Ontology-Based Data Access: A Survey

Guohui Xiao[1], **Diego Calvanese**[1], **Roman Kontchakov**[2], **Domenico Lembo**[3], **Antonella Poggi**[3], **Riccardo Rosati**[3] and **Michael Zakharyaschev**[2]

[1] KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano, Italy
[2] Department of Computer Science and Information Systems, Birkbeck, University of London, UK
[3] Dip. di Ing. Informatica Automatica e Gestionale, Sapienza Università di Roma, Italy

Abstract

We present the framework of ontology-based data access, a semantic paradigm for providing a convenient and user-friendly access to data repositories, which has been actively developed and studied in the past decade. Focusing on relational data sources, we discuss the main ingredients of ontology-based data access, key theoretical results, techniques, applications and future challenges.

1 Introduction

Ontology-based data access (OBDA, for short) is a semantic technology that has been developed since the mid 2000s [Poggi *et al.*, 2008] with the aim of facilitating access to various types of data sources. It originates in real-world scenarios such as the one outlined below (see `http://purl.org/slegge`).

Example 1 Statoil (Equinor), a Norwegian multinational oil&gas company, stores data in a large relational database (DB) *Slegge* with about 1500 tables (and 1700 views). Prior to making decisions on drilling wellbores, geologists at Statoil need to gather relevant information. For instance, geologists' information needs may include the following question:

(009) *In my area of interest, return all pressure data tagged with key stratigraphy information with understandable QC attributes* (*and suitable for further filtering*).

Translating such an information need into the standard database query language *SQL* is usually a big challenge for geologists, who are not supposed to know how Slegge is organised. In fact, the main table for wellbores has 38 columns; a four-table join with two additional filters is needed to obtain formation pressure for a wellbore, and stratigraphic information requires a join with 5 more tables. Using existing *SQL* templates and manipulating the answers is error prone, and calling an IT expert is time-consuming (it can take days or even weeks). OBDA offers a different approach to formalising and answering **(009)**. Domain experts at Statoil designed a Subsurface Exploration ontology (`SE`) that captures terms of the user information needs such as `Wellbore`, `hasFormationPressure`, etc. IT experts wrote a mapping that declaratively connects (through *SQL* queries) the ontology predicates to the Slegge DB. The task of the geologist now is to reformulate **(009)** in the vocabulary of `SE`—possibly using a visual query interface such as OptiqueVQS[1] —as a query in the W3C standard *SPARQL*[2], which could look as follows:

```
SELECT ?w ?depth ?strat_unit WHERE {
   ?w a :Wellbore . ?w :hasMeasurement ?p .
   ?p a :Pressure . ?p :hasDepth ?depth
   OPTIONAL { ?depth :inWellboreInterval ?strat_zone .
              ?strat_zone :hasUnit ?strat_unit } }.
```

This query retrieves all assignments to the variables `?w`, `?depth`, `?strat_unit` in the `SELECT` clause that satisfy the conditions of the `WHERE` clause. The latter consists of *triple patterns* 'subject-predicate-object' (separated by dots) required to match the data. The first four triple patterns say that `?w` is an instance of class `Wellbore` and has measurements, `?p`, which are instances of `Pressure` and have their `?depth` recorded by property `hasDepth`. The two triple patterns in `OPTIONAL` return additional information, if available, about the stratigraphic units `?strat_unit` of the wellbore intervals for the depth measurements. It is *optional* in the sense that the variable `?strat_unit` is assigned no value if the stratigraphic information is absent for the depth measurement.

An OBDA system would automatically rewrite this *SPARQL* query using the ontology and mapping to a *SQL* query over the DB, optimise it, and evaluate it by Slegge. ❑

In general, gathering information even from a company's DB is a hard task for non-IT-expert users. One of the main reasons is that DBs are usually designed to serve applications: their structure and meaning are obscure for most of the users; and the stored data is often redundant, mixed with information only needed to support company processes, and incomplete with respect to the business domain. Collecting, integrating, reconciling and efficiently extracting information from heterogeneous and autonomous data sources is regarded as a major challenge, with 'most companies [...] capturing only a fraction of the potential value from data and analytics'.[3]

The OBDA paradigm addresses this issue by providing ac-

[1] `http://optique-project.eu/training-programme/module-vqs`
[2] `http://www.w3.org/TR/sparql11-query`
[3] "The age of analytics: competing in a data-driven world", McKinsey Global Institute, December 2016.

cess to the data layer, consisting of autonomous *data sources* (e.g., DBs), through the mediation of a conceptual domain view, given in terms of an *ontology*, and the use of a declarative *mapping* between the data layer and the ontology. OBDA users do not have to know details of the data sources and can express their information needs as queries over the conceptual domain model. By applying knowledge representation and automated reasoning techniques, an OBDA system uses the ontology and mapping to reformulate the user queries into standard DB queries that are executed directly by the database management systems (DBMSs) of the sources. Thus, OBDA relies upon both KR&R and DB technologies.

OBDA systems implementing this paradigm include Mastro [Calvanese *et al.*, 2011], Morph [Priyatna *et al.*, 2014], Ontop [Calvanese *et al.*, 2017], Stardog[4] and Ultrawrap [Sequeda and Miranker, 2013]. They were adopted in many industrial projects and use cases, e.g., at Statoil and Siemens[5], the Italian Ministry of Economy and Finance [Antonioli *et al.*, 2014], in projects on Smart Cities [López *et al.*, 2015], Electronic Health Records [Rahimi *et al.*, 2014], and Manufacturing [Petersen *et al.*, 2017].

