further Map-on with the following features:

- Implementing methods for automatically suggest mappings using the database and the domain ontology as inputs. In a later stage, users, thanks to the ontology mapping interface, would correct and extend the proposed mappings.

[7] www.eecities.com

[8] http://semanco-project.eu/

[9] http://www.w3.org/TR/owl2-profiles/#OWL_2_QL

- Allowing the representation of complex SQL queries be compatible with the current mapping visualization.
- Integrating an existing OBDA system for evaluating the mappings generated by the editor. In this way, users can check in real-time weather the mappings are producing the RDF outputs they expect or not.

Acknowledgments

This work has been supported by the project of SEMANCO which is being carried out with the support of the Seventh Framework Programme "ICT for Energy Systems" 2011-2014, under the grant agreement no. 287534.

References

[1] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini and R. Rosati, Linking Data to Ontologies, *Journal on Data Semantics*, X: 133–173, 2008.

[2] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodríguez-Muro, R. Rosati, M. Ruzzi and D.F. Savo, The Mastro system for ontology-based data access, *Semantic Web*, 2(1): 43–53, 2011.

[3] C. Pinkel, C. Binnig, E. Kharlamov, P. Haase, IncMap: Pay-as-you-go Matching of Relational Schemata to OWL Ontologies, *In Proc. of the 8th International Workshop on Ontology Matching*, volume 1111 of CEUR, ceur-ws.org, pages 37–48, 2013.

[4] C.A. Knoblock, P.A. Szekely, J.L. Ambite, A. Goel, S. Gupta, K. Lerman, M. Muslea, M. Taheriyan and P. Mallick, Semi-automatically mapping structured sources into the semantic web, *In Proc. of the 9th Extended Semantic Web Conference*, volume 7295, pages 375–390, 2012.

[5] D.F. Savo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodríguez-Muro, V. Romagnoli, M. Ruzzi, and G. Stella. MASTRO at work: Experiences on ontology-based data access. *In Proc. of DL 2010*, volume 573 of CEUR, ceur-ws.org, pages 20–31, 2010.

[6] N. Antonioli, F. Castanò, C. Civili, S. Coletta, S. Grossi, D. Lembo, M. Lenzerini, A. Poggi, D.F. Savo and E. Virardi, Ontology-Based Data Access: The Experience at the Italian Department of Treasury. *In Proc. of the Industrial Track of the 25th Int. Conf. on Advanced Information Systems Engineering 2013*, volume 1017 of CEUR, ceur-ws.org, pages 9–16, 2013.

[7] C. Civili, M. Console, G. De Giacomo, D. Lembo, M. Lenzerini, L. Lepore, R. Mancini, A. Poggi, R. Rosati, M. Ruzzi, V. Santarelli and D.F. Savo, MASTRO STUDIO: Managing ontology-based data access applications, *In Proc. of the VLDB Endowment*, volume 6, number 12, pages 1314–1317, 2013.

[8] P. Bellini, M. Benigni, R. Billero, P. Nesi and N. Rauch, Km4City ontology building vs data harvesting and cleaning for smart-city services, *Journal of Visual Languages & Computing*, 25(6):827–839, 2014.

[9] M. Rodríguez-Muro and D. Calvanese, -ontop- framework (2012), http://obda.inf.unibz.it/protege-plugin/ (accessed January 16, 2015).

[10] K. Sengupta, P. Haase, M. Schmidt and P. Hitzler, Editing R2RML Mappings Made Easy, *In Proc. Of the International Semantic Web Conference ISWC*, pages 101–104, 2013.

[11] M. Lanzenberger, J. Sampson and M. Rester, Visualization in ontology tools. *In Proc. of the International Conference on Complex, Intelligent and Software Intensive Systems*, pages 705–711, 2009.

[12] F. Priyatna, B. Villazón-Terrazas, J. Barrasa and J. Schulte, ODEMapster (2011) http://neon-toolkit.org/wiki/ODEMapster (accessed January 16, 2015).

[13] L.E.T. Neto, V.M.P. Vidal, M.A. Casanova and J.M. Monteiro, R2RML by assertion: A semiautomatic tool for generating customised R2RML mappings, *In Proc. of the 9th Extended Semantic Web Conference*, pages 248–252, 2013.

[14] D. Lembo, R. Rosati, M. Ruzzi, D.F. Savo and E. Tocci, Visualization and Management of Mappings in Ontology-based Data Access, *In Informal Proceedings of the 27th International Workshop on Description Logics*, volume 1193 of CEUR, ceur-ws.org, pages 595–607, 2014.

[15] C. Pinkel, C. Binnig, P. Haase, C. Martin, K. Sengupta and J. Trame, How to Best Find a Partner? An Evaluation of Editing Approaches to Construct R2RML Mappings, In Proceedings of ESWC, *In Proc. of the Extended Semantic Web Conference*, pages: 675–690, 2014.

[16] S. Lohmann, S. Negru, F, Haag and T. Ertl, VOWL 2: User-Oriented Visualization of Ontologies, *In Proc. of the 19th International Conference on Knowledge Engineering and Knowledge Management*, Springer, pages: 266–281, 2014.

[17] L. Dodds and I. Davis. Linked Data Patterns, A pattern catalogue for modelling, publishing, and consuming Linked Data, http://patterns.dataincubator.org/book/linked-data-patterns.pdf (accessed January 16, 2015).

[18] A. Nolle and G. Nemirovski, ELITE: An Entailment-Based Federated Query Engine for Complete and Transparent Semantic Data Integration, *In Informal Proceedings of the 26th International Workshop on Description Logics*, volume 1014 of CEUR, ceur-ws.org, pages 854–867, 2013.

[19] M. Rodrıguez-Muro and D. Calvanese, High performance query answering over DL-Lite ontologies, In Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning, pages: 308–318, 2012.

[20] G. Nemirovski, A. Nolle, Á. Sicilia, I. Ballarini and V. Corrado, Data integration driven ontology design, case study smart city, *In Proc. of the 3rd International Conference on Web Intelligence, Mining and Semantics*, pages 1–10, 2013.