CEUR-WS.org/Vol-2518/paper-DAOSI1.pdf

Towards *All-In-One* OBDA Systems

Germán BRAUN [a,b,c], Laura CECCHI [a] and Pablo FILLOTTRANI [c,d]
[a] *Informatics Faculty, Universidad Nacional del Comahue, Argentina*
[b] *Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina*
[c] *LISSI, Computer Science and Engineering Department, Universidad Nacional del Sur, Argentina*
[d] *Comisión de Investigaciones Científicas de la provincia de Buenos Aires (CIC), Argentina*

Abstract In this work we introduce the notion of *configuration* of an OBDA system, departing from a set of off-the-shelf tools that interoperate between each other in order to facilitate to users the access to data sources through queries on a conceptual level. The result is an extensible *all-in-one* framework with well-defined interfaces aimed at experimenting with different combination of tools. Finally, we discuss the advantages and limitations of this proposal.

Keywords. ontology based data access, software engineering, automated tools

1. Introduction and Motivation

Ontology-based data access (OBDA) [15, 37] is a new paradigm of data management based on semantic technology and aimed at facilitating access to diverse types of data sources by mediating between data consumers and data sources. To do this, the OBDA paradigm requires a conceptual domain view in terms of an *ontology* and a declarative *mapping* between the data and such ontology. OBDA users will express their queries over the conceptual domain model, which are reformulated into standard DB queries to be executed by the respective database management system. Thus, OBDA relies upon both DB and KR technologies for hiding the structure of the data sources, and providing a semantic access so that users no longer need an understanding of such sources. In the last years there have been important advances in precisely defining the theoretical aspects of this approach: query answering, which is its main reasoning task, query rewriting and optimisation [30], mapping languages [13], computational properties [9] and DL-Lite family [2]. On the tools hand, many of them have been also developed ranging from ontology editors [24, 11] to mapping tools [32, 4], query answering systems [8, 14, 31, 3], and visual query platforms [34]. A more complete description of the tooling ecosystem around OBDA is found in [15].

Notwithstanding nowadays some OBDA systems based on these tools have been successfully deployed at companies [16, 23], findings and perspectives derived from these implementations suggest that important technological and practical challenges

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

should be addressed and many of these problems have been tackled in isolation [23]. Such conclusions are associated to the semi-automatic or automatic development of mappings and ontologies, an efficient query processing and the user-friendliness of the OBDA system for expressing queries. Thus, this lead to a very close interaction between basic and applied computer science motivated on the need to define how should the tools in the previously detailed ecosystem interoperate each other for moving towards facilitating the deployment of more user-friendly OBDA systems in real environments.

In this work we outline and describe the main tool interfaces needed for interacting each other in a OBDA system. The aim is to have a picture of the requirements of tools in order to go towards highly interoperable technological solutions. For instance, for a typical scenario ontology-mappings-databases-queries, it is needed to model an ontology representing the domain of interest, map data sources to the representation, and finally, interact with the sources through queries, interfacing database engines. In this context, modellers, domain experts and customers should be involved in ontology and mapping engineering tasks and thus requiring tools providing graphical support for manipulating ontologies and mappings and newly generating the corresponding OWL 2 [36] and R2RML (or other) specifications. Some implementations such as ontop [8] and D2R [3] provide bootstrappers to automatically extract ontologies and mappings from relational schemes, however, they should work along with other tools in order to validate the ontology with experts. As a consequence, mappings also could be updated considering possible extensions of the domain semantic model. Lastly, end-users need to interact with OBDA systems more smoothly and thus tools with a graphical interface need coupled to a system in real environments. The importance of this study has been already considered in recent works referenced here [15, 23] and justified through realistic applications, although, to the best of our knowledge, no model nor descriptive analysis of the tool interfaces in the OBDA ecosystem has been yet reported. Thus, we outline on the inputs and outputs that diverse tools require and present the notion of *configuration* of an OBDA system departing from an *all-in-one* framework aiming at experimenting with diverse combinations of tools. Such framework is being implemented for extending a concrete tool named *crowd*[1] [7, 6].

