MASTRO STUDIO: Managing Ontology-Based Data Access applications

Cristina Civili, Marco Console, Giuseppe De Giacomo, Domenico Lembo,
Maurizio Lenzerini, Lorenzo Lepore, Riccardo Mancini, Antonella Poggi,
Riccardo Rosati, Marco Ruzzi, Valerio Santarelli, and Domenico Fabio Savo
DIAG, Sapienza Università di Roma
<lastname>@dis.uniroma1.it

ABSTRACT

Ontology-based data access (OBDA) is a novel paradigm for accessing large data repositories through an ontology, that is a formal description of a domain of interest. Supporting the management of OBDA applications poses new challenges, as it requires to provide effective tools for (i) allowing both expert and non-expert users to analyze the OBDA specification, (ii) collaboratively documenting the ontology, (iii) exploiting OBDA services, such as query answering and automated reasoning over ontologies, e.g., to support data quality check, and (iv) tuning the OBDA application towards optimized performances. To fulfill these challenges, we have built a novel system, called MASTRO STUDIO, based on a tool for automated reasoning over ontologies, enhanced with a suite of tools and optimization facilities for managing OBDA applications. To show the effectiveness of MASTRO STUDIO, we demonstrate its usage in one OBDA application developed in collaboration with the Italian Ministry of Economy and Finance.

1. INTRODUCTION

A key requirement for many organizations nowadays is that they can make use of advanced methods and systems for accessing their data. Indeed data are often dispersed in heterogeneous and autonomously evolving systems, or have been adapted through the years to the needs of the applications they serve, which makes it difficult to extract them in an useful format for the business of the organization.

Data integration solutions [4] provide some support to this problem. The tools they have produced are usually classified into *materialized* (*aka* Extract-Transform-Load, ETL) and *virtual* systems (*aka* mediators). In particular, the latter aim at providing access to autonomous data sources, through a unified virtual global schema. Thus, by taking into account declarative mappings from the sources to the schema, queries over the global schema are rewritten in terms of appropriate queries over the sources. However, the integrated global view offered by this kind of systems is often merely a structure accommodating the various data at the sources, whose semantics is typically unclear to information consumers.

Proceedings of the VLDB Endowment, Vol. 6, No. 12

Ontology-based data access (OBDA) [8] is a novel paradigm that is similar in spirit to virtual data integration, but relies on the idea of replacing the global schema by an explicit formal representation of the domain of interest. So, OBDA resorts to a three-level architecture, constituted by an ontology, the data sources, and the mapping between the two. By virtue of the use of an *ontology* as "a single point of semantic data access", OBDA overcomes the above mentioned drawback. Indeed, ontologies allow one to express information needs in terms of predicates whose semantics is explicitly defined in the ontology itself and *natively* abstracts from data. We note also that ontologies are nowadays extremely popular, as proven by a bunch of related logic-based standards (e.g., the W3C Ontology Web Language, OWL[1]) and practical tools for their exploitation, such as *ontology reasoners* (e.g., [11, 12]). This makes them ideally suited to provide access to actual information consumers.

In this work, we present MASTRO STUDIO, a new system offering effective capabilities for the management of OBDA applications. It is based on the MASTRO reasoner for OBDA. Hence, internally, ontologies are specified in logics of the *DL-Lite* family of Description Logics [2], well-known for providing a good tradeoff between expressivity and reasoning computational complexity. *DL-Lite* logics essentially capture standard conceptual modeling formalisms, such as UML Class Diagrams and Entity-Relationship Schemas, and are at the basis of OWL 2 QL, one of the tractable profiles of OWL 2, the current W3C standard language for ontologies[2]. Also, in MASTRO, data sources are seen as relational databases. Finally, the relationship between the sources and the ontology is essentially expressed by a set of GAV assertions [7], which associate ontology elements with queries specified on the underlying database. By virtue of these design choices, OBDA services, such as query answering, are realized in MASTRO through a very efficient technique that reduces them, via query rewriting, to standard SQL query evaluation.

In order to support the management of OBDA applications, MASTRO STUDIO is equipped with a suite of effective features. First, ontologies are specified and represented by means of a novel graphical language aiming both at making them accessible to non-experts of logical and ontology formalisms, and at capturing the main modeling features of OWL. This effectively supports the definition and the analysis of the ontology. Second, MASTRO STUDIO provides the capability to equip the ontology with a wiki-like documentation that, for every ontology element (concept, attribute or role) (i) specifies its meaning (in natural language) and (ii) reports

[1] http://www.w3.org/TR/owl2-overview/
[2] http://www.w3.org/TR/owl-profiles/

on the ontology and mappings assertions in which it is involved. Third, MASTRO STUDIO allows to exploit all OBDA services offered by MASTRO. Specifically, it allows to analyse the domain ontology exploiting intensional reasoning. Also, it allows to invoke query answering, and it provides support to the check of the quality of data with respect to the ontology, by returning to the user ontology assertions that are not satisfied together with the actual data that violate them. Finally, MASTRO STUDIO is equipped with a semi-automatic tuning mechanism, aiming at optimizing OBDA applications.

In order to demonstrate MASTRO STUDIO we will invite attendees to experiment all its capabilities through an OBDA application recently developed in collaboration with the Italian Ministry of Economy and Finance (MEF) directorate that is responsible for the Italian public dept management.

2. TECHNICAL BACKGROUND

In this section, we provide a brief overview of the theoretical background of our approach. We start by introducing the notion of OBDA specification. Then we discuss the choice we made for the languages used in MASTRO STUDIO to express the various components of an OBDA specification. We also illustrate the OBDA tasks that our system is able to perform. Finally, we briefly discuss the issue of tuning an OBDA application towards the optimization of query answering.

OBDA specification. An OBDA specification is a triple $\langle \mathcal{O}, \mathcal{M}, \mathcal{D} \rangle$, where $\mathcal{O}$ is an ontology, $\mathcal{D}$ is a relational database, and $\mathcal{M}$ is the mapping between $\mathcal{O}$ and $\mathcal{D}$. In principle, $\mathcal{O}$ is a set of axioms expressed in any fragment of first-order logic, whose goal is to provide a formal account of the domain of interest. The database $\mathcal{D}$ models the actual resources where real data are stored. We assume that $\mathcal{D}$ is expressed in the relational model. $\mathcal{M}$ is a set of assertions of the form $\Phi \rightsquigarrow \Psi$, where Φ is an SQL query over $\mathcal{D}$, and Ψ is a query over the alphabet of $\mathcal{O}$, without existential variables, of the same arity as Φ. Intuitively, such a mapping assertion specifies that the tuples returned by the query Φ satisfies the formula Ψ, and therefore create the bridge between the data in the sources and the objects satisfying the predicates in the ontology.

The semantics of an OBDA specification is given in terms of FOL interpretations. A FOL interpretation $\mathcal{I}$ is a model for an ontology $\mathcal{O}$ if it satisfies (in the classical FOL sense) all logical axioms specified in $\mathcal{O}$ [2]. Then, given an OBDA specification $\mathcal{B} = \langle \mathcal{O}, \mathcal{M}, \mathcal{D} \rangle$, a FOL interpretation $\mathcal{I}$ is a *model for* $\mathcal{B}$ if (i) $\mathcal{I}$ is a model for $\mathcal{O}$, and (ii) $\mathcal{I}$ satisfies $\mathcal{M}$, i.e., for each mapping assertion $\Phi \rightsquigarrow \psi$ and each tuple $\vec{t}$, if $\vec{t}$ is answer to the query Φ over $\mathcal{D}$, then $\vec{t}$ satisfies Ψ in $\mathcal{I}$ (see also [8]). Notice that the above notion of mapping satisfaction corresponds to the classical notion of satisfaction of *sound* GAV mapping in data integration [7]. An OBDA specification $\mathcal{B}$ is *satisfiable* if it admits at least one model.

