OBDA with the Ontop Framework

Diego Calvanese\textsuperscript{1}, Benjamin Cogrel\textsuperscript{1}, Elem Gzel Kalayci\textsuperscript{1}, Sarah Komla Ebri\textsuperscript{1}, Roman Kontchakov\textsuperscript{2}, Davide Lanti\textsuperscript{1}, Martin Rezk\textsuperscript{1}, Mariano Rodriguez-Muro\textsuperscript{3}, and Guohui Xiao\textsuperscript{1}

\textsuperscript{1}Faculty of Computer Science, Free University of Bozen-Bolzano, Italy  
\textsuperscript{2}Dept. of Computer Science, Birkbeck, University of London, UK  
\textsuperscript{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 \cite{r2rml} 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
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 framework for OBDA. The system has solid theoretical foundations \[1,3,13,14,17\] 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

Octopus 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, Octopus is the first OBDA system that supports all W3C standards related to OBDA.

We use the following running example to briefly illustrate how Octopus 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:

<table>
<thead>
<tr>
<th>bid</th>
<th>name</th>
<th>shared</th>
<th>type</th>
<th>rid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>'Alice'</td>
<td>false</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>'Bob'</td>
<td>true</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Octopus 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, Octopus 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 See [https://github.com/ontop/ontop/wiki/ontopOBDAModel](https://github.com/ontop/ontop/wiki/ontopOBDAModel) for a detailed description of the Octopus 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:

\[
\begin{align*}
&\text{:db1/\{bid\} rdf:type :Booking .} \\
&\text{← SELECT bid FROM tbl_booking} \\
&\text{:db1/room/\{rid\} rdf:type :Private .} \\
&\text{← SELECT rid, type FROM tbl_booking WHERE shared = false} \\
&\text{:db1/room/\{rid\} rdf:type :Shared .} \\
&\text{← SELECT rid, type FROM tbl_booking WHERE shared = true} \\
&\text{:db1/\{bid\} :hasGuestName \{name\} .} \\
&\text{← SELECT bid, name FROM tbl_booking} \\
&\text{:db1/\{bid\} :bookedRoom :db1/room/\{rid\} .} \\
&\text{← SELECT bid, rid FROM tbl_booking} \\
&\text{:db1/room/\{rid\} :hasType :4-Single-Beds .} \\
&\text{← SELECT rid FROM tbl_booking WHERE type = 5}
\end{align*}
\]

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:

\[
\text{SELECT ?name WHERE }
\begin{align*}
&\{ ?c rdf:type :Booking .} \\
&\text{?c :hasGuestName ?name .} \\
&\text{?c :bookedRoom ?room .} \\
&\text{?room :hasType :4-Single-Beds .}
\end{align*}
\]

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:

\[
\text{:db1/room/\{rid\} rdf:type :Room} .
\]

\[←\text{SELECT \ rid \ FROM \ tbl\_booking}\]

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

\[
\text{SELECT } ?\text{room} \text{ WHERE } \{ \\
\text{ ?room rdf:type :Room} ; \\
\text{ :hasType :4-Single-Beds } . \} 
\]

which is represented by the following tree:

![Tree representation of the SPARQL query](image)

**Ontop** traverses the SPARQL algebra tree in a bottom-up fashion and creates the list of nodes: in Example 5, the list is \([T_1, T_2, \text{JOIN}, \text{PROJECT}]\). 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 (\text{PROJECT}, \text{JOIN}, \text{FILTER}, \text{OPTIONAL}, and \text{UNION}) into the corresponding SQL operators (\text{Project}, \text{InnerJoin}, \text{Filter}, \text{LeftJoin}, and \text{Union}).

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

![Tree representation of the SQL query](image)

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:

\[
\begin{align*}
\text{SELECT} & \quad \text{Concat('':db1\text{/room}/', rid)} \\
\text{FROM} & \quad \text{tbl_booking} \\
\text{WHERE} & \quad \text{type} = 5
\end{align*}
\]

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

2.3 API Layer

\textit{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 \textit{Ontop} framework provides a plugin for the open-source ontology editor Prot\'eg\'e\[4\]. 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.

\textit{Ontop} is also available as a SPARQL end-point through the Sesame Workbench.

3 Industrial Applications

The \textit{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). \textit{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 \textit{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,

\[\text{http://protege.stanford.edu/}\]

\[\text{https://github.com/ontop/ontop/wiki/UseCases/}\]
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


