Integrating data into an OWL Knowledge Base via the DBOM Protégé plug-in

Raphaël Squelbut, Olivier Curé
Université de Marne-la-Vallée, Laboratoire ISIS
5, boulevard Descartes
Marne-la-Vallée 77454 France
Email: (ocure,rsquelbu)@univ-mlv.fr

Abstract

This paper presents a Protégé plug-in which enables end-users to design and instantiate an OWL knowledge base from multiple existing relational databases. This approach is inspired by data integration and exchange solutions and presents some functionalities which facilitate the development and maintenance of Semantic Web applications from frequently updated and domain-concerned databases.

1 Introduction

The omnipresence of information technologies is leading to the increase of available data sources. This situation motivates many application designers to access and combine several data sources within a unique solution. The approach we have adopted to tackle this problem consists in merging data contained in the sources into one single, materialized model. The main motivation of this solution is that data sources may be frequently updated and not always accessible at application run-time. In such a context, a data integration approach [2], with a virtualized target, is not adapted because replying to queries will not be possible as data sources are not all accessible on-demand.

A peculiarity of our approach is characterized by the target model we have adopted : a Description Logics (DL) [6] compliant with the Semantic Web, namely OWL DL. It is well-understood that the success of the next generation web partly depends on the availability of efficient ontological engineering tools. The implemented framework proposes to process the following set of ontological engineering's tasks : (i) creation of an ontology schema from the schemata of multiple relational databases (DBs), (ii) instantiation of the knowledge base (KB) with tuples from the sources.

In the context of practical and expressive ontologies, these tasks are considered complex, time-consuming and financially expensive because they require a collaboration between knowledge engineers and domain experts. Thus approaches aiming to facilitate the design of ontologies may be valuable for organizations willing to implement Semantic Web applications.

The system we have developed, named DBOM (DataBase Ontology Mapping), tackles very expressive ontologies, those enabling the expression of general logical constraints [3]. This choice is motivated by our need to semantically enrich the data contained in relational DBs and fulfils a need to implement practical and efficient inference enabled services which are based on application-dependent ontologies. We believe that DBOM's features may encourage the design of ontologies based on practical databases exploited in the "Deep Web" and thus accelerate the adoption of the Semantic Web.

The first application implemented using DBOM's functionalities is a medical informatics web application named XIMSA (eXtented Interactive Multimedia System for Automedication) [1]. In this system, it is essential to integrate several data sources, mostly related to drugs (the French Health Products Safety Agency (AFSSAPS) drug database, ATC/DDD system, etc.), in order to design a KB which enables inferences on drugs and symptoms related to self-medication.

A first non graphical version of DBOM was limited to processing XML maping files in order to create a DL TBox and instantiate the DL ABox. In the context of our medical informatics application,

mapping files are usually created and maintained by health care professionals. These tasks, usually performed within text editors, are considered error-prone and painful for end-users. Thus we aimed to develop a graphical user interface via a Protégé plug-in. Additionally, with this approach, many new functionalities, not available in the previous DBOM implementation, are provided and help the mapping designers during their tasks.

The remainder of this article is organized as follows. Section 2 presents the available solution of DBOM and its principle. Our plugin and its features are described in Section 3. Section 4 provides a discussion and presents some future work on this plug-in.

2 The DBOM Framework

In DBOM, the design of an ontology is created from the DB conceptual schema in a semi-automatic way. End-users familiar with the DB schema are responsible for the design of the mapping file. Still all the other operations are processed automatically : creation of the TBox, instantiation and maintenance of the ABox.

The DBOM system is supported by a migration system which is a set of formulas linking a set of source schemas of DBs and the target ontology schema formalized in OWL DL. The DBOM system is supported by a migration system $\mathcal{MI}$ which is a triple $(\mathcal{S}, \mathcal{O}, \mathcal{M})$, where :

- $\mathcal{S}$ is a set of source schemas of relational DBs.
- $\mathcal{O}$ is the (target) ontology schema formalized in OWL DL.
- $\mathcal{M}$ is a set of formulas of a language $\mathcal{L}_{\mathcal{M}}$ over $\mathcal{S}$ and $\mathcal{O}$.

The definition of $\mathcal{MI}$ emphasizes relations with data exchange [5] and data integration [2] systems. We now contrast the DBOM approach with the comparison of data exchange and integration provided in [4] :

- as in both data exchange and integration, the source schemas are given and the mapping is a set of formulas constructed by a human expert.
- as in data integration, the ontology (target) schema is a reconciliation of the sources and is constructed from the processing of the source schemas given a mapping.
- as in data exchange, the target instances are materialized, while they are virtualized in the case of data integration.

In this work, we consider that data stored at the sources are always locally consistent, meaning that they respect their set of integrity constraints. Anyhow, the integration of these autonomous and consistent sources may result in an inconsistent KB. This is due to violations of integrity constraints specified on the target schema.

In the related domain of data integration, two approaches are proposed to deal with inconsistent data : (i) a procedural approach which is based on domain-specific transformation and cleaning, (ii) a declarative approach. In this last approach, information integration semantics given in terms of *repairs* is usually proposed to solve inconsistency.

Our method to deal with inconsistent data adopts a declarative approach which exploits information on view preferences. In this solution, we consider that all sources do not have the same level of reliability and we enable the mapping designer to set confidence values over the views of the mapping. Using these information at mapping processing time, there is a motivation to prefer values coming from a view wrt data retrieved from another view.

The structure of the mapping imposes to define (i) a prolog which contains the namespaces used in the target ontology schema (declared in Protégé's *Metadata* tab) and bindings to the database sources, (ii) the body of the mapping which contains concrete members and their associated SQL queries.

We refer to "members" of the mapping as the set of concepts, denoting sets of objects, and object properties, denoting binary relations between objects. We make the distinction between concrete and

abstract members. The comprehension of concrete and abstract members is relatively straightforward as it is equivalent to the assumption made in Object-Oriented Programming. Thus instances (individuals) can be created for a concrete concept and a concrete object property assertions relate two existing individuals. Abstract members can not be instanciated and they aim to design a hierarchy of members where final (leafs in a tree representation) members should be concrete. In the context of the DBOM Protégé plug-in, the abstract members are defined in the *OWL Classes* and *Properties* tabs while the concrete members are declared and defined in the *DBOM* tab. The creation of concrete classes and properties are made with SELECT data manipulation operations of the SQL language.

3 The DBOM plug-in

The DBOM plug-in proposes to load DBs in the DBOM widget tab (fig.1 point 1) and represents the related schemas using a tree shape (fig.1 point 2, this area also proposes a view of the ontology tree). Thus all informations (tables, attributes and their types, primary keys, etc..) on the DB are efficiently represented in the main Protégé window.

The declaration of concrete members is processed in the following order

- choose the type of the member to create (concept or object property in area 3 of fig.1),
- optionally define a super arbitrary (meaning that this can be either concrete or abstract) member for this member. In the case of an object property, the end-user has to declare the arbitrary concepts related to the domain and range of this role (area 4 of fig.1),
- type an SQL query to be associated to this member (area 6). Several queries (views), usually defined over different sources, can be associated to a given member with the button of area 5. Each view can be accessed and maintained with the tab of area 5.
- optinally define a confidence value associated with this member's view (area 7). Intuitively, confidences on the views of a given member define a partial order on the views. This order serves to instantiate consistently the DL ABox at mapping processing time.
- bind to each elements in the SELECT clause of this query a previously defined datatype property (area 8 of fig.1).

Interactions between Protégé tabs enable to envision and enrich the TBox. Once the end-user is done with the mapping, she has the opportunity to save the TBox and the mapping file. The mapping instance can then be processed to instantiate the DL ABox.

4 Discussion

In this paper, we have introduced a solution to enable the migration of data stored in multiple data sources to a DL knowledge base. Because data from the sources may not be retrieved and shared on-demand at query time, we have opted for a materialization of the migrated data. In order to deal with possible cases of inconsistencies, we support the setting of confidence values over the views of the mapping. Based on a *global-as-view* (GAV) approach, our solution uses a notion of possible answers, corresponding to credulous entailment, in order to instantiate the ABox.

In order to ease the creation of mapping files, the DBOM Protégé plug-in has been implemented. Although additional tests still need to be conducted with end-users, we are already very positive about the first results. The integration of our approach in the Protégé faciliates and accelerates the design of mappings for end-users comfortable with this ontology editor and knowledge-base framework (creation of abstract members, axiomatizations, determine the OWL sublanguage, etc.).

The creation of this plug-in opened new perspectives with the possibility to apply DBOM functionalities to existing ontologies. This approach was not possible with the previous stand alone DBOM implementation : the mapping was responsible for the creation from scratch of the TBox.