Over the past decade, the theory and practice of OBDA have become a hot topic in the areas of Knowledge Representation (Description Logics), Semantic Technologies and Databases, with numerous papers published in top CS journals (including AIJ, JACM, JAIR, TODS) and conferences, and deep connections with such prominent disciplines as Constraint Satisfaction and Circuit Complexity established.

In this brief survey, we introduce the framework of OBDA and discuss main results, techniques and challenges. We first describe the classical OBDA framework in Section 2. Then, in Section 3, we consider the process of query answering in OBDA. In Section 4, we focus on mapping management and analysis. In Section 5, we outline extensions of the classical OBDA framework. Finally, Section 6 discusses some of the most important research directions.

We assume the reader is familiar with the basics of databases (at a standard undergraduate DB course level).

2 OBDA Framework

We begin by presenting a formal framework for OBDA, distinguishing between the extensional (instance) and intensional (schema) levels. The former is given by a *source DB* $\mathcal{D}$ conforming to the data source schema $\mathcal{S}$ (which typically includes integrity constraints), and the latter by an *OBDA specification* $\mathcal{P} = (\mathcal{O}, \mathcal{M}, \mathcal{S})$, where $\mathcal{O}$ is an ontology, $\mathcal{S}$ a data source schema and $\mathcal{M}$ a mapping from $\mathcal{S}$ to $\mathcal{O}$ (signatures of the ontology and schema are disjoint). The role of $\mathcal{O}$ is to provide the users with a high-level conceptual view of the data and a convenient vocabulary for their queries; it can also enrich incomplete data with background knowledge.

Example 2 The Subsurface Exploration ontology (SE) in Example 1 contains, among others, the following axioms, given in description logic (DL) syntax [Baader *et al.*, 2017]:

$$\begin{aligned}
\texttt{FormationPressure} &\sqsubseteq \texttt{Pressure},\\
\texttt{FormationPressure} \sqcap \texttt{HydrostaticPressure} &\sqsubseteq \bot,\\
\texttt{hasFormationPressure} &\sqsubseteq \texttt{hasMeasurement},\\
\exists \texttt{hasFormationPressure}^{-}.\top &\sqsubseteq \texttt{FormationPressure},\\
\texttt{FormationPressure} &\sqsubseteq \exists \texttt{hasDepth}.\texttt{Depth}.
\end{aligned}$$

The first three are *inclusions* between, respectively, unary predicates (*concepts* in DL or *classes* in Semantic Web parlance) and binary predicates (*roles* or *properties*); their first-order (FO) equivalents look as follows:

$$\begin{aligned}
&\forall x\,(\texttt{FormationPressure}(x) \rightarrow \texttt{Pressure}(x)),\\
&\forall x\,(\texttt{FormationPressure}(x) \wedge \texttt{HydrostaticPressure}(x) \rightarrow \bot),\\
&\forall xy\,(\texttt{hasFormationPressure}(x,y) \rightarrow \texttt{hasMeasurement}(x,y)).
\end{aligned}$$

The fourth axiom restricts the range of hasFormationPressure, while the fifth involves existential quantification:

$$\begin{aligned}
&\forall xy\,(\texttt{hasFormationPressure}(y,x) \rightarrow \texttt{FormationPressure}(x)),\\
&\forall x\,(\texttt{FormationPressure}(x) \rightarrow \exists y(\texttt{hasDepth}(x,y) \wedge \texttt{Depth}(y))).
\end{aligned}$$

❑

The *mapping* $\mathcal{M}$ in $\mathcal{P}$ specifies how the ontology predicates are populated by data from the source DB. In the SE example, each wellbore, which is identified by the column IDENTIFIER in the WELLBORE table, is given an IRI (Internationalised Resource Identifier) of the form http://slegger.gitlab.io/data#Wellbore-n to represent the wellbore in the ontology; in the sequel, we omit the prefixes and shorten such IRIs to Wellbore-n. Then the mapping connecting SE to the Slegge database contains the assertion

```
SELECT IDENTIFIER FROM WELLBORE
WHERE REF_EXISTENCE_KIND = 'actual'
        ⇝ Wellbore(iri("Wellbore-", IDENTIFIER))
```

populating the class Wellbore with the answers to the *SQL* query to the left of $\rightsquigarrow$. In general, mapping assertions are of the form $\varphi(\boldsymbol{x}) \rightsquigarrow \psi(\boldsymbol{x})$, where $\varphi(\boldsymbol{x})$ and $\psi(\boldsymbol{x})$ are FO-formulas in the signatures of $\mathcal{S}$ and $\mathcal{O}$, respectively. In our examples, we use *SQL* queries to conveniently represent the formulas $\varphi(\boldsymbol{x})$ (recall that WELLBORE has 38 columns). A special function iri (of variable arity) is used in $\psi(\boldsymbol{x})$ to construct IRIs for ontology objects: the parameters of iri are strings and DB columns (variables in $\boldsymbol{x}$), and the value of an iri term is the concatenation of its parameter values.

The pair $(\mathcal{P}, \mathcal{D})$ of an OBDA specification $\mathcal{P}$ and a source DB $\mathcal{D}$ is called an *OBDA instance*. To define its semantics, let $\mathcal{M}(\mathcal{D})$ be the minimal set of atoms in the signature of $\mathcal{O}$ that satisfies $\psi(\boldsymbol{a})$, for all $\varphi(\boldsymbol{x}) \rightsquigarrow \psi(\boldsymbol{x})$ in $\mathcal{M}$ and all tuples $\boldsymbol{a}$ of constants in $\mathcal{D}$ such that $\varphi(\boldsymbol{a})$ holds in $\mathcal{D}$. For example, in the SE setting, if the table WELLBORE contains

IDENTIFIER	REF_EXISTENCE_KIND	...
16/1-29_S	actual	...
30/8-5	actual	...
33/10-12	planned	...

then the mapping will produce the following two ground atoms (corresponding to *ABox assertions* or *RDF triples*):

$$\texttt{Wellbore}(\texttt{Wellbore-16/1-29\_S}), \texttt{Wellbore}(\texttt{Wellbore-30/8-5}).$$

We call an FO-structure $\mathcal{I}$ over the signature of $\mathcal{O}$ a *model* of $(\mathcal{P}, \mathcal{D})$ and write $\mathcal{I} \models (\mathcal{P}, \mathcal{D})$, if $\mathcal{I} \models \mathcal{O}$ and $\mathcal{I} \models \mathcal{M}(\mathcal{D})$. Thus, the two ground atoms above form an FO-structure that

[4] http://www.stardog.com

[5] http://optique-project.eu/results-downloads

is a model of our example OBDA instance. The additional mapping assertion

```
SELECT WELLBORE.IDENTIFIER, PRESSURE.PRESSURE_S
FROM WELLBORE, PRESSURE
WHERE WELLBORE.REF_EXISTENCE_KIND = 'actual' AND
      WELLBORE.WELLBORE_S = PRESSURE.FACILITY_S
  ⇝ hasFormationPressure(iri("Wellbore-", IDENTIFIER),
                         iri("FP-", PRESSURE_S)),
```

which, for brevity, represents a join of three tables as a single 'table' PRESSURE, can produce the ABox assertion

$$\texttt{hasFormationPressure}(\texttt{Wellbore-16/1-29\_S}, \texttt{FP-1249}).$$

The ontology will then imply the ground atoms

$$\texttt{hasMeasurement}(\texttt{Wellbore-16/1-29\_S}, \texttt{FP-1249}),$$
$$\texttt{FormationPressure}(\texttt{FP-1249}),\ \texttt{Pressure}(\texttt{FP-1249}),$$

which will hold in every model of our OBDA instance. Every model will also have to satisfy atoms $\texttt{hasDepth}(\texttt{FP-1249}, a)$ and $\texttt{Depth}(a)$, for some (possibly unknown) a.

The most important inference task in OBDA is *query answering*. Given a query $\boldsymbol{q}(\boldsymbol{x})$ in the signature of $\mathcal{O}$ with answer variables $\boldsymbol{x}$, a tuple $\boldsymbol{a}$ of constants in $\mathcal{D}$ is called a *certain answer* to $\boldsymbol{q}(\boldsymbol{x})$ over $(\mathcal{P}, \mathcal{D})$ if $\mathcal{I} \models \boldsymbol{q}(\boldsymbol{a})$, for every model $\mathcal{I}$ of $(\mathcal{P}, \mathcal{D})$. In our running example, FP-1249 is a certain answer to the conjunctive query (cf. Example 1)

$$\exists x\, [\texttt{Wellbore}(x) \land \texttt{hasMeasurement}(x, y) \land \texttt{Pressure}(y)]. \quad (1)$$

Conjunctive queries (CQs) are essentially SELECT-PROJECT-JOIN *SQL* queries. We discuss the main inference task next.

3 OBDA Query Answering

To make query answering viable in practice, the OBDA paradigm relies on reducing the problem of finding certain answers to answering FO queries directly over the data. More precisely, given an OBDA specification $\mathcal{P} = (\mathcal{O}, \mathcal{M}, \mathcal{S})$, we say that a query $\boldsymbol{q}(\boldsymbol{x})$ is *FO-rewritable* if there is an FO query $\boldsymbol{q}'(\boldsymbol{x})$ such that, for every source DB $\mathcal{D}$ for $\mathcal{P}$, a tuple $\boldsymbol{a}$ is a certain answer to $\boldsymbol{q}(\boldsymbol{x})$ over $(\mathcal{P}, \mathcal{D})$ iff $\mathcal{D} \models \boldsymbol{q}'(\boldsymbol{a})$. The query $\boldsymbol{q}'(\boldsymbol{x})$ is called an *FO-rewriting of* $\boldsymbol{q}(\boldsymbol{x})$ (with respect to $\mathcal{P}$). Thus, in the context of the example in Section 2, the following *SQL* query is an FO-rewriting of (1):

```
SELECT "FP-" || PRESSURE_S FROM WELLBORE, PRESSURE
WHERE WELLBORE.REF_EXISTENCE_KIND = 'actual' AND
      WELLBORE.WELLBORE_S = PRESSURE.FACILITY_S,
```

where || is the string concatenation operation. Since the FO-rewriting of a query does not depend on the data, computing the certain answers to an FO-rewritable query has the same data complexity, viz. AC^0, as classical DB query evaluation. Recall that, under *data complexity*, the data instance is the only input to the query answering problem, while the OBDA specification and query are fixed [Vardi, 1982].

The traditional approach [Poggi *et al.*, 2008] to computing FO-rewritings of a query $\boldsymbol{q}(\boldsymbol{x})$ with respect to an OBDA specification $\mathcal{P} = (\mathcal{O}, \mathcal{M}, \mathcal{S})$ proceeds in two stages. In Stage 1, the *ontology-mediated query* (OMQ) $(\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$ is rewritten into an 'equivalent' FO-query $\boldsymbol{q}'(\boldsymbol{x})$ in the signature of $\mathcal{O}$. In Stage 2, $\boldsymbol{q}'(\boldsymbol{x})$ is unfolded using the mapping $\mathcal{M}$ into an FO-query in the signature of $\mathcal{S}$, which gives the required FO-rewriting of $\boldsymbol{q}(\boldsymbol{x})$ with respect to $\mathcal{P}$. To ensure FO-rewritability, the ontology and mapping languages need to be chosen with care.

3.1 Ontology-Mediated Query Rewriting

OWL 2 QL is a profile[6] of the ontology language *OWL 2*, standardised by the W3C, that has been specifically designed for OBDA. It is based on the *DL-Lite* family [Calvanese *et al.*, 2007; Artale *et al.*, 2009], a suite of DLs closely related to conceptual modelling formalisms for DB design and software engineering. The *DL-Lite* constructs are illustrated by Example 2. *OWL 2 QL* was tailored to ensure FO-rewritability of OMQs: if $\mathcal{O}$ is an *OWL 2 QL* ontology, then any CQ $\boldsymbol{q}(\boldsymbol{x})$ is FO-rewritable with respect to the OBDA specification $\mathcal{P}_\mathcal{O}$ with ontology $\mathcal{O}$ and schema containing a unary table for each class and a binary table for each property, whose mapping is an isomorphism. An FO-rewriting of $\boldsymbol{q}(\boldsymbol{x})$ with respect to $\mathcal{P}_\mathcal{O}$ is called an *FO-rewriting* of the OMQ $(\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$, which is also said to be *FO-rewritable*.

The first OMQ rewriting algorithm PerfectRef [Calvanese *et al.*, 2007] was essentially based on backward chaining. To illustrate, consider the fragment $\mathcal{O}$ of the SE ontology in Example 2 and the CQ $\boldsymbol{q}(y)$ given by (1). PerfectRef rewrites OMQ $(\mathcal{O}, \boldsymbol{q}(y))$ into the *union* of the following CQs (UCQ):

$$\exists x\, [\texttt{Wellbore}(x) \land \texttt{hasMeasurement}(x, y) \land \texttt{Pressure}(y)],$$
$$\exists x\, [\texttt{Wellbore}(x) \land \texttt{hasMeasurement}(x, y) \land \texttt{FormationPressure}(y)],$$
$$\exists x\, [\texttt{Wellbore}(x) \land \texttt{hasFormationPressure}(x, y) \land \texttt{Pressure}(y)],$$
$$\exists x\, [\texttt{Wellbore}(x) \land \texttt{hasFormationPressure}(x, y) \land \texttt{FormationPressure}(y)],$$
$$\exists x\, [\texttt{Wellbore}(x) \land \texttt{hasFormationPressure}(x, y)].$$

First experiments with Mastro [Calvanese *et al.*, 2011] revealed that rewritings produced by PerfectRef were often prohibitively large for execution by DBMSs, which spurred the investigation of various OMQ rewriting techniques and optimisations. For example, optimisations based on CQ containment can significantly reduce the size of rewritings (the last CQ above subsumes the two preceding CQs); see, e.g., the work by Mora and Corcho [2013] and references therein for advances in this direction. Also, the UCQ above can be represented succinctly as a positive existential formula:

$$\exists x\, \big(\texttt{Wellbore}(x, y) \land [\texttt{hasFormationPressure}(x, y) \lor (\texttt{hasMeasurement}(x, y) \land (\texttt{Pressure}(y) \lor \texttt{FormationPressure}(y)))]\big).$$

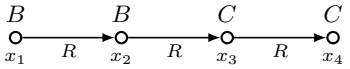
It turns out, however, that in the worst case, rewritings can be of exponential size even if represented more succinctly as positive existential formulas or non-recursive Datalog programs; rewritings in the form of arbitrary FO-formulas can be of superpolynomial size unless $\mathrm{NP} \subseteq \mathrm{P/poly}$. For a comprehensive study of the succinctness problem and the combined complexity of OMQ answering (depending on the shape of CQs and the existential depth of ontologies), consult [Bienvenu *et al.*, 2018], which also provides further references to various types of OMQ rewritings developed so far.

Ontology languages for which not all OMQs are *uniformly* FO-rewritable (because of LOGSPACE-hardness) were first

[6] http://www.w3.org/TR/owl2-profiles

studied in 2006 [Calvanese *et al.*, 2013], recasting also early results by Schaerf [1993] on DLs with coNP-hard query answering. Typical examples of non-FO-rewritable OMQs are

$$(\{\forall xy\,(R(x,y) \land A(y) \to A(x))\},\ \ A(x)) \quad \text{and}$$
$$(\{\forall x\,(A(x) \to B(x) \lor C(x))\},\ \ \exists \boldsymbol{x}\,\varphi(\boldsymbol{x})),$$

where $\varphi(x_1,x_2,x_3,x_4)$ is 

The former OMQ, expressible in the DL $\mathcal{EL}$, encodes NL-hard digraph reachability, while the latter, expressible in $\mathcal{ALU}$, is coNP-complete [Gerasimova *et al.*, 2017].

3.2 Unfolding Rewritings with Mappings

The FO-rewriting $\boldsymbol{q}'(\boldsymbol{x})$ of an OMQ can be made executable over the source DB by means of *unfolding*. Consider first GAV (*global-as-view*) mappings, in which the assertions are of the form $\varphi(\boldsymbol{y}) \rightsquigarrow S(\boldsymbol{y})$ for a predicate S (without iri-terms). In this case, unfolding boils down to replacing each atom $S(\boldsymbol{z})$ in $\boldsymbol{q}'(\boldsymbol{x})$ by the query $\varphi(\boldsymbol{z})$ from the mapping [Lenzerini, 2002] (which is similar to expanding a query with views in DBs or partial evaluation of Datalog programs).

The W3C standardised *R2RML*[7] as a language for mapping relational DBs to RDF graphs (sets of RDF triples), where mapping assertions are of the form $\varphi(\boldsymbol{x}) \rightsquigarrow \psi(\boldsymbol{x})$, for a conjunction $\psi(\boldsymbol{x})$ of atoms over variables $\boldsymbol{x}$ and iri-terms (expressed by means of IRI templates). Interestingly, De Giacomo *et al.* [2018] show that iri-terms can be used to encode more general GLAV mappings, where the right-hand side query is a full-fledged CQ [Lenzerini, 2002]. This result heavily depends on the lack of functional properties and the unique name assumption (UNA) in *OWL 2 QL* [Calvanese *et al.*, 2008]. In spite of this expressive power, all CQs are FO-rewritable with respect to OBDA specifications with *OWL 2 QL* ontologies and *R2RML* mappings [Poggi *et al.*, 2008]: indeed, any OMQ rewriting can be unfolded with an *R2RML* (or, equivalently, GLAV) mapping by using careful unification for query fragments [Calvanese *et al.*, 2012].

Unfolding an FO-rewriting of an OMQ can result in an exponential blowup. For example, the class FormationPressure with the (simplified) mapping assertion

$$\texttt{SELECT PRESSURE\_S FROM PRESSURE} \rightsquigarrow \texttt{FormationPressure}(\texttt{iri}(\texttt{"FP-"}, \texttt{PRESSURE\_S}))$$

includes (as a subset) the range of hasFormationPressure; cf. Example 2. Thus, the rewriting of a CQ with an atom $\texttt{FormationPressure}(y)$ will have to 'expand' it both with itself and with $\texttt{hasFormationPressure}(z,y)$ for some z. Unfolding will then introduce a redundant *SQL* subquery for $\texttt{hasFormationPressure}(z,y)$. Indeed, this subquery gives no new answers compared to the subquery for $\texttt{FormationPressure}(y)$ because the latter subsumes the former, which selects only those tuples in PRESSURE that have matches in WELLBORE. The problem is exacerbated in the real SE because PRESSURE is a join of three tables. Kontchakov *et al.* [2014] and Sequeda *et al.* [2014] observed that optimisations removing such redundancies in rewritings of OMQs need to be made only once (as offline preprocessing) by combining mapping and ontology. Indeed, the system can produce a mapping assertion for FormationPressure that gives *all certain answers* to $\texttt{FormationPressure}(y)$, so that its rewriting does not have to include predicates for redundant subqueries. Such mappings are called *saturated* or *T-mappings*.

Even with optimised saturated mappings, the *SQL* queries produced by unfolding may contain many redundant self-joins and unions. Indeed, in the SE setting, data property name and object property hasDepth for the class FormationPressure are populated from columns IDENTIFIER and PRESSURE_S of the same table PRESSURE. Hence, the CQ

$$\texttt{FormationPressure}(x) \land \texttt{name}(x,y) \land \texttt{hasDepth}(x,z)$$

is naturally unfolded into a join of 3 copies of table PRESSURE:

```
SELECT "FP-" || P1.PRESSURE_S, P2.IDENTIFIER
       "PressureMeasuredDepth-" || P3.PRESSURE_S,
FROM PRESSURE P1, PRESSURE P2, PRESSURE P3
WHERE ("FP-" || P1.PRESSURE_S) = ("FP-" || P2.PRESSURE_S)
  AND ("FP-" || P1.PRESSURE_S) = ("FP-" || P3.PRESSURE_S).
```

Such redundancies can be detected and removed using integrity constraints of the DB schema (e.g., the primary key PRESSURE_S of PRESSURE above [Kontchakov *et al.*, 2014]), or using additional constraints in the OBDA specification [Di Pinto *et al.*, 2013; Hovland *et al.*, 2016] (industrial DBs often contain few integrity constraints because they may impact system performance). We note that most of such semantic optimisations are well-known in DB theory; however, they are either not implemented in the state-of-the-art RDMSs (partly because naively unfolded *SQL* queries often contain obvious redundancies, such as the self-join above, not expected in human-written *SQL* queries) or use assumptions specific to OBDA (e.g., *SPARQL* joins are translated into joins of concatenated strings for the IRIs rather than standard database joins over columns as in the example above).

A promising direction in OBDA query optimisation is estimation of the evaluation cost for alternative equivalent forms of unfolded and rewritten queries in order to choose the best candidate; this can be done for OMQ answering [Bursztyn *et al.*, 2015] or OBDA [Lanti *et al.*, 2017].

4 Mapping Management and Analysis

Besides efficient query answering algorithms, design-time support is also crucial in OBDA, since the construction, debugging and maintenance of an OBDA specification are particularly demanding. Mapping creation and management is probably the most complicated OBDA design-time task, as the mapping specifies the semantics of the data sources in terms of the ontology, and so bridges the typically large conceptual gap between the source schema and the ontology. On the other hand, the form of the mapping and its interaction with the ontology affect query answering performance, and so should be taken into account by OMQ rewriting algorithms (cf., e.g., mapping saturation by Kontchakov *et al.* [2014]).

The first approach to formal analysis of OBDA mappings was proposed by Lembo *et al.* [2015], who focused on identifying inconsistencies and redundancies. A mapping $\mathcal{M}$ is called *inconsistent* with an ontology $\mathcal{O}$ if simultaneously activating all of its assertions leads to a contradiction in $\mathcal{O}$.

[7] http://www.w3.org/TR/r2rml

Example 3 The axioms $\texttt{Well} \sqsubseteq \texttt{Asset}$, $\texttt{Well} \sqsubseteq \texttt{Facility}$, and $\texttt{Well} \sqcap \exists \texttt{isOutsourcedTo}.\top \sqsubseteq \bot$ state that a well is both an asset and a company's facility, and it is not outsourced to an external operator. Consider the following mapping $\mathcal{M}$:

```
SELECT IDENTIFIER I FROM FACILITY WHERE KIND_S = 'WELL'
    ⇝ Well(iri("Facility-", I)),
SELECT IDENTIFIER I, SUPPLIER S FROM FACILITY
    ⇝ isOutsourcedTo(iri("Facility-",I), iri("Op-",S)).
```

It is easy to see that $\mathcal{M}$ is inconsistent since every time the second assertion produces facts instantiating the ontology (i.e., it is activated), the first one is also activated, causing a violation of disjointness of $\texttt{Well}$ and $\exists \texttt{isOutsourcedTo}.\top$. ❑

A mapping $\mathcal{M}'$ is *redundant* for an OBDA specification $\mathcal{P} = (\mathcal{O}, \mathcal{M}, \mathcal{S})$ if adding the assertions in $\mathcal{M}'$ to $\mathcal{P}$ does not change its semantics, i.e., $(\mathcal{P}, \mathcal{D})$ and $((\mathcal{O}, \mathcal{M} \cup \mathcal{M}', \mathcal{S}), \mathcal{D})$ have the same models for every source DB $\mathcal{D}$. Besides the *global* notions of consistency and redundancy, Lembo *et al.* [2015] also consider their *local* counterparts, where the focus is on individual mapping assertions. They study the computational complexity of deciding both local and global consistency and redundancy for ontologies in *OWL 2* and its tractable profiles, and different forms of mappings (GLAV or GAV). It turns out that mapping analysis can be carried out by composing standard reasoning tasks for the ontology and data sources, and is indeed not harder than these standard tasks.

The form of mapping analysis presented by Bienvenu and Rosati [2016] consists of a query-based notion of entailment and equivalence between OBDA specifications. In particular, two OBDA specifications are regarded as equivalent if they give the same answers to the same queries (in a certain class), for all possible source DBs. The paper studies the complexity of deciding entailment and equivalence between OBDA specifications with respect to different classes of queries (CQs or instance queries), ontology languages (of the *DL-Lite* family) and forms of mappings.

Lembo *et al.* [2017] focus on the evolution of OBDA specifications and consider the scenario where the ontology and/or the source schema change. This is a typical situation since, in applications, new data sources (or new portions of data sources) are usually incrementally added to an existing OBDA installation; moreover, the ontology often needs to be updated in the light of a deeper understanding of the domain of interest. In these cases, the mapping may have to be modified to restore consistency. Two notions of *mapping repair* are proposed. The first, called DM-Repair, considers as mapping repairs all maximal subsets of the original mapping that are consistent with the updated ontology and source schema. In Example 3, $\mathcal{M}$ is inconsistent with $\mathcal{O}$, and its repairs are the two mappings containing a single assertion each. The second notion, called EM-Repair, aims at maximising preservation of the information inferred by the original OBDA specification that is still consistent after the update. In Example 3, EM-Repairs are the DM-Repairs augmented with the following inferred mapping assertion:

```
SELECT IDENTIFIER I FROM FACILITY WHERE KIND_S = 'WELL'
    ⇝ Asset(iri("Facility-", I)).
```

Lembo *et al.* [2017] study the data and combined complexity of CQ entailment under the aforementioned notions of repair for *OWL 2 QL* ontologies and various forms of mappings.

We note that the works on mapping management discussed above do not make any assumptions on the way the mapping is originally designed. In order to support this activity, some approaches have recently been proposed, whose purpose is to automatically bootstrap the ontology and the mapping from a DB [Sequeda *et al.*, 2011]. Typically, the bootstrapped ontology strongly depends on the abstraction level of the DB schema, which is rather low in real scenarios due to the fact that the database in practice plays the role of a (persistent) data structure for the enterprise applications. Consequently, the ontology needs to be manually refined to obtain a better conceptualisation of the domain of interest. BOOTOX [Jiménez-Ruiz *et al.*, 2015] provides some functionalities to help the designer improve the bootstrapped ontology by, e.g., aligning it to an existing domain ontology.

5 Extensions

We now survey extensions to the components of the OBDA framework that have been proposed in order to increase its expressive power and the scope of applications.

SPARQL, which has been adopted as the de facto standard query language in OBDA, was designed to deal with incomplete information (as exemplified by `OPTIONAL` in Example 1, where the variable `?strat_unit` is not necessarily assigned a value). Its standard semantics is based on graph matching: the basic graph patterns (BGPs) of the query are matched to the data (which corresponds to CQ evaluation in classical DBs), and then the resulting sets of answers are combined using *SPARQL* operators such as `OPTIONAL` and `UNION`. Xiao *et al.* [2018b] develop an efficient translation of a large part of SPARQL into SQL, which uses column nullability in optimisations. In *SPARQL 1.1*, the entailment regimes were defined to account for reasoning in the ontological layer [Glimm and Krötzsch, 2010], a crucial aspect in OBDA. For example, the *OWL 2* Direct Semantics entailment regime replaces graph matching for BGPs with entailment by a given *OWL 2* ontology. This approach leads to a modular implementation of reasoning in existing *SPARQL* engines. However, it also results in some counter-intuitive behaviour due to the lack of interaction between the certain answer semantics of BGPs and the `OPTIONAL` operator. Kostylev and Cuenca Grau [2015] and Ahmetaj *et al.* [2016] propose more intuitive semantics for the *well-designed fragment* of *SPARQL*.

Bag semantics. According to the W3C specifications, every *RDF* graph $\mathcal{M}(\mathcal{D})$ is a *set* of triples, but *SPARQL* queries are evaluated under the *bag* semantics. However, the current OBDA research mostly adopts the set semantics for answering *SPARQL* queries. On the other hand, the bag semantics is important, in particular, for DB-style aggregate queries. Nikolaou *et al.* [2017] propose an alternative bag semantics for OBDA, where duplicate triples in $\mathcal{M}(\mathcal{D})$ are retained. They show that although such semantics makes answering CQs CONP-hard in general (and so not FO-rewritable), there is a large class of CQs rewritable to BALG, a generalisation of the relational algebra to bags.

Expressive Ontologies. The original OBDA paradigm relies on FO-rewritability of *all* OMQs with an *OWL 2 QL* ontol-

ogy. There are different approaches to extending the expressive power of this *DL-Lite*-based ontology language. One was suggested by the DB community, which aimed at overcoming the restriction of DLs to unary and binary predicates only and designed various FO-rewritable fragments of the language of tuple-generating dependencies (aka Datalog$^{\pm}$ or existential rules)—that is, the standard language for DB constraints [Krötzsch and Rudolph, 2011; Gottlob *et al.*, 2014; König *et al.*, 2015]. Another type of limitation, pointed out by the DL community, is that real-world ontologies often use constructs that are not available in *DL-Lite*, for example, $\exists R.C$ on the left-hand side of concept inclusions or $C \sqcup C'$ on the right-hand side. Adding such constructs to *DL-Lite* would ruin the *uniform* FO-rewritablity of all OMQs; cf. Section 3.1. However, some useful OMQs with ontologies in an expressive language may still be FO-rewritable. A systematic investigation of the data complexity of answering individual OMQs in this *non-uniform* approach was launched by Lutz and Wolter [2017] and Bienvenu *et al.* [2014], who show, in particular, that FO- and Datalog-rewritability of expressive OMQs can be decided in exponential time. For recent results and further references, see the work by Hernich *et al.* [2017] and Lutz and Sabellek [2017]. The latter, for example, establish an AC^0/NL/P data complexity trichotomy for OMQs with $\mathcal{EL}$ ontologies and atomic queries.

The approach of Botoeva *et al.* [2016] aims at extending OBDA to more expressive ontology languages while still leveraging the underlying relational technology for query answering. It does so by encoding part of the domain semantics of rich ontology languages in the mapping layer. More precisely, by replacing ontology axioms by additional mapping assertions, an OBDA specification with an expressive ontology is rewritten to an equivalent one with an *OWL 2 QL* ontology, if possible, and approximated otherwise.

A more radical way of increasing expressiveness is to allow OMQ rewritings into query languages with polynomial (as opposed to AC^0) data complexity, e.g., Datalog; a recent survey is provided by Bienvenu and Ortiz [2015].

Rules. An alternative approach to encoding more domain semantics in the ontology is to extend the OBDA paradigm with Datalog-style rules. For example, in the context of the SE ontology, the rule

$$\texttt{intervalPerm}(i, v) \leftarrow \texttt{extractedFrom}(c, i),\ \texttt{hasCoreSample}(c, s), \texttt{hasPerm}(s, p), \texttt{valueInStdUnit}(p, v)$$

says that permeability of a wellbore interval can be obtained by chaining four roles (binary predicates), while the two rules

$$\texttt{ancestorUnitOf}(x, y) \leftarrow \texttt{parentUnitOf}(x, y),$$
$$\texttt{ancestorUnitOf}(x, y) \leftarrow \texttt{parentUnitOf}(x, z), \texttt{ancestorUnitOf}(z, y)$$

define ancestorUnitOf by means of linear recursion on the binary relation parentUnitOf. Xiao *et al.* [2014] extend the classical query rewriting algorithm for *SPARQL* queries to deal with such rules by exploiting *recursive common table expressions* introduced in *SQL:1999*.

Spatial OBDA. The OGC standard *GeoSPARQL* query language[8] is an extension of *SPARQL* with geospatial features. It defines, in particular, a set of rules for transforming *qualitative* spatial queries into equivalent *quantitative* ones. For example, the qualitative *RDF* triple o_1 geo:sfContains o_2 is true iff the geometry of object o_1 contains the geometry of o_2. So, given the *GeoSPARQL* query

```
SELECT ?fa ?wp WHERE {
  ?fa a :FieldArea . ?wp a :WellborePoint .
  ?fa geo:sfContains ?wp },
```

we obtain a *SPARQL* query with a union that contains, among others, the quantitative query that explicitly extracts geometries and relates them by the built-in function geo:contains:

```
SELECT ?fa ?wp WHERE {
  ?fa a :FieldArea . ?fa geo:hasGeometry ?gfa .
  ?wp a :WellborePoint . ?wp geo:hasGeometry ?gwp .
  ?gfa geo:asWKT ?wktfa . ?gwp geo:asWKT ?wktwp .
  FILTER geo:contains(?wktfa,?wktwp) }.
```

GeoSPARQL is supported by the approach proposed by Bereta and Koubarakis [2016], which relies on OBDA to access geospatial relational DBs (e.g., PostGIS, Oracle).

Temporal OBDA. Facilitating access to temporal (in particular, streaming) data has recently become a hot topic in the OBDA community. A typical example where temporal OBDA can be of great practical help is monitoring and analysing complex events based on timestamped sensor measurements stored in DBs. For instance, engineers at Siemens Remote Diagnostic Centres could be interested in active power trips of gas turbines, that is, events when the active power of a turbine was above 1.5MW for 10 seconds, and within 3 seconds after that there was a one-minute period when the active power was below 0.15MW.

Two approaches to extending the classical OBDA languages with temporal constructs capable of capturing such events have been proposed. One approach suggested by Klarman and Meyer [2014], Borgwardt *et al.* [2015] and Kharlamov *et al.* [2017] is to retain *OWL 2 QL* as the ontology language under the assumption that the ontology axioms hold at all times, but enrich the query language with constructs from a standard temporal logic such as the linear temporal logic *LTL*, Halpern-Shoham's interval logic *HS* or the metric temporal logic *MTL*. For example, using *MTL*, one can encode the active power trip of a turbine v as the query

$$\boldsymbol{q}(v) \;=\; \texttt{Turbine}(v) \wedge \Box^{-}_{[0,60]}\texttt{AP\_Below0.15}(v) \wedge \Diamond^{-}_{[60,63]}\Box^{-}_{[0,10]}\texttt{AP\_Above1.5}(v),$$

where $\Box^{-}_{[t_1,t_2]}A$ (respectively, $\Diamond^{-}_{[t_1,t_2]}A$) holds at moment of time t iff A holds everywhere (respectively, somewhere) in the interval $[t - t_2, t - t_1]$. Although this approach usually preserves the data complexity and FO-rewritability of OMQs (see, e.g., [Baader *et al.*, 2015]), it presupposes that users are capable of capturing complex events in temporal logic.

Alternatively and in the spirit of OBDA, one can shift the burden of representing complex temporal events to domain experts by adding temporal operators to ontology languages

[8] http://www.opengeospatial.org/standards/geosparql

and keeping queries simple. For example, the *ontology rule*

$$\texttt{AP\_Trip}(v) \leftarrow \Box^{-}_{[0,60]}\texttt{AP\_Below0.15}(v) \wedge \Diamond^{-}_{[60,63]}\Box^{-}_{[0,10]}\texttt{AP\_Above1.5}(v), \quad (2)$$

which holds at all times, would reduce $\boldsymbol{q}(v)$ to a simple query $\texttt{Turbine}(v) \wedge \texttt{AP\_Trip}(v)$. Unfortunately, temporal operators often increase the complexity of OMQ answering and ruin FO-rewritability. Finding a trade-off between complexity and practically useful expressive power of temporal ontology languages remains a challenge; Artale *et al.* [2017] provide a recent survey. Brandt *et al.* [2017] and Kontchakov *et al.* [2016] demonstrate on a few real-world use cases sufficient expressiveness of non-recursive Datalog queries with *MTL* and *HS* rules such as (2) and reasonable scalability of their *SQL* rewritings (with window functions and aggregation).

Identifier Management for Data Integration. When the underlying data in OBDA is actually stored in a collection of DBs that need to be queried in an integrated way, we speak of *ontology-based data integration*. An important aspect that differentiates it from OBDA is the fact that the same conceptual entity (object) may be represented in different DBs by different identifiers. In order to express equivalence between entity identifiers, Calvanese *et al.* [2015] exploit the `owl:sameAs` construct in mappings. In addition, entities can be assigned *canonical identifiers* to avoid redundant query answers caused by `owl:sameAs` [Xiao *et al.*, 2018a].

Materialisation and the Combined Approach. We described the classical approach to OBDA above, where no additional data is stored and used for query answering. This is also called *virtual OBDA* because $\mathcal{M}(\mathcal{D})$ is virtual and not materialised. When the data may actually be extended, the DB engine can, for efficiency reasons, e.g., materialise some of the queries in the saturated mapping. Indeed, such materialised views can lead to drastically simpler *SQL* queries [Sequeda *et al.*, 2014]. In the extreme, an OBDA system could attempt to materialise all logical consequences of the data, ontology and mapping, or, in DB parlance, to chase the data under the tuple-generating dependencies of the mapping and ontology [Abiteboul *et al.*, 1995, Sections 8.4 and 10.2]. For most of the DL-based ontology languages, however, this is not possible as the chase is infinite. Still, even then the chase can be represented as a finite structure provided that queries are appropriately modified and/or their answers are filtered [Lutz *et al.*, 2013].

6 Perspectives

We conclude by identifying a few important directions for future research, both in theory and practice.

Data Quality. The problem of assessing the overall quality of data requires measuring various *data quality dimensions* such as *consistency* (i.e., coherency with business rules), *completeness* (i.e., data contains the information needed for the task at hand) and *currency* (or freshness). When data comes from multiple independent data sources, OBDA provides a formal means to base data quality dimensions on a common ground, i.e., the domain ontology. Console and Lenzerini [2014] define a framework for data consistency in OBDA, considering coherence with the ontology axioms of both the content of the sources (extensional level) and their schema (intensional level). Algorithms and a complexity analysis are presented for checking different aspects of consistency for various classes of OBDA specifications. This investigation lays the foundations of a formal approach to data quality, focusing on the semantics of data sources. It remains to be broadened to other data quality dimensions beyond consistency.

Updates. With the exception of some preliminary efforts [De Giacomo *et al.*, 2017], the OBDA framework has been mostly considered as read-only up to now. Obviously, the capability of the framework to react to insertion, removal or change of logically implied facts or axioms is a problem that deserves a systematic investigation.

Benchmarking. In order to assess the performance of OBDA query answering and understand the effectiveness of optimisation techniques, some benchmarks have been developed: e.g., [Lanti *et al.*, 2015; Hovland *et al.*, 2017] and the collection at `obda-benchmark.org`. Still, more challenging benchmarks, with complex ontologies, mappings and large data instances are needed, as well as benchmarks with tuneable components, which would allow one to study the impact of each of the components.

Non-Uniform OBDA. An ideal solution to the perennial expressiveness vs. complexity and rewritability problem would be an OBDA system that, