Realizing Ontology Based Data Access: A Plug-in for Protégé

Mariano Rodriguez-Muro, Lina Lubyte, Diego Calvanese
KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano
Piazza Domenicani 3, Bolzano 39100, Italy
`rodriguez|lubyte|calvanese@inf.unibz.it`

Abstract— **In Ontology-Based Data Access (OBDA), the aim is to use an ontology to mediate the access to data sources. We present a plug-in for the standard ontology editor Protégé that allows users to model ontologies with mappings to data sources in order to perform OBDA. We argue that our plug-in, together with an OBDA-Enabled reasoner, allows users to build, test, and deploy OBDA Systems in academic or industrial settings.**

I. Introduction

With a review of recent publications in Databases, Semantic Web, or E-Science journals, one can notice the high frequency of keywords like *Ontology Based Data Integration*, *Semantic Access to Information*, *Ontology Based Data Management*, *Data Integration and incomplete information*. This is a witness of the desire of a sector of the research and industrial community to go in the direction of applications that provide semantically driven access to data, or as this sometimes is called, Ontology Based Data Access (OBDA) [1]. More precisely, in OBDA, the aim is to use an ontology to mediate the access to data. The added value of OBDA, w.r.t. accessing a data source directly, is, on the one hand, that the ontology provides a semantic account of the information stored in the data source. On the other hand, the answer to user queries may be enriched by exploiting the constraints expressed by the ontology, thus overcoming incompleteness that may be present in the data.

A lot of work has been invested in the last years to realize the required technologies. Efforts have been put in defining the appropriate language for the semantic layer, defining the structure and language of the *mappings* used to link the data and the semantic layer, studying the complexity of offering a set of useful services such as query answering, database schema extraction, inconsistency management, etc. Prototype and industrial level systems have been and are being built within academic and industrial research labs, including those of major database players such as IBM [2].

However, as pointed out in [3] in the context of data integration, the gap between theory and practice is still wide. The systems which have been made available to the eager community of early adopters do not offer a cohesive structure. They are either based on different assumptions, rely on different user interaction mechanism or provide limited functionality. As a result, when brought together, sometimes in very rough ways, their interaction turns out to be poor.

The application presented here, the OBDA Plug-in for Protégé is designed to ease the problems noted by Haas, in the OBDA context. We argue that the combination of the standard ontology editing tool Protégé [4], the OBDA Plug-in and an OBDA-Enabled Reasoner, will result in a tool capable of managing the development process of OBDA Systems from end to end.

The rest of the paper is structured as follows: In Section II, we present a quick description of the structure of OBDA Systems and the functionality offered by our plug-in. In Section III, we present two scenarios demonstrating how the OBDA plug-in and an OBDA-Enabled Reasoner can be used to build an OBDA System. In Section IV, we conclude with a brief discussion of the features that will be presented during the demonstration session of the workshop.

II. OBDA: Systems, Reasoners and Plug-in

As argued before, the problems pointed out by Haas for data integration are already present in the case of OBDA. OBDA-Enabled reasoners have already appeared [5], [6] and are being integrated in OBDA Systems. Due to a lack of standardization in the interaction with such reasoners, complicated techniques have to be used to bring all the system components together. Therefore, our objective is to provide a platform based on widely accepted standards that eases access to OBDA-Enabled reasoners, and hence facilitates the construction of OBDA Systems that integrate such reasoners.

To explain how we have targeted the above objective, we first overview the structure of an OBDA System. OBDA Systems have a common structure (see Figure 1): (*i*) They have a semantic layer which is in the form of an Ontology expressed in some ontology language. (*ii*) They allow one to interact with one or more data sources, possibly of different kinds. (*iii*) They manage a set of *mappings* expressing correspondences between data in the data layer and objects in the semantic layer. (*iv*) They allow the user to pose queries over the elements of the ontology. The role of an OBDA-Enabled reasoner in an OBDA System is then to answer the users queries taking into account the ontology, the mappings, and the data in the source(s). Note that, all the components that the reasoner uses are already configured, i.e., the user built an

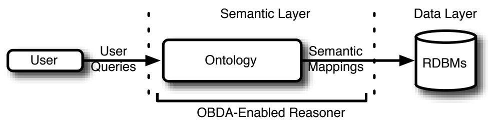


Fig. 1. Prototypical OBDA System architecture

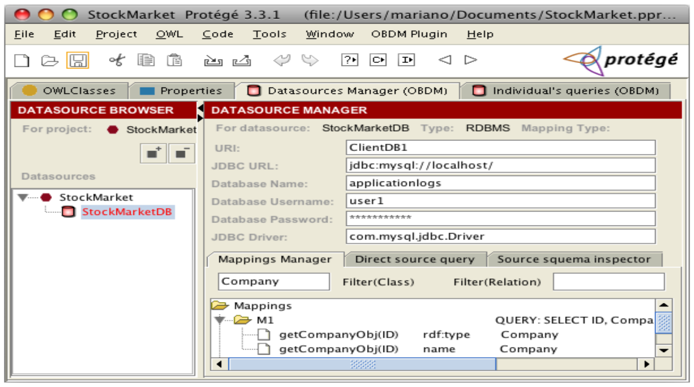
ontology for his application domain, modeled the mappings to the data, etc.

This is where the OBDA Plug-in comes in. Within the plug-in, we offer tools required to model and verify the components of an OBDA System. We do so by extending Protégé with OBDA related operations that allow us to take advantage of the ontology editing functionalities. To do so we had to identify and take into account the differences among OBDA-Enabled reasoners. Such reasoners can differ in many aspects, for example, the specific kind of mappings they allow, the language used to express these mappings, the number and kind of data sources they are able to handle, the specific ontology language used to express the ontology, the language provided to express the queries, etc. The modeling facilities that we provide should be able to adapt to as many of the mentioned differences as possible. Fortunately, we can rely on previous work describing general frameworks that capture most of the differences. With respect to the ontology language, since we are committing to Protégé, the supported modeling language is OWL-DL [7], which is a language that seems to adapt to the requirements of the majority of users. As for the mappings component, we can rely on the vast literature on data integration (see, e.g., [8]), where a mapping assertion relates a query over the ontology to a query over the data sources (GLAV mappings). With respect to the user query language, we support the class of Union of Conjunctive Queries (UCQs) over the ontology. With respect to the type of data sources, we limit the scope of the current release of the plug-in to only support RDBMS data sources. Finally, communication with the OBDA-Enabled Reasoner must be assured in order to provide the reasoner with the modeled components. To address this point, we implement in the OBDA Plug-in the OBDA related extensions to the DIG protocol [9], assuring that any OBDA-Enabled Reasoner will be able to communicate with Protégé and the OBDA Plug-in by also implementing the OBDA extensions to the DIG protocol.

Many aspects of these components cannot be detailed here for lack of space. But they will be dealt with during the demo session, where we will give a detailed description of the plug-in and the interaction between it and OBDA-Enabled reasoners. It is worth mentioning that, apart from the facilities to model the main components of OBDA Systems, we have incorporated in the plug-in tools and features that allow users to do modeling work for their everyday needs, and not just toy modeling. For example, data source inspection facilities, query management features (history and archiving), query result import (e.g., as csv tables, or as OWL assertions), mapping enabling/disabling features (global and per query), among others.

To exemplify the use of the OBDA Plug-in, we now present two scenarios in the context of an industrial application. For these scenarios and for the demo session we use the DIG Server for QuOnto, an OBDA-Enabled reasoner developed at *La Sapienza* University of Rome in collaboration with the KRDB Research Centre within the QuOnto[1] Project. Both, the OBDA Plug-in and the DIG Server for QuOnto will be released for public use in early 2008 through the project's web-

[1]http://www.dis.uniroma1.it/~quonto/


StockMarket Protégé 3.3.1 (file:/Users/mariano/Documents/StockMarket.ppr…
File Edit Project OWL Code Tools Window OBDM Plugin Help
protégé
OWLClasses Properties Datasources Manager (OBDM) Individual's queries (OBDM)
DATASOURCE BROWSER
DATASOURCE MANAGER
Mappings Manager Direct source query Source squema inspector

For project: StockMarket

Datasources
StockMarket
StockMarketDB

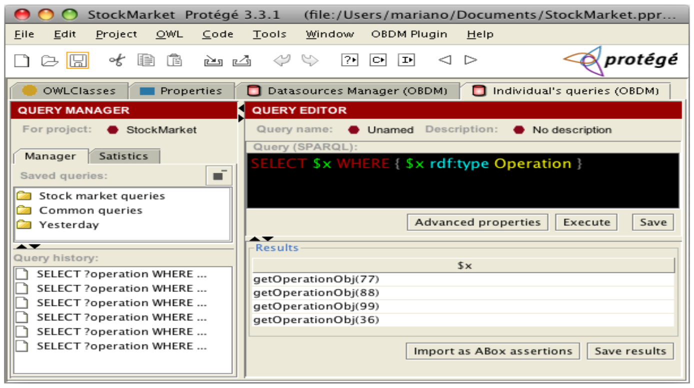
For datasource: StockMarketDB Type: RDBMS Mapping Type:

URI: ClientDB1
JDBC URL: jdbc:mysql://localhost/
Database Name: applicationlogs
Database Username: user1
Database Password: ***********
JDBC Driver: com.mysql.jdbc.Driver

Company Filter(Class) Filter(Relation)

```
Mappings
  M1                                QUERY: SELECT ID, Compa
    getCompanyObj(ID)   rdf:type    Company
    getCompanyObj(ID)   name        Company
```

Fig. 2. Source Tab


StockMarket Protégé 3.3.1 (file:/Users/mariano/Documents/StockMarket.ppr…
File Edit Project OWL Code Tools Window OBDM Plugin Help
protégé
OWLClasses Properties Datasources Manager (OBDM) Individual's queries (OBDM)
QUERY MANAGER
QUERY EDITOR
Manager Satistics
Advanced properties Execute Save
Import as ABox assertions Save results

For project: StockMarket

Saved queries:
Stock market queries
Common queries
Yesterday

Query history:
SELECT ?operation WHERE ...
SELECT ?operation WHERE ...
SELECT ?operation WHERE ...
SELECT ?operation WHERE ...
SELECT ?operation WHERE ...
SELECT ?operation WHERE ...

Query name: Unamed Description: No description

Query (SPARQL):

```
SELECT $x WHERE { $x rdf:type Operation }
```

Results

$x
getOperationObj(77)
getOperationObj(88)
getOperationObj(99)
getOperationObj(36)

Fig. 3. Query Tab

site[2]. Currently, QuOnto is the only DL Reasoner specifically built for OBDA operations, which incorporates facilities to specify data sources and mappings and takes these into account when reasoning. It implements traditional reasoning services (subsumption, satisfiability checking, etc.) as well as answering UCQs for the description logic *DL-Lite*$_A$ [1]. *DL-Lite*$_A$ is a fragment of OWL-DL designed to maximize expressivity while keeping reasoning algorithms tractable. Specifically, reasoning in *DL-Lite*$_A$ is LogSpace in data complexity, as efficient as query answering in relational databases. Therefore, instance level reasoning in *DL-Lite*$_A$ is more efficient than in more complex DLs such as OWL-DL.

III. Demonstration Scenario

Consider the situation of European Union's (EU) financial institutions faced with the introduction of an EU harmonization directive, such as the *Markets in Financial Instruments*

[2]http://www.inf.unibz.it/~rodriguez/OBDA/

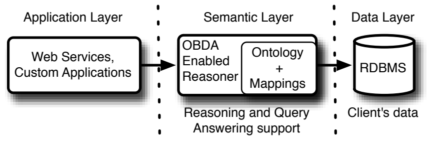


Fig. 4. Consulting company's IT framework

\$x - (a)
getOperationObj(77)
getOperationObj(88)

\$x - (b)
getOperationObj(77)
getOperationObj(88)
getOperationObj(99)

Fig. 5. Results for query q1, a) using mapping m2 b) using mapping m2b

Directive (MiFID) [10]. With its appearance in 2004, the directive obliges financial market institutions to modify their business processes to comply with the regulations stipulated in it. The regulations include legal and procedural changes which potentially involve modifications to the IT infrastructure of the affected financial institutions. The deadline for the adoption of these regulations was November, 2007. To date, still many institutions are working to modify their operational processes and IT support. In this context, consulting companies assist financial market institutions in the task of modifying their operational processes to comply with MiFID. We place our demonstration scenarios within the services that one of these consulting companies provides. In the scenarios we focus on the MiFID *transparency requirement*, which obliges EU financial institutions that trade stocks to publish the operations they perform. The methods they should use include software publishing, e.g., web-services, aggregation sites, etc. Having to comply with this requirement in a limited time frame, financial institutions rely on the expertise of these consultants to extend their IT infrastructure.

The following scenarios present a consulting company which uses OBDA technology to save time and costs in the development of such IT requirements. It do so by modeling its software solution as an OBDA System; that is, incorporating a semantic layer between the data layer of the client and the application layer it provides, as shown in Figure 4. The semantic layer is composed of an OBDA-Enabled Reasoner, an ontology describing the domain of a MiFID compliant financial institution, and mappings linking the data of the client to the MiFID Ontology. The consultant uses the OBDA Plug-in to model the mappings between the MiFID Ontology and the data layer, aspect of the system which changes between different clients. Once this has been established and verified, the consultant deploys the structure presented in Figure 4 into the client's IT infrastructure. In this way, the consultant can adapt the same framework to different clients.

We now describe the mentioned scenarios. In each scenario, the clients of the consultant are Stock Exchanges, i.e., a financial markets' institutions that provides services for stock trading. It is important to note that the clients in Scenario 1 and Scenario 2 are different.

Scenario 1. This scenario illustrates the core functionality of the plug-in, i.e., mapping specification and validation. We will show how mappings can be used to handle hidden data semantics or filter data.

Consider the fragment of the client's database presented in Figure 6, which we call Customer's Database-a (CuDBa). The consultant starts with analyzing the CuDBa to understand the relationship between the schema/data of the database and the MiFID Ontology. The result is a set of mappings between the database schema and the MiFID Ontology (see mappings m1-m6 in Scenario 1 of Figures 6 and 7).

Using the mapping management facilities of the plug-in

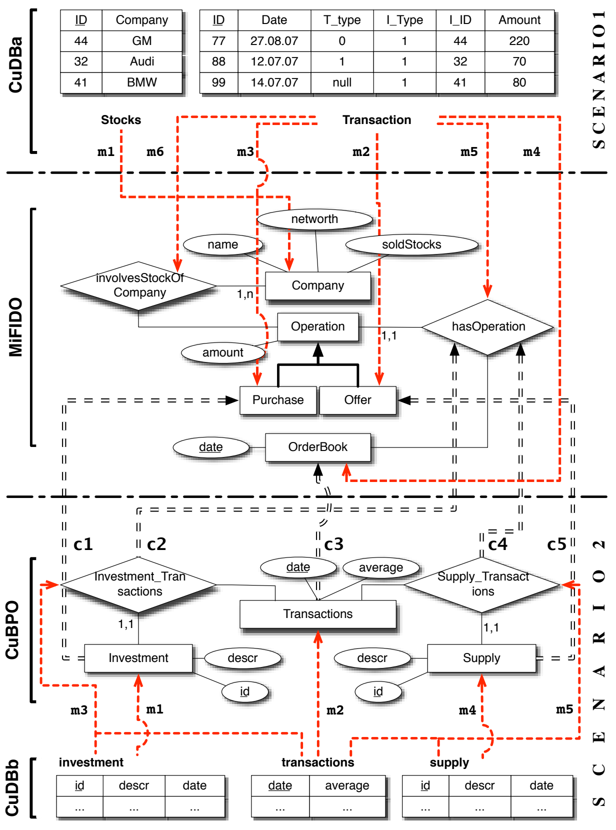


Fig. 6. Scenarios, MiFID Ontology and mappings

(see Figure 2), the consultant incorporates these mappings into the project. To verify them, he/she uses the plug-in's Query Tab (see Figure 3) to issue queries (in SPARQL syntax) over the classes, roles, and attributes of the MiFID Ontology. He compares the results that are returned by the OBDA-Enabled reasoner with the expected results. For example, consider a) a query asking for all objects of the *Operation* entity, $q1 = \texttt{SELECT \$x WHERE \{\$x rdf:type Operation\}}$, b) the mappings **m2** and **m3**, which respectively populate the classes *Offer* and *Purchase* with entities created with values from the relation *Transaction* in the CuDBa database (notice the use of the `WHERE` clause to select the matching tuples for each class with respect to the attribute T_type) and c) the results returned by the reasoner and presented in Figure 5(a). Using the database inspector of the OBDA Plug-in, the consultant notices that an operation object corresponding to the tuple with ID=99 in *Transaction* is missing. He/she realizes that the reason for this discrepancy is that some of the client's applications consider transactions with T_type = NULL as *Offer* transactions instead of only T_type=1 as in **m2**. Doing a quick adjustment of the mapping using the mapping manager, he/she corrects the problem by modifying the mapping **m2** into **m2b**. The results of this operation are shown in Figure 5(b). Through this iterative process the consultant models the rest of the mappings in the system. Once the model has been extensively tested, the application is deployed at the client.

Scenario 2. The task of specifying mappings between the

SCENARIO1

m1:	Company(getCompanyObj(ID)), name(getCompanyObj(ID), Company)	⇝	SELECT ID, Company FROM Stocks;
m2:	Offer(getOperationObj(ID)), amount(getOperationObj(ID), Amount)	⇝	SELECT ID, Amount FROM Transaction WHERE T_Type=1;
m3:	Purchase(getOperationObj(ID)), amount(getOperationObj(ID), Amount)	⇝	SELECT ID, Amount FROM Transaction WHERE T_Type=0;
m4:	OrderBook(getOrderBookObj(Date)), date(getOrderBookObj(Date), Date)	⇝	SELECT DISTINCT Date FROM Transaction;
m5:	hasOperation(getOrderBookObj(Date)), getOperationObj(ID)	⇝	SELECT ID, Date FROM Transaction WHERE I_Type=1;
m6:	InvolvesStockOfCompany(getOperationObj(Transaction.ID), getCompanyObj(Stocks.ID))	⇝	SELECT Transaction.ID, Stocks.ID FROM Stocks JOIN Transaction WHERE I_Type=1 OR I_Type=0;
m2b:	Offer(getOperationObj(ID)), amount(getOperationObj(ID), Amount)	⇝	SELECT ID, Amount FROM Transaction WHERE T_Type=1 OR T_Type=NULL;

SCENARIO2

m1:	Investment(getInvObj(id)),id(getInvObj(id),id),descr(getInvObj(id),descr)	⇝	SELECT id,descr FROM investment;
m2:	Transactions(getTransObj(date)),date(getTransObj(date),date),average(getTransObj(average),average)	⇝	SELECT date,average FROM transactions;
m3:	Investment_Transactions(getInvObj(id),getTransObj(date))	⇝	SELECT investment.id,transactions.date FROM investment JOIN transactions ON transactions.date=investment.date;
m4:	Supply(getSupplyObj(id)),id(getSupplyObj(id),id),descr(getSupplyObj(id),descr)	⇝	SELECT id,descr FROM supply;
m5:	Supply_Transactions(getSupplyObj(id),getTransObj(date))	⇝	SELECT supply.id,transactions.date FROM supply JOIN transactions ON transactions.date=supply.date;

Fig. 7. Mappings for Scenarios 1 and 2. Each mapping is given as a query over the ontology (left) paired with a query over the source (right). The mappings specify how the ontology is to be populated from the data in the source. Notice the use of Skolem functions in the ontology query in order to build object identifiers from source data, addressing the *impedance mismatch* problem. For a detailed explanation of the semantics of these mappings we refer to [1].

ontology and the data sources as described above is clearly time-consuming. Instead of establishing them manually, it is desirable to have automatic support for this task. This scenario shows how the consultant can use the ontology extraction functionality and inter-schema correspondences to link the MiFID Ontology to well-structured data sources. If the underlying database is properly normalized (i.e., is in third normal form), the extracted ontology is optimal and no further refinements are needed. In the general case, user intervention is required to verify and revise the obtained ontology. The actual extraction algorithm and its properties are described in [11].

Suppose the information of the customer's company is stored in a database CuDBb, as shown in Figure 6, SCENARIO2. It's in 3NF and the consultant uses the OBDA Plug-in to automatically extract the conceptual view from the CuDBb database, which we now call *Customer's Business Process Ontology* (CuBP Ontology). The mappings (single dashed lines labeled m1−m5 in Figure 6) connecting the database and the extracted ontology are also generated during the extraction process and are defined as shown in Figure 7, SCENARIO2.

Once the CuBP Ontology is obtained, the consultant specifies the correspondences between the elements this and elements of the MiFID Ontology, which is shown graphically in Figure 6 with dashed lines labeled $c1 - c5$. These correspondences are specified with a set of inclusion assertions of the form: c1: Investment $\sqsubseteq$ Purchase, c2: Investment_Transactions $\sqsubseteq$ hasOperation, etc. To verify these mappings before the system's deployment, the consultant uses querying facilities described earlier. The answers returned in this case take into account the inter-schema mappings and the mappings between the extracted CuBP Ontology and the CuDBb database. The major advantage of this approach is that specifying the required assertions at the ontology level is an easier task than dealing with mappings to the data layer.

IV. Conclusions and Demo session overview

We have described some of the operations that users can perform using the OBDA Plug-in. At the same time we have shown how, in conjunction with an OBDA-Enabled Reasoner, users can build OBDA Systems. It is worth mentioning that although in this paper the OBDA Plug-in was used as part of the cycle to build an OBDA System, the plug-in could also be used independently. That is, it could also be used as the main tool to perform ontology based access to the data. This would be the case in research and academic projects which do not have an application layer and are mainly interested in exploiting their data.

We conclude by listing some of the features which are not detailed here due to space restrictions but that have been implemented and will be presented during the demonstration session: (*i*) SPARQL query console and results table, (*ii*) incremental query answering support, (*iii*) result importing facilities, e.g., into csv, owl, xml, files. We will do performance tests soon after the release of the software to the public.

Acknowledgements: We would like to thank Sergio Tessaris, Alexandra Eckert, and Evgeny Kharlamov for important directions and comments that helped to shape this article. This research has been partially supported by EU FP6-7603 FET project TONES (Thinking ONtologiES).

References

[1] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati, "Ontology-based database access," in *Proc. of SEBD 2007*, 2007, pp. 324–331.

[2] L. Ma, J. Mei, Y. Pan, K. Kulkarni, A. Fokoue, and A. Ranganathan, "Semantic web technologies and data management," in *Proc. of W3C Workshop on RDF Access to Relational Databases*, 2007.

[3] L. M. Haas, "Beauty and the beast: The theory and practice of information integration," in *Proc. of ICDT 2007*, 2007, pp. 28–43.

[4] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubezy, H. Eriksson, N. F. Noy, and S. W. Tu, "The evolution of Protégé: an environment for knowledge-based systems development," *Int. J. of Human-Computer Studies*, vol. 58, no. 1, pp. 89–123, 2003.

[5] A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati, "QuOnto: Querying ontologies," in *Proc. of AAAI 2005*, 2005, pp. 1670–1671.

[6] E. Thomas, J. Z. Pan, and D. Sleeman, "ONTOSEARCH2: searching ontologies semantically," in *Proc. of OWLED 2007*, 2007.

[7] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen, "From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language," *J. of Web Semantics*, vol. 1, no. 1, pp. 7–26, 2003.

[8] M. Lenzerini, "Data integration: A theoretical perspective." in *Proc. of PODS 2002*, 2002, pp. 233–246.

[9] D. Calvanese and M. Rodríguez, "An extension of DIG 2.0 for handling bulk data," in *Proc. of OWLED 2007*, ser. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-258/, vol. 258, 2007.

[10] J. Giraud and C. D'Hondt, *MiFID – Convergence towards a Unified European Capital Markets Industry*. London: Risk Books, 2006.

[11] L. Lubyte and S. Tessaris, "Extracting ontologies from relational databases," in *Proc. of DL 2007*, ser. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-250/, vol. 250, 2007, pp. 387–394.