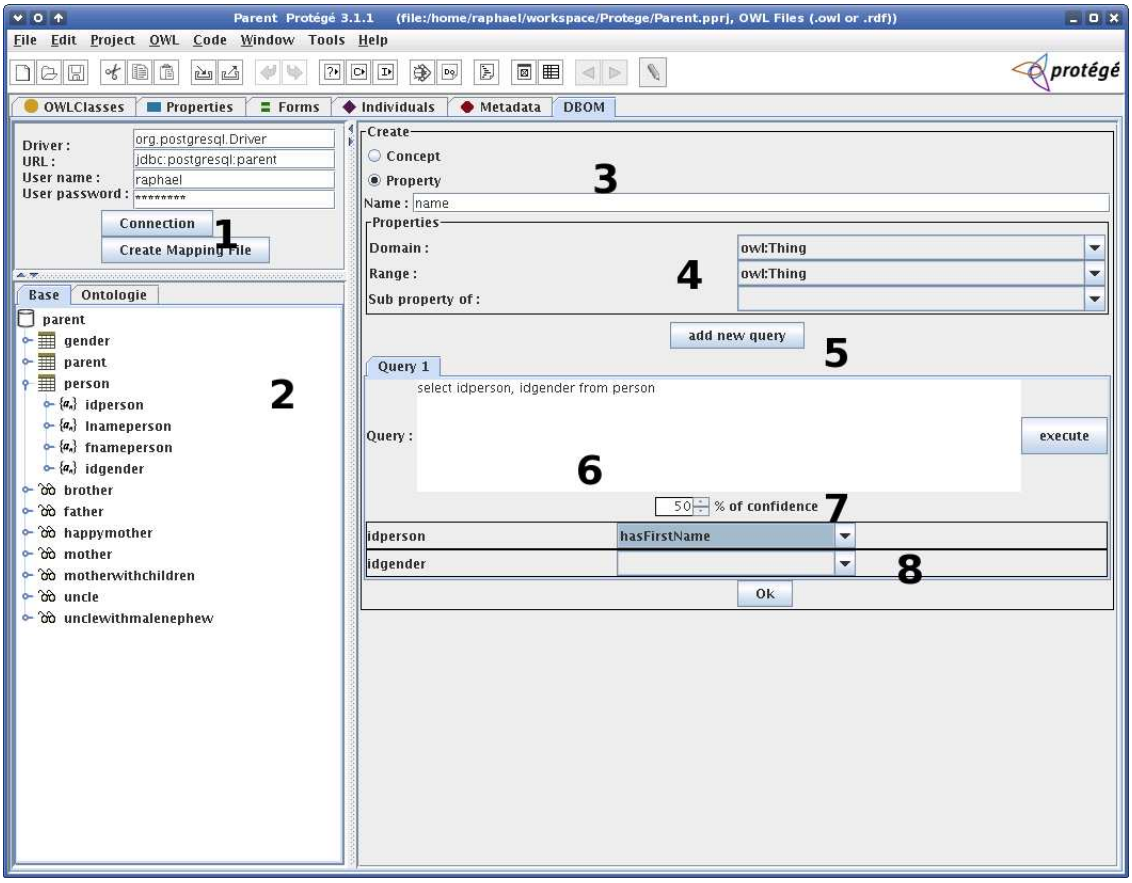

Figure 1: DBOM widget tab

In order to enhance the user-friendliness of the plug-in, we are currently implementing a Query By Example (QBE) solution to design queries associated with concrete members. This solution will be based on interactions with the DB trees of area 2 (fig.1) during the definition of members' views.

References

[1] Curé, O. : *XIMSA : eXtended Interactive Multimedia System for Auto-medication* Proceedings of IEEE Computer-Based Medical Systems (CBMS) 2004. pp 570-575

[2] M. Lenzerini : *Data integration : a theoretical perspective*. Proceedings of ACM Symposium on Principles of Database Systems (PODS), pp 233-246, 2002.

[3] Gómez-Pérez, A., Fernández-López, M., Corcho, O. *Ontological Engineering*. Springer-Verlag. November 2003.

[4] Fagin, R., Kolaitis, P., Miller, R., Popa, L. : *Data exchange : semantics and query answering* Proceedings of ICDT 2003. LNCS 2572. pp 207-224.

[5] P. Kolaitis : *Schema mappings, data exchange, and metadata management* Proceedings of ACM Symposium on Principles of Database Systems (PODS), pp 61-75, 2005.

[6] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. : *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press (2003).