This paper is structured as followed. Section 2 details the context in which our research takes place and including most relevant related works. Section 3 presents the description of tool interfaces, the proposed *all-in-one* OBDA model and formalises the notion of *configuration* of an OBDA system based on both the previous concepts. Section 4 presents final discussion and conclusions aiming at generating a deeper insight in interactions between OBDA tools.

2. Context: OBDA Systems

The technological features associated to each implementation cannot be predicted at all and workarounds are needed to make tools cooperate, which makes the whole creation of knowledge graphs hardly predictable in cost and effort. In this sense, we describe the following systems with diverse degrees of support to the OBDA dimensions: ontology and mapping tools, query answering systems, database engines and federators, and tools

[1]`http://crowd.fi.uncoma.edu.ar/`

for formulating user queries. From this analysis we bring to the light the need for a clear definition of the interfaces of involving tools.

Mastro Studio [10] is a web platform for data management based on OBDA, currently commercialised by OBDA Systems[2]. Ontologies in Mastro are specified in DL-Lite and visualised in Graphol, while mappings are provided as a set of assertions between elements of the ontology and an SQL query, or possibly a conjunction of them. Mappings can be inspected in a particular textual environment. The core of Mastro Studio is the OBDA reasoner Mastro [14]. Finally, the platform provides a query answering environment for SPARQL queries and their responses. Other capabilities include data source and intentional reasoning inspection. A user study of Mastro Studio has been reported in [1].

Optique [16] is another end-to-end platform for OBDA for semantic access to oil and gas data from industry. Optique is based on ontop for query answering, BootOX [21] as ontology and mapping bootstrappers allowing specialists to write and edit, and import OWL ontologies and mappings from the underlying relational DBs. Moreover, the platform incorporate a novel OptiqueVQS [34], which allow users to formulate queries that are automatically translated into SPARQL.

Equinor solution [23] describes the development and deployment of an OBDA system for accessing geological data. Similarly to Optique, this proposal is a ad-hoc deployment so that it presents conclusions from practice experiences in accessing big data. In particular, the solution is equipped with a module for semi-automatic creation of ontologies and mappings, a query processing module for efficient OBDA query processing, a federated query execution module, and a query formulation platform based on OptiqueVQS. Some others user studies related to ontop have been reported in [28, 22]

3. *All-In-One* OBDA Systems

In spite of the fact that few user studies have been reported in the literature and being still an important research direction [15], they reveal several issues related not only to the needs of dealing with the *right* ontology or mapping languages but also to the way in which existing tools should interact each other, which are their interfaces, inputs and outputs. The importance of such issues flows from the OWL standarisation and the subsequent development of tools and infrastructure that can be used to support the development and deployment of OWL ontologies [19]. Similarly, providing pieces of infrastructure for communication with reasoners and ontology manipulation have been the main motivation for the development of the OWLlink protocol [25] and the well-known OWL API [18]. Moreover, methodological aspects are required in all the stages for setting up OBDA implementations, particularly, during the engineering of ontologies and mappings [20] and where supporting tools are also needed to communicate models between domain experts and developers.

In the same direction, OBDA systems also require to be supported by an ecosystem of additional tools with acceptable level of interoperability among them. For a typical scenario ontology-mappings-databases-queries, we need to model an ontology representing the domain of interest, map data sources to such representation, and finally, interact with the sources through queries, interfacing database engines. Thus, we sketch a

[2] `http://www.obdasystems.com/`

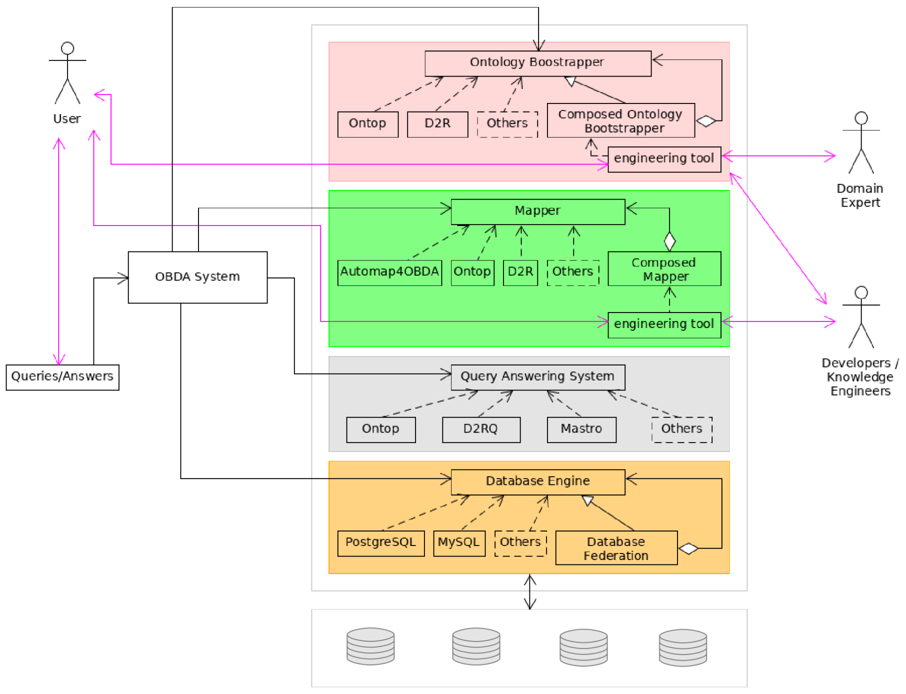


Figure 1. Schematic diagram of our model for OBDA systems.

preliminary scheme including the components involving in OBDA system, describe each component and detail the interfaces of each them in order to deploy system along with the right tools. Finally, we introduce and formalise the concept of *configuration* of an OBDA system. Fig. 1 depicts our preliminary proposal, where an OBDA system can be configured by setting an ontology bootstrapper that could interact with an engineering tool as a complement to the semi-automatic or automatic generation of the ontology; a mapper to automatic or semi-automatic generation of mappings, also supported by respective engineering tools; a query answering system; and finally, a database engine, which can be extended to support federators. Below we briefly describe each component and their interfaces. The specific tools in the diagram of the Fig. 1 illustrate the proposal, however, other ones could be considered in a concrete implementation.

Ontology Bootstrapper (OB) Given a scheme database S, the ontology bootstrapper extracts an ontology O from S with an associated vocabulary V. This capability is provided for tools such as ontop, D2R and BootOX, among others. However, as has been recently reported in [23], extensions of a bootstrapped ontology must be considered for introducing knowledge from domain experts. Tools for ontology engineering have been widely studied but without a accepted visual support nor ontology engineering processes clearly defined at all [35]. So that engineering tools incorporated to OBDA systems should provide OWL importing and exporting capabilities in order to bring back the formal ontology to the system. The standard language for expressing an ontology is OWL 2, which allows to model a hierarchy of classes, and domain and range of properties. The main query answering systems support reasoning over OWL 2 QL ontologies, whose formal foundation is the description logic DL-Lite and which capture the main modelling features of a variety of representation languages, maintaining low the complexity of reasoning (query answering in AC^0 for data complexity and tractable KB satisfiability for combined complexity) [2].

input: A database scheme S.
output: An OWL 2 specification representing S.

Mapper (M) Next component of the OBDA configuration is the mapper, which is also a bootstrapper to extract a mapping specification from an ontology O and a given database scheme S. The output of a mapper is a specification in a mapping language, i.e. a R2RML, which is also a W3C standard[3], or ad-hoc languages, such as the ontop native mapping format, among others. The aim of this component is going towards the automatic generation of mappings, which is implemented by ontop, D2R, Automap4OBDA [33], among other ones. Different from the two first, some tools as Automap4OBDA allows interfacing with only PostgreSQL engines and thus limiting the flexibility of the resulting configuration. With reference to visual tools or visual notations for mapping engineering, some of them are being currently proposed: MapOWL [17], gra.fo[4], among others, however, to the best of our knowledge, none of them have been widely accepted yet. Such visual tools incorporated to OBDA systems should provide a comprehensive view of the domain of interest and also about the data sources [15]. Finally, they should provide features to export new mappings in standard mapping specifications and thus being newly considered in the current OBDA process.

input: A database scheme S and an OWL 2 ontology O.
output: A mapping specification M between S and O

Query Answering System (QAS) Query answering systems is central in a OBDA system because it implements the query reformulation required to express SPARQL queries in terms of one or more SQL queries. So they should be set with standard SPARQL and SQL, in addition to OWL 2 ontologies and mapping specifications. The output of this component is the query results to be send to end-users. The well-known ontop, DR2Q and Mastro are such systems although other as Ultrawrap can be also integrated. Particularly, ontop has been used for various successful deployments in the industry [16, 23] and throwing good performance results.

input: A database instance D of S, an OWL 2 ontology O, a mapping M and a SPARQL query Q.
output: the result of Q.

Database Engine (DE) Engines are off-the-shelf components aiming at managing the access to the databases underlying to an OBDA system. Commercial and Open source ones are supported by ontop, Mastro and D2R via JDBC so that ontology and mapping bootstrappers, and query answering systems can be run with diverse database technologies. Other tools as Automap4OBDA also can be deployed as part of an OBDA system,

[3]`https://www.w3.org/TR/r2rml/`
[4]`http://gra.fo/`

however, it presents some limitations because it supports only the PosgreSQL engine.

After describing the main components of our model, we formalise a configuration of an OBDA system. First of all, in standard OBDA, an *OBDA specification* $\mathcal{P} =< \mathcal{O}, \mathcal{M}, \mathcal{S} >$ consists of an ontology $\mathcal{O}$, a data source scheme $\mathcal{S}$ and a mapping $\mathcal{M}$ from $\mathcal{S}$ to $\mathcal{O}$. Moreover, an *OBDA instance* is a pair $(\mathcal{P}, \mathcal{D})$, being $\mathcal{D}$ a database instance conforming to $\mathcal{S}$ [37]. Then, a configuration for an OBDA system dealing with relational databases is the following:

Definition 1 (An OBDA Configuration)**.** *Let $\mathcal{S}$ be a relational database scheme, $\mathcal{P} =< \mathcal{O}, \mathcal{M}, \mathcal{S} >$ an* OBDA specification *and $\mathcal{D}$ a relational database instance conforming to $\mathcal{S}$. An* OBDA configuration *(*iOBDA*) is a tuple $< OB, M, QAS, DE >$ where:*

- *OB is an ontology bootsrapper generating $\mathcal{O}$.*
- *M is a mapper generating $\mathcal{M}$.*
- *QAS is a query answering system for rewriting SPARQL queries into SQL ones.*
- *DE is a database engine for interfacing $\mathcal{D}$.*

Thus, a configuration of a OBDA system, named iOBDA, aiming at providing access to the data layer, through the orchestration of all of the tools involved in the OBDA process and involving to users into the manipulation of ontologies or mappings through respective engineering tools fulfilling the requirements previously detailed about their interfaces.

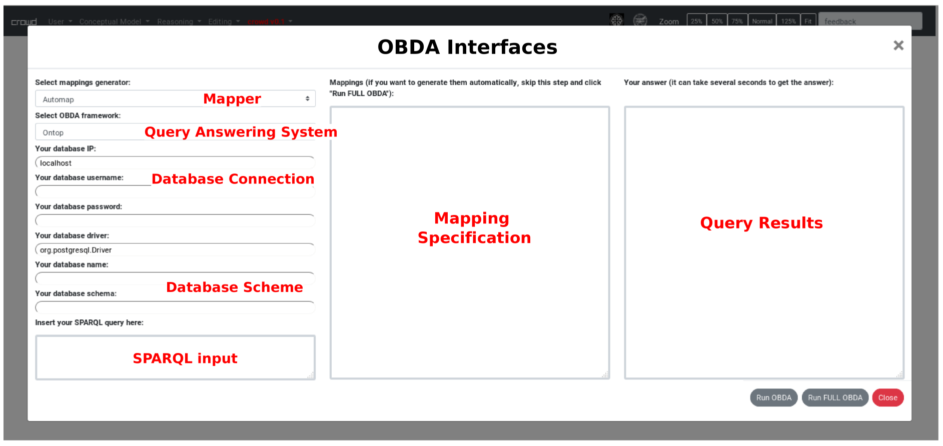
A possible OBDA configuration could be <D2R (Protégé), BootOX (RMLEditor), mastro, MySQL>, where D2R bootstraps an initial ontology from an underlying DB and thus being manipulated in Protégé by domain experts and developers. BootOX and RMLEditor[5] are set as a composed mapper, firstly extracting an initial mapping specification and lastly, involving developers and knowledge engineers in manipulating such specification through an engineering tool. Finally, the OBDA configuration considers mastro as the query answering system. Thus, end users queries expressiveness is limited to a restricted fragment of SPARQL that corresponds to unions of conjunctive queries (union-select-project-join queries). If the expressiveness of these queries needs to be increased, the OBDA system should be re-configured by replacing mastro by ontop, <D2R (Protégé), BootOX (RMLEditor), ontop, MySQL >, which supports SPARQL OPTIONAL for OBDA for dealing with missing information [38].

The Fig. 2 shows the OBDA components implemented in our prototype *crowd*, which enables the configuration of an OBDA system as explained in our model. The preliminary interface does not include any ontology bootstrapper but it takes an ontology previously defined in *crowd*. SPARQL inputs are also provided in a textual way. Lastly, evaluation of this implementation should be run in depth once more tools are coupled to *crowd* and, particularly with data sources from real domains.

4. Discussions and Conclusions

In spite of the fact that this proposal deal with some perspectives extracted from [15, 23], in this work we bring to light the need of agreements about how the tools around the

[5]`http://rml.io/RMLeditor.html`


OBDA Interfaces

Select mappings generator: Automap

Select OBDA framework: Ontop

Your database IP: localhost

Your database username:

Your database password:

Your database driver: org.postgresql.Driver

Your database name:

Your database schema:

Insert your SPARQL query here:

Mappings (if you want to generate them automatically, skip this step and click "Run FULL OBDA"):

Your answer (it can take several seconds to get the answer):

Figure 2. *crowd* widget showing settings to get an OBDA configuration (iOBDA).

OBDA paradigm should interoperate between them and thus going towards a standardisation of this interaction. Some previous experiences coming from ontology engineering remark this need, where even nowadays is not enough clear the effectiveness of the tools supporting ontologies [35]. OBDA shares some of these weaknesses because processes associated to this approach also appear fragmented across many tools or workarounds. In this discussion, we highlight the main applicability issues and limitations to be addressed.

Scalability of the solution Currently, our preliminary tool only supports single database instances and textual SPARQL queries. In order to scale it, we should explore federators as well as platforms for SPARQL visual queries as [34]. On the other hand and as reported in previous experiences [23], deploying OBDA systems in real companies could require going beyond the state-of-the-art systems so that as a conclusion, achieving effective and reusable solutions is an important practical challenge. Other issues associated to the deployment of OBDA systems involve aligning ontologies with those bootstrapped ones from the databases and ensuring the quality of the whole system [12]. Accessing to data stored in NoSQL databases should be also considered [27, 5, 26].

Manual vs. Automatic or Semi-automatic generation of ontologies and mappings Our proposal mainly considers automatic or semi-automatic generation of ontologies and mapping specification, although to the best of our knowledge, no analysis has been done for evaluating the effectiveness of such generators. Nevertheless, these tools reduce the costs associated to model ontologies and mappings from scratch. On this hand, tools as *crowd* are integrated to our work in order to help users to extend, inspect and debug their ontologies. Mapping editor should provide similar features but this is not supported by the first version of our model.

Standardisation of Tool Interfaces Platform-based approaches leads to standardisation of procedures, workflows and technology within an organisation [29]. Such platforms state technology bases on which other technologies or processes are built. OBDA systems need of a common platform, whose parts can be used in diverse configurations. Nevertheless, this customisation requires of artefacts to be sufficiently adaptable to fit into the different systems meaning that we have to identify where the underlying technologies differ or agree.

As a conclusion, in the last years there have been important theoretical advances in precisely defining the semantics and properties of the framework of ontology-based data access [37]. Such theoretical contributions induce the development of new technologies to be used in more effective tools and thus leading to a very close interaction between basic and applied computer science, which distinguishes the research area. In order to go towards this interaction, we define a configuration of an OBDA system and present an initial model for setting such configurations. This proposal aims to bring together well-established tools in the OBDA tooling ecosystem in a common and holistic platform and thus establishing an integrated connection from users to data sources. We also remark the need for well-defined interfaces for automatic or semi-automatic generation of ontologies and mappings as well as for interacting with diverse query answering systems and databases engines.

References

[1] N. Antonioli, F. Castanò, S. Coletta, S. Grossi, D. Lembo, M. Lenzerini, A. Poggi, E. Virardi, and P. Castracane. Ontology-based Data Management for the Italian Public Debt. In *Formal Ontology in Information Systems*, 2014.

[2] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The DL-Lite Family and Relations. *J Artif. Intell. Res.*, 2009.

[3] C. Bizer and R. Cyganiak. D2RQ - Lessons Learned. *W3C Workshop on RDF Access to Relational Databases*, 2007.

[4] C. Bizer and A. Seaborne. D2RQ - treating non-RDF databases as virtual RDF graphs. In *ISWC&Posters*, 2004.

[5] E. Botoeva, D. Calvanese, B. Cogrel, M. Rezk, and G. Xiao. OBDA Over Non-Relational Databases. In *10th Alberto Mendelzon International Workshop*, 2016.

[6] G. Braun. *Metodologías y Herramientas Visuales para Ingeniería Ontológica*. PhD thesis, Universidad Nacional del Sur, Argentina, 2019.

[7] G. Braun, E. Estevez, and P. Fillottrani. A Reference Architecture for Ontology Engineering Web Environments. *Journal of Computer Science & Technology*, 2018.

[8] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao. Ontop: Answering SPARQL queries over relational databases. *Semantic Web Journal*, 2017.

[9] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. *Artificial Intelligence*, 2013.

[10] 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. Fabio Savo. MASTRO STUDIO: managing ontology-based data access applications. *PVLDB*, 2013.

[11] M. Console, D. Lembo, V. Santarelli, and D. Savo. Graphol: Ontology Representation through Diagrams. In *Description Logics*, 2014.

[12] M. Console and M. Lenzerini. Data Quality in Ontology-based Data Access: The Case of Consistency. In *Proceedings of the Twenty-Eighth AAAI*, 2014.

[13] S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF Mapping Language, 2011.

[14] G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, R. Rosati, M. Ruzzi, and D. Fabio Savo. MASTRO: A reasoner for effective ontology-based data access. In *ORE*. CEUR-WS.org, 2012.

[15] G. Xiao; L. Ding; B. Cogrel; D. Calvanese. Virtual Knowledge Graphs: An Overview of Systems and Use Cases. *Data Intelligence*, 2019.

[16] M. Giese, A. Soylu, G. Vega-Gorgojo, A. Waaler, P. Haase, E. Jiménez-Ruiz, D. Lanti, M. Rezk, G. Xiao, and O. Özçep and R. Rosati. Optique: Zooming in on Big Data. *IEEE Computer*, 2015.

[17] P. Heyvaert, A. Dimou, B. De Meester, T. Seymoens, A. Herregodts, R. Verborgh, D. Schuurman, and E. Mannens. Specification and implementation of mapping rule visualization and editing: MapVOWL and the RMLEditor. *Journal of Web Semantics*, 2018.

[18] M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL Ontologies. *Semantic Web Journal*, 2011.

[19] I. Horrocks. Tool Support for Ontology Engineering. In *Foundations for the Web of Information and Services - A Review of 20 Years of Semantic Web Research.*, 2011.

[20] Juan J. Sequeda and D. Miranker. A Pay-As-You-Go Methodology for Ontology-Based Data Access. *IEEE Internet Computing*, 21, 2017.

[21] E. Jiménez-Ruiz, E. Kharlamov, D. Zheleznyakov, I. Horrocks, C. Pinkel, M. Skjæveland, E. Thorstensen, and J. Mora. BootOX: Practical mapping of RDBs to OWL 2. In *International Semantic Web Conference*. Springer, 2015.

[22] E. Kharlamov, T. Mailis, G. Mehdi, C. Neuenstadt, Ö. Özçep, M. Roshchin, N. Solomakhina, A. Soylu, C. Svingos, S. Brandt, M. Giese, Y. Ioannidis, S. Lamparter, R. Möller, Y. Kotidis, and A. Waaler. Semantic access to streaming and static data at siemens. *Journal of Web Semantics*, 2017.

[23] E. Kharlamov, M. Skjæveland, D. Hovland, T. Mailis, E. Jimenez-Ruiz, G. Xiao, A. Soylu, I. Horrocks, and A. Waaler. Finding data should be easier than finding oil. In *2018 IEEE International Conference on Big Data*, 2018.

[24] H. Knublauch, R. Fergerson, N. Noy, and M. Musen. The Protégé OWL plugin: An Open Development Environment for Semantic Web Applications. In *The Semantic Web – ISWC*, 2004.

[25] T. Liebig, M. Luther, M. Rodriguez, D. Calvanese, M. Wessel, R. Möller, M. Horridge, S. Bechhofer, D. Tsarkov, and E. Sirin. OWLlink: DIG for OWL 2. In *OWLED*. CEUR-WS.org, 2008.

[26] M. Mami, D. Graux, S. Scerri, H. Jabeen, and S. Auer. Querying Data Lakes Using Spark and Presto. In *The World Wide Web Conference*, 2019.

[27] F. Michel, Z. Faron, and J. Montagnat. A Mapping-based Method to Query MongoDB Documents with SPARQL. In *27th International Conference on Database and Expert Systems Applications*, 2016.

[28] A. Mosca, F. Roda, and G. Rull. UNiCS - The Ontology for Research and Innovation Policy Making. In *Formal Ontology in Information Systems*, 2018.

[29] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag New York, Inc., 2005.

[30] J. Sequeda, M. Arenas, and D. Miranker. OBDA: query rewriting or materialization? in practice, both! In *The Semantic Web - ISWC*, 2014.

[31] J. Sequeda and D. Miranker. Ultrawrap: SPARQL execution on relational data. *Journal of Web Semantics*, 2013.

[32] J. Sequeda and D. Miranker. Ultrawrap Mapper: A Semi-Automatic Relational Database to RDF (RDB2RDF) Mapping Tool. In *The Semantic Web - ISWC*, 2015.

[33] A. Sicilia and G. Nemirovski. AutoMap4OBDA: Automated Generation of R2RML Mappings for OBDA. In *International Conference on Knowledge Engineering and Knowledge Management*, 2016.

[34] A. Soylu, E. Kharlamov, D. Zheleznyakov, E. Jiménez-Ruiz, M. Giese, M. Skjæveland, D. Hovland, R. Schlatte, S. Brandt, H. Lie, and I. Horrocks. OptiqueVQS: A visual query system over ontologies for industry. *Semantic Web Journal*, 2018.

[35] M. Vigo, S. Bail, C. Jay, and R. Stevens. Overcoming the pitfalls of ontology authoring: Strategies and implications for tool design. *International Journal of Human Computer Studies*, 2014.

[36] World Wide Web Consortium (W3C). OWL 2 Web Ontology Language Document Overview (Second Edition), 2012. `http://www.w3.org/TR/owl2-overview/`, accedido en Junio de 2013.

[37] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, and M. Zakharyaschev. Ontology-Based Data Access: A Survey. In *IJCAI*, 2018.

[38] G. Xiao, R. Kontchakov, B. Cogrel, D. Calvanese, and E. Botoeva. Efficient Handling of SPARQL OPTIONAL for OBDA. In *The Semantic Web - ISWC*, 2018.