Languages in MASTRO STUDIO**.** The ultimate goal of an OBDA system is to provide several services to the user. One notable service is computing the answers to queries expressed over the ontology $\mathcal{O}$. It is immediate easy to see that the tractability, and even the feasibility, of this task depends on the languages used to express the ontology and the mappings of the OBDA specification. The choices we have made in MASTRO STUDIO aims at an optimal compromise between expressive power of languages and computational complexity the reasoning services. To this end, a logic of the *DL-Lite* family of lightweight Description Logics (DLs) [2] specifically designed for the tractability requirement in OBDA, is used in MASTRO STUDIO. Analogously, MASTRO STUDIO limits the expressive power of the mapping language by letting specify only GAV mappings [7]. So, $\mathcal{M}$ is a set of assertions of the form $\Phi \rightsquigarrow \psi$, where Φ is an SQL query specified over the schema of $\mathcal{D}$, and ψ is an element of the ontology $\mathcal{O}$, i.e., a concept, a role, or an attribute (see also [8]). With such choices, it can be shown that query answering can be done with the same data complexity as SQL [8].

Services provided by MASTRO STUDIO**.** Services provided by MASTRO STUDIO can be classified into *intensional* and *extensional* reasoning services. The former are concerned with reasoning over the ontology, independently of the mappings to the sources. Essentially, all intensional reasoning services rely on the ability, given a formula, to check whether it is logically implied by the axioms constituting the ontology. The main services regarding the extensional level, i.e., the one with the mappings and the data sources, are *query answering* and *data quality checking*. Query answering amounts to compute the so-called certains answers to (unions of) conjunctive queries ((U)CQs) expressed over the ontology $\mathcal{O}$. Given an OBDA specification $\mathcal{B} = \langle \mathcal{O}, \mathcal{M}, \mathcal{D} \rangle$, the *certain answers*, to query Q, denoted *CertAns*$(Q, \mathcal{B})$,are the the tuples that satisy Q in every model of $\mathcal{B}$ (the FOL interpretation of a UCQ is the standard one [1]). MASTRO STUDIO computes such answers by evaluating over the database $\mathcal{D}$ the so-called perfect rewriting of Q, where a query Q_{DB} over $\mathcal{D}$ is a *perfect rewriting* of a query Q under $\mathcal{O}$ if the evaluation of Q_{DB} over $\mathcal{D}$ returns the set *CertAns*$(Q, \mathcal{B})$. In our setting, the perfect rewriting of a UCQ Q posed over $\mathcal{O}$ can be obtained in two steps: (i) compute an *ontology-rewriting* Q' of Q with respect to the ontology $\mathcal{O}$; (ii) compute the *mapping-rewriting* of Q' by using the mapping $\mathcal{M}$, thus obtaining an SQL query on $\mathcal{D}$. Intuitively, an ontology-rewriting of Q is another query Q', expressed over $\mathcal{O}$, which incorporates all the relevant properties of the ontology axioms, so that, by using Q', we can compute the certain answers of Q by ignoring $\mathcal{O}$, i.e., *CertAns*$(Q, \langle \mathcal{O}, \mathcal{M}, \mathcal{D} \rangle) =$ *CertAns*$(Q', \langle \emptyset, \mathcal{M}, \mathcal{D} \rangle)$. Then, the mapping-rewriting step can be seen as a variant of the unfolding procedure in GAV data integration, as it essentially substitutes each atom in the query Q' with the SQL query that the mapping associates to the atom predicate. After the rewriting process, the query is fully expressed in SQL and can be directly evaluated over the sources. Data quality checking amounts to perform several checks aiming at comparing the data at the sources with the axioms of the ontology. A notable example is satisfiability checking, i.e., checking whether there are patterns in the data contradicting the axioms in the ontology. In MASTRO STUDIO, satisfiability can be reduced to query answering, based on the fact that to each ontology axiom we can associate a query aiming at identifying the existence of patterns representing violations in the data.

Tuning ODBA applications. While in recent years we have seen many approaches to the ontology rewriting step and its optimizations, (see, e.g., [2, 10, 5]), very little has been done towards the optimization of the mapping rewriting step. Note the literature on data integration has mainly focused on the LAV approach to mappings [9, 6], where mapping rewriting is a form of view-based query rewriting, a well-known NP-complete problem. Differently, query answering under GAV mappings does not suffer from the intractability problem, and has been considered somehow trivial. On the contrary, according to our experiments in real world scenarios, the mapping rewriting phase is a bottleneck of query rewriting in OBDA, even under GAV mappings. In particular, computing the mapping rewriting of a CQ may be prohibitive if the mapping is even moderately complex. To address this problem, MASTRO STUDIO provides mechanisms for optimizing mapping rewriting [3]. We add *view inclusions* to the OBDA specification, i.e., inclusion assertions between (projections of) the SQL queries used

in mapping assertions, or, more precisely, between the corresponding view predicates. Based on such inclusions, derived automatically through a theorem prover, we are able to eliminate conjunctive queries contained into other conjunctive queries of the rewritten query. Using view inclusions, we then use a further optimization method, based on the use of so-called *perfect mapping assertions*. These are special assertions logically entailed by the OBDA specification, which allow the ontology and the mapping rewriting processes to handle whole subqueries as single atoms. We have shown that their usage leads to a drastic reduction of the combinatorial explosion of the mapping rewriting phase. In particular, we have designed a module that learns and exploit a suitable set of perfect mappings, so as to tune the application and make query asnwering more efficient. For further details, see [3].

3. SYSTEM OVERVIEW

MASTRO STUDIO components can be organized into three layers: a *Graphical User Interface (GUI)* layer, a *Utilities* layer and a *Reasoning* layer.

GUI layer. The MASTRO STUDIO functionalities are provided through a web-based GUI, realized through the Drupal[3] open source CMS (Content Management System). Hence, besides comprising Drupal core modules, the GUI layer includes contributed Drupal modules, for the management and the moderation of collaborative editing of the ontology wiki-like documentation. Furthermore, the GUI layer comprises custom modules (that is, extensions of the Drupal CMS) for (i) the loading and the analysis of the OBDA specification, (ii) the invocation of reasoning services over the OBDA application and the analysis of their results.

Utilities layer. This layer comprises modules to (i) translate a graphical representation of an ontology into the OWL functional-syntax representation required by components of the reasoning layer, (ii) automatically generate and update a structured wiki-like documentation, starting from the OBDA specification, by creating and updating a wiki page, for each concept, attribute and role of the ontology, according to a predefined template that, besides manually inserted description, includes related ontology and mapping assertions.

Reasoning layer. Components of this layer are modules that invoke and exploit the MASTRO reasoning services, through a web-service interface.

4. DEMONSTRATION

We demonstrate MASTRO STUDIO through a real world OBDA application that we developed in a joint project with the Italian Ministry of Economics and Finance (MEF), and more precisely with the MEF directorate, responsible for the management of the Italian public debt. Within such a context, the use of OBDA techniques has several motivations, e.g.,:

- Each office working group has a clear understanding only on particular portions of the public debt domain, adopts its own (informal) representation of it, and refers to common concepts with a specific terminology. This results in the lack of a shared (and formalized) specification of the knowledge on the overall public debt domain.
- Data within MEF are managed in various systems, which underwent several modifications in the years, often to serve specific application needs, so that they have lost the original shape and modeling, often without an adequate documentation, and are now easily accessible only by few experts of the systems, whereas current databases in use at MEF are essentially incomprehensible for the domain users.
- Integrity constraints on data are very often not forced in the systems, or are easily circumvented, so that their quality is compromised.
- Each time a new information need would arise, managers of the directorate would have to launch a new complicate process that would typically require several days and an important amount of money to be accomplished.

[3] http://drupal.org

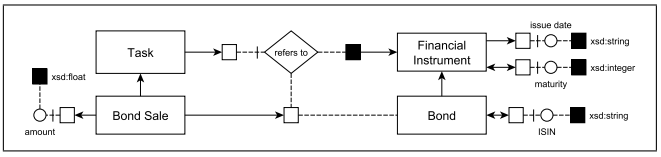

Figure 1: MEF ontology excerpt

We will show how MASTRO STUDIO, instantiated to the MEF OBDA application helps to face and solve the above issues. To this aim, we next provide a brief overview of the MEF ontology. The ontology describes the financial instruments used by the Ministry to generate debt, as well as the debt composition and the main features of debt components, such as the amount and the expiry date. It also describes how the debt evolves, across tasks aiming at producing new debt, increasing, reducing or extinguishing current debt. The ontology that we realized is specified through around 1440 *DL-Lite* assertions.

To have an idea of the graphical representation, consider the small excerpt of the ontology that will be the heart of the MASTRO STUDIO demonstration, given in Figure 1. As in Entity-Relationship (ER) diagrams, we represent (named) concepts through labeled rectangle, (named) roles through labeled diamond, (named) attributes through labeled circles. Differently from ER, besides named (a.k.a. atomic) concepts, roles, or attributes, also complex ones can be represented, through suitable graphic constructs. For example, the white (resp. black) square connected to a role R via a dotted line (possibly with a vertical dash) denotes the set of objects occurring in the first (resp. second) component of R, i.e., its *domain* (resp. *range*) (we remind that roles are binary relationships between sets of objects). The optional vertical dash on the dotted line between R and the white (resp. black) square imposes that the role (resp. its inverse) is functional, i.e., the every object in its domain (resp. range) is associated via R to at most one object. Analogously if we connect the square to an attribute, where however the range denotes a set of values belonging to predefined data types (cf. the labels associated to back squares connected to attributes). Also, the white square connected to both a role R and a concept C through two different dotted lines, denotes the set of objects that participates in the first component of R and are associated via R to at least one instance of C (this corresponds to so-called *existential qualified restrictions* in *DL-Lite* and OWL). Finally, solid arrows denote inclusions between concepts, which can be used to model ISA between concepts, mandatory participations of concepts into roles, typings of roles and attributes, ecc. In words, the fragment of ontology in Figure 1 says that: a Task refers to exactly one particular Financial Instrument (notice that the concept Task is included in the domain of refers to, i.e., it has a mandatory participation in such role, and that refers to is functional); a Bond is a particular Financial Instrument; a Bond Sale is a particular Task, which refers to a Bond (notice that Bond Sale is included in the existential restriction on the concept Bond of the role refers to); a Financial Instrument has an issue date and a maturity

(i.e., a duration), each Bond as an ISIN (International Securities Identification Numbering), and each Bond Sale is associated to the sold amount. All mentioned attributes are functional. Notice that, differently from ER, typing of roles is not compulsory in the ontology. For example the domain of refers to is not typed, which means that also objects not necessarily instances of Task can occur in the domain of refers to. Similarly for the attributes issue date and amount.

We are now ready to describe the various MASTRO STUDIO functionalities that will be demonstrated.

Browsing the ontology documentation Participants will experience the effectiveness of the ontology wiki-like documentation provided by MASTRO STUDIO, which will introduce them to the overall ontology semantics, as well as to the semantics of each ontology element. Participants will test the collaborative semi-automatic process supporting the production of such documentation.

Analysing the ontology Participants will be introduced to the richness of the Italian public debt scenario, by analysing the ontology through the MASTRO STUDIO GUI and the intensional reasoning facilities it offers. Participants will be able to test the usefulness of the diagrammatic representation as a means to get an easy access to the ontology and to disclose it to users not willing to go through complex formalizations.

Analysing the data sources and the mappings. Participants will be able to analyse the sources and to issue directly SQL queries over them. This will give the feeling about how difficult is to query the data sources without the mediation of the ontology (and the mapping). Then, they will have a look at the mappings: these will give an insight of the "cognitive distance" between the ontology and the sources, i.e., the huge difference between the data schema of the sources and the conceptualization of the domain, and will provide at the same time a mechanism to understand the sources in the light of the ontology, offering a valid documentation means.

Checking the quality of data. Participants will be able to identify unsatisfiable ontology assertions, and retrieve source data that violate them. This will show how, in the MEF scenario, MASTRO STUDIO allowed us to localize inconsistencies in the data, thus resulting a valid support for data quality management.

Querying the system. Participants will be invited to issue queries over the ontology, and for each query, to access, besides its results, both the ontology rewriting and the mapping rewriting. For example, they will be able to ask for the amount of the Italian public debt at a certain date and how much such a debt costs in terms of interests to the state. By analysing the ontology rewriting, they will discover the kind of reasoning at the ontology level which is automatically performed by the system to produce the result. In the example query mentioned, they can discover which are the components of the debt and which are the financial instruments that can produce debt. Furthermore, by analysing the mapping rewriting, they will be able to see how and from which sources results to specific queries come from. Figure 2 shows a screenshot of the MASTRO STUDIO GUI, with the SPARQL query that asks, for each financial instrument, its ISIN, type, and the amount of debt currently generated by it. This simple way of querying the OBDA specification can be immediately contrasted with the current complex way in which reports corresponding to such queries are produced at MEF.

Tuning the system. We will invite participants to have a closer look at the mapping specification and will show how this evolves, by acquiring new perfect mapping assertions derived from perfect rewritings computed during the evaluation of previous queries. This corresponds to a tuning of the system, which is able to learn from previous processing of queries in order to avoid to execute some reasoning steps it already performed. We will demonstrate how queries, corresponding to reports of crucial importance for MEF, can be executed only thanks to this tuning.

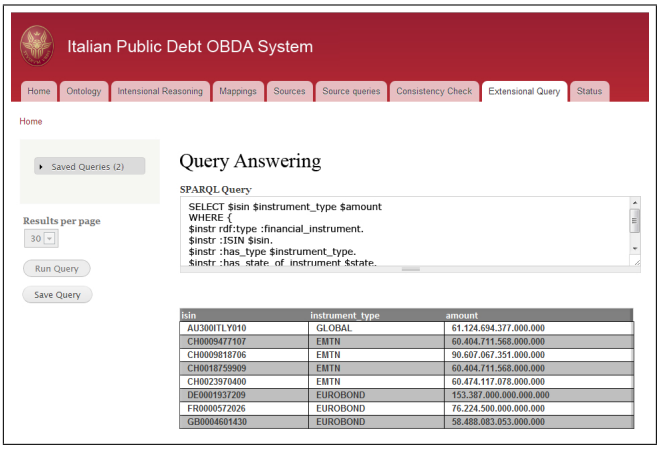

Figure 2: MASTRO GUI screenshot

Acknowledgments. Research on OBDA, and in particular on mapping management, has been partially funded by the EU under FP7 project Optique – Scalable End-user Access to Big Data (grant n. FP7-318338).

5. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., 1995.

[2] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.

[3] F. Di Pinto, D. Lembo, M. Lenzerini, R. Mancini, A. Poggi, R. Rosati, M. Ruzzi, and D. F. Savo. Optimizing query rewriting in ontology-based data access. In *Proc. of EDBT 2013*, 2013.

[4] A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[5] G. Gottlob, G. Orsi, and A. Pieris. Ontological queries: Rewriting and optimization. In *Proc. of ICDE 2011*, pages 2–13, 2011.

[6] G. Konstantinidis and J. L. Ambite. Scalable query rewriting: a graph-based approach. In *Proc. of ACM SIGMOD*, pages 97–108, 2011.

[7] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS 2002*, pages 233–246, 2002.

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

[9] R. Pottinger and A. Y. Halevy. MiniCon: A scalable algorithm for answering queries using views. *VLDB Journal*, 10(2–3):182–198, 2001.

[10] M. Rodriguez-Muro and D. Calvanese. High performance query answering over *DL-Lite* ontologies. In *Proc. of KR 2012*, pages 308–318, 2012.

[11] E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *J. of Web Semantics*, 5(2):51–53, 2007.

[12] D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of IJCAR 2006*, pages 292–297, 2006.