OBDA with the Ontop Framework

Diego Calvanese[1], Benjamin Cogrel[1], Elem Guzel Kalayci[1], Sarah Komla Ebri[1], Roman Kontchakov[2], Davide Lanti[1], Martin Rezk[1], Mariano Rodriguez-Muro[3], and Guohui Xiao[1]

[1]Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
[2]Dept. of Computer Science, Birkbeck, University of London, UK
[3]IBM T.J. Watson Research Center, Yorktown Heights, NY, USA

Abstract. Ontology-based data access (OBDA) has become a popular paradigm for accessing data stored in legacy sources using Semantic Web technologies. In the OBDA setting, users access the data through a conceptual layer, which provides a convenient query vocabulary abstracting from specific aspects related to the data sources. This conceptual layer is typically expressed as an RDF(S) or OWL ontology, and it is connected to the underlying relational databases using R2RML mappings. When the ontology is queried in SPARQL, the OBDA system exploits the mappings to retrieve elements from the data sources and construct the answers expected by the user. Different approaches for query processing in OBDA have been proposed. We focus here on the virtual approach, which avoids materializing triples retrieved through mappings and answers the SPARQL queries by translating them into SQL queries over the data sources. In this paper we present our mature open-source OBDA framework *Ontop*, which supports all W3C standards related to OBDA, and which produces efficiently executable SQL queries.

Keywords: OBDA, SPARQL, RDF, OWL, R2RML, Ontop, Optique

1 Introduction

The *Ontology-Based Data Access* (OBDA) paradigm consists in exposing, for the purpose of querying, a conceptual view of the domain of interest, given in the form of an *ontology* that hides the structure of *the data sources*. Queries can then be posed over this high-level conceptual view, and end users no longer need an understanding of the data sources, the relation between them, or the encoding of the data. User queries are translated by the OBDA system into queries over one or multiple data sources. The ontology is connected to the data sources through *mappings*, a declarative specification that relates symbols in the ontology (classes and properties) to SQL views over data. The W3C standard R2RML [5] was created with the goal of providing a language for the specification of mappings in the OBDA setting. The ontology together with the mappings exposes a virtual RDF graph, which can be queried using SPARQL, the standard query language in the Semantic Web community. This RDF graph can be materialized, generating RDF triples that can be loaded into an RDF

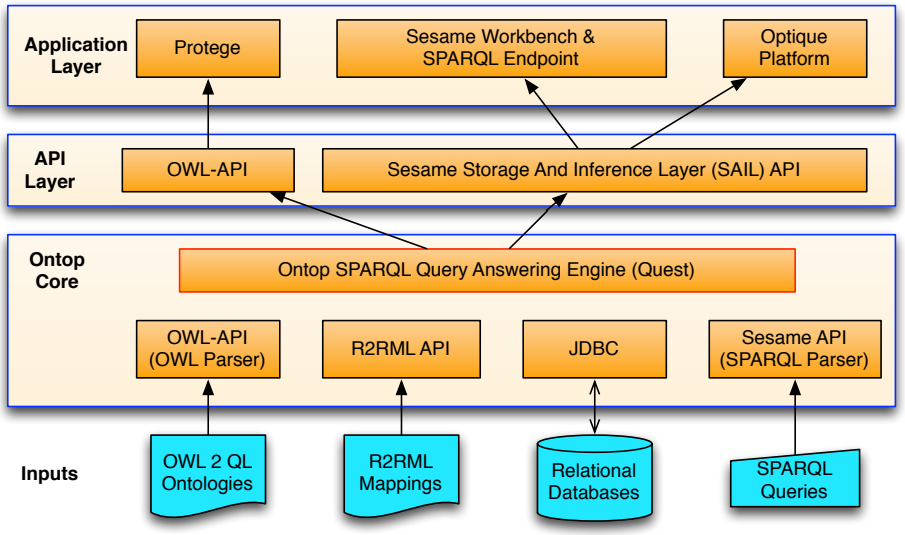


Fig. 1. Architecture of the *Ontop* framework

triplestore, or alternatively the RDF graph can be kept virtual and queried only during query execution. The virtual approach to OBDA avoids the cost of materialization and can benefit from the maturity of relational systems.

In this paper we introduce the *Ontop*[1] framework for OBDA. The system has solid theoretical foundations [4,13,14,17] and supports all the relevant W3C standards and major relational databases. It currently acts as the core query transformation module of the European project Optique[2].

The organization of the paper is as follows. Section 2 presents an overview of the *Ontop* framework. Section 3 gives a brief description of industrial applications of *Ontop* system within the Optique project, in particular, the Statoil and Siemens use cases. Finally, Section 4 concludes the paper.

2 The Ontop Framework

Ontop is a mature open-source OBDA framework developed at the Free University of Bozen-Bolzano and released under the Apache license. *Ontop* is the core component of the Optique platform. It enables querying virtual RDF graphs using SPARQL by translating the queries into SQL.

The architecture of *Ontop* is illustrated in Fig. 1. It is composed of four layers: (1) inputs (ontologies, mappings, databases, and queries), (2) *Ontop* core (query answering engine *Quest*), (3) high-level APIs, and (4) applications.

[1] `http://ontop.inf.unibz.it/`
[2] `http://optique-project.eu/`

2.1 Input Layer

Ontop supports all major relational databases, e.g., MySQL, DB2, Oracle, MS SQL Server, PostgreSQL, and H2. It also supports multiple W3C recommendations: SPARQL, OWL, R2RML, and SWRL. To the best of our knowledge, *Ontop* is the first OBDA system that supports all W3C standards related to OBDA.

We use the following running example to briefly illustrate how *Ontop* is structured and how it works.

Example 1. We consider a simplified online room booking service. One table named `tbl_booking` keeps data about bookings and has five columns: booking id (`bid`), guest name (`name`), a boolean flag (`shared`) indicating whether a room is shared or private, the type of the room (`type`), and its id (`rid`). Private rooms can only be the rooms with 1 or 2 single beds, or with 1 double bed. Shared rooms have 3 or 4 single beds. The type of the room is encoded as a positive integer value (`type`) as follows:

1–3 for a private room of types 1-single-bed, 1-double-bed, 2-single-beds, resp.;
4–5 for a shared room of types 3-single-beds and 4-single-beds, respectively.

For example, table `tbl_booking` may contain the following:

bid	name	shared	type	rid
1	'Alice'	false	1	4
2	'Bob'	true	5	5

■

Ontop supports the OWL 2 QL [12] and RDFS [2] ontology languages. In addition, it has been extended to support also a fragment of SWRL [17].

Example 2. The ontology for our simplified room booking service in Example 1 can be represented by the following OWL axioms:

```
   :Private rdfs:subClassOf :HotelRoom .
    :Shared rdfs:subClassOf :HotelRoom .
 :HotelRoom rdfs:subClassOf :Room .
:bookedRoom rdfs:domain :Booking .
:bookedRoom rdfs:range :Room .
:hasGuestName rdf:type owl:DatatypeProperty .
   :hasType rdf:type owl:ObjectProperty .
```

■

R2RML is a W3C standard language for mappings from relational databases to RDF datasets [5]. Besides R2RML, *Ontop* also supports its native mapping language, which is more compact and easier to use (the framework provides conversions between the two mapping languages). Intuitively, each mapping consists of *(i)* a source, which is an SQL query, and *(ii)* a target, which is an RDF triple pattern with placeholders for the values of attributes from the source query.[3] In

[3] See `https://github.com/ontop/ontop/wiki/ontopOBDAModel` for a detailed description of the *Ontop* mapping language.

real-world scenarios, the mappings are often bootstrapped from the database, following the W3C Direct Mapping Recommendation [1], or constructed semi-automatically by domain experts (see e.g., [16]).

Example 3. The mappings that relate the vocabulary of the ontology in Example 2 to the data source in Example 1 could be as follows:

```
:db1/{bid} rdf:type :Booking .
              ⟵ SELECT bid FROM tbl_booking
:db1/room/{rid} rdf:type :Private .
              ⟵ SELECT rid, type FROM tbl_booking
                   WHERE shared = false
:db1/room/{rid} rdf:type :Shared .
              ⟵ SELECT rid, type FROM tbl_booking
                   WHERE shared = true
:db1/{bid} :hasGuestName {name} .
              ⟵ SELECT bid, name FROM tbl_booking}
:db1/{bid} :bookedRoom :db1/room/{rid} .
              ⟵ SELECT bid, rid FROM tbl_booking
:db1/room/{rid} :hasType :4-Single-Beds .
              ⟵ SELECT rid FROM tbl_booking
                   WHERE type = 5
```

■

SPARQL is the standard query language for RDF, and *Ontop* supports SPARQL 1.0 and the OWL 2 QL entailment regime of SPARQL 1.1 [11].

Example 4. The following SPARQL query retrieves names of guests who booked 4-bed dormitories:

```
SELECT ?name WHERE {
      ?c rdf:type :Booking .
      ?c :hasGuestName ?name .
      ?c :bookedRoom ?room .
      ?room :hasType :4-Single-Beds . }
```

■

So far it can be seen that for writing a SPARQL query, the user does not have to know how types are encoded to the database.

2.2 Core Layer

Quest, the *Ontop* SPARQL query answering engine, is the core of the *Ontop* framework. In brief, *Quest* translates end user's SPARQL queries over virtual RDF graphs into SQL queries, which are then executed by the relational database engine. The workflow of *Ontop* can be divided into an offline and an online stage. The most critical task during start-up (the offline stage) is compiling the ontology into the mappings and generating the so-called T-mappings [14]. We illustrate the process using the mappings in Example 3 and the ontology in Example 2. Although there is no mapping for `:Room`, *Ontop* uses the fact `:HotelRoom` is a subclass of `:Room` and its own two subclasses, `:Shared` and `:Private`, have

mappings and thus, generate instances of `:Room`. Moreover, the range of property `:bookedRoom` is included in `:Room`, and so, its mapping also generates instances of `:Room`. By taking these three mappings into account and using the Semantic Query Optimization techniques (for details, see [14]), *Ontop* derives the following optimized T-mapping:

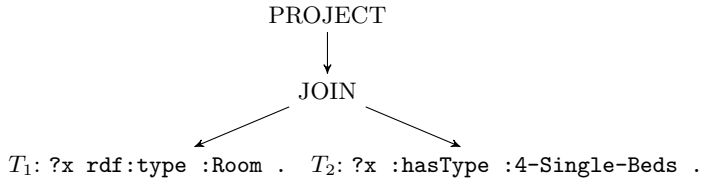
```
:db1/room/{rid} rdf:type :Room .
                    ← SELECT rid FROM tbl_booking
```

During query execution (the online stage), *Ontop* transforms an input SPARQL query into an optimized SQL query by exploiting the T-mappings and the database integrity constraints. *Ontop* decomposes the SPARQL query into a tree, and each node of the tree is transformed into an SQL expression. The following example illustrates the decomposition of the SPARQL query.

Example 5. Consider the SPARQL query

```
SELECT ?room WHERE {
    ?room rdf:type :Room ;
          :hasType :4-Single-Beds . }
```

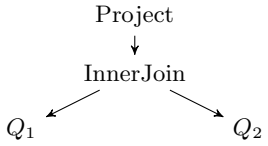
which is represented by the following tree:



■

Ontop traverses the SPARQL algebra tree in a bottom-up fashion and creates the list of nodes: in Example 5, the list is [T_1, T_2, JOIN, PROJECT]. First, *Ontop* replaces each leaf of the tree by the union of the SQL queries defining its predicate in the T-mappings. It then processes the intermediate nodes: it translates SPARQL operators (PROJECT, JOIN, FILTER, OPTIONAL, and UNION) into the corresponding SQL operators (Project, InnerJoin, Filter, LeftJoin, and Union).

Example 6. The translation of the query in Example 5 is the following:



The leaves, Q_1 and Q_2, are respectively the SQL definitions of the class `:Room` and of the property `:hasType` in the T-mapping rules (see above for the former and Example 3 for the latter). ■

During the translation of this intermediate representation of the query into SQL, critical optimizations are applied [15]. Their goal is to avoid redundant self-joins, sub-queries and joins over complex expressions.

Example 7. The resulting SQL query is the following:

```
SELECT Concat(':db1/room/', rid)
FROM tbl_booking
WHERE type = 5
```

■

Finally, the execution of SQL queries is delegated to RDBMS.

2.3 API Layer

Ontop implements two widely used Java APIs: OWL API and Sesame API. OWL API [7] is a Java interface for OWL which enables creating and managing OWL ontologies. Sesame [3] is an open-source RDF framework which offers querying and RDFS inferencing with the Sesame Storage and Inference Layer APIs. Both APIs are available as Maven artifacts.

2.4 Application Layer

The *Ontop* framework provides a plugin for the open-source ontology editor Protégé[4]. This plugin introduces a mapping editor and several additional features, such as R2RML mapping import/export, RDF triple materialization, and consistency checking of the ontology.

Ontop is also available as a SPARQL end-point through the Sesame Workbench.

3 Industrial Applications

The *Ontop* system is actively used both in academic and in industrial applications. It is used in many academic research projects[5] as an OBDA system for accessing and querying temporal and streaming data related to different domains (e.g., health, tourism). *Ontop* is also used in a joint project with industrial partners, called ACEPROM. This project achieves an automatic matching between job applicants and job offers.

An important usage of *Ontop* is for the use cases of Siemens Energy and Statoil, which are the industrial partners of the European project Optique. Siemens Energy maintains thousands of power generation facilities, in particular, gas and steam turbines. Approximately 2000 sensors are used to monitor the functioning of a single turbine. Siemens Energy provides operational support through more than 50 service centers. All of these centers are linked to a common database,

[4] http://protege.stanford.edu/

[5] https://github.com/ontop/ontop/wiki/UseCases/

which stores these massively increasing data streams coming from sensors in several thousands databases [8]. For each turbine around 150 tables are maintained. The daily amount of generated data is about 30 GB and the total size of tabulations of sensors and event data is in the order of hundreds of terabytes [10]. The main bottleneck is finding relevant data from these autonomously evolving and growing databases. The second industrial partner in the Optique project, Statoil, is an international gas and oil extraction company. Geology and geophysics experts of Statoil benefit from previously acquired geographical data in order to create stratigraphic models of unexplored areas. The volume of this geological data is in the order of several petabytes. This data is accessible through different schemata and more than 2000 tables [9].

In these two companies, only IT experts are able to directly express SQL queries, due to the complexity of the database schemas. Hence, domain experts could previously access data only through the mediation of IT experts. Thanks to the Optique platform, which is based on *Ontop*, they can now directly formulate the queries corresponding to their information needs through an intuitive visual query formulation interface, using the familiar ontology vocabulary [6].

4 Conclusion

Ontop is a mature OBDA system, which has been adopted by researchers and industry. It enables non-experts to formulate their queries over conceptual models using familiar domain terms. The system translates the user queries into efficiently executable SQL queries, and delegates their execution to a RDBMS. *Ontop* supports all the W3C standards related to OBDA and works with major RDBMSs.

Acknowledgements

This work was supported by the EU FP7 project Optique (grant n. 318338).

References

1. Arenas, M., Prud'hommeaux, E., Sequeda, J.: A direct mapping of relational data to RDF. W3C Recommendation, World Wide Web Consortium (Mar 2011), available at `http://www.w3.org/TR/2011/WD-rdb-direct-mapping-20110324/`
2. Brickley, D., Guha, R.V.: RDF vocabulary description language 1.0: RDF Schema. W3C Recommendation, World Wide Web Consortium (Feb 2004), available at `http://www.w3.org/TR/rdf-schema/`
3. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A generic architecture for storing and querying RDF and RDF schema. In: Proc. of the 1st Int. Semantic Web Conf. (ISWC). Lecture Notes in Computer Science, vol. 2342, pp. 54–68. Springer (2002)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. of Automated Reasoning 39(3), 385–429 (2007)

5. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF mapping language. W3C Recommendation, World Wide Web Consortium (Sep 2012), available at `http://www.w3.org/TR/r2rml/`
6. Giese, M., Calvanese, D., Haase, P., Horrocks, I., Ioannidis, Y., Kllapi, H., Koubarakis, M., Lenzerini, M., Möller, R., Özçep, Ö., Rodriguez-Muro, M., Rosati, R., Schlatte, R., Schmidt, M., Soylu, A., Waaler, A.: Scalable end-user access to Big Data. In: Akerkar, R. (ed.) Big Data Computing. CRC Press (2013)
7. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Semantic Web J. 2(1), 11–21 (2011)
8. Hubauer, T.M., Lamparter, S., Roshchin, M., Solomakhina, N., Watson, S.: Analysis of data quality issues in real-world industrial data. In: Proc. of the 2013 Annual Conf. of the Prognostics and Health Management Society (2013)
9. Kharlamov, E., Jiménez-Ruiz, E., Zheleznyakov, D., Bilidas, D., Giese, M., Haase, P., Horrocks, I., Kllapi, H., Koubarakis, M., Özçep, Ö.L., Rodriguez-Muro, M., Rosati, R., Schmidt, M., Schlatte, R., Soylu, A., Waaler, A.: Optique: Towards OBDA systems for industry. In: Revised Selected Papers of ESWC 2013 Satellite Events. Lecture Notes in Computer Science, vol. 7955, pp. 125–140. Springer (2013)
10. Kharlamov, E., Solomakhina, N., Özçep, Ö.L., Zheleznyakov, D., Hubauer, T., Lamparter, S., Roshchin, M., Soylu, A., Watson, S.: How semantic technologies can enhance data access at Siemens Energy. In: Proc. of the 13th Int. Semantic Web Conf. (ISWC). Lecture Notes in Computer Science, vol. 8796, pp. 601–619. Springer (2014)
11. Kontchakov, R., Rezk, M., Rodriguez-Muro, M., Xiao, G., Zakharyaschev, M.: Answering SPARQL queries over databases under OWL 2 QL entailment regime. In: Proc. of the 13th Int. Semantic Web Conf. (ISWC). Lecture Notes in Computer Science, vol. 8796, pp. 552–567. Springer (2014)
12. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language profiles (second edition). W3C Recommendation, World Wide Web Consortium (Dec 2012), available at `http://www.w3.org/TR/owl2-profiles/`
13. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. on Data Semantics X, 133–173 (2008)
14. Rodriguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: Ontology-based data access: Ontop of databases. In: Proc. of the 12th Int. Semantic Web Conf. (ISWC). Lecture Notes in Computer Science, vol. 8218, pp. 558–573. Springer (2013)
15. Rodriguez-Muro, M., Rezk, M.: Efficient SPARQL-to-SQL with R2RML mappings. Journal of Web Semantics: Science, Services and Agents on the World Wide Web (2015)
16. Skjæveland, M.G., Lian, E.H., Horrocks, I.: Publishing the Norwegian Petroleum Directorate's FactPages as Semantic Web data. In: Proc. of the 12th Int. Semantic Web Conf. (ISWC). Lecture Notes in Computer Science, vol. 8219, pp. 162–177. Springer (2013)
17. Xiao, G., Rezk, M., Rodriguez-Muro, M., Calvanese, D.: Rules and ontology based data access. In: Proc. of the 8th Int. Conf. on Web Reasoning and Rule Systems (RR). Lecture Notes in Computer Science, vol. 8741, pp. 157–172. Springer (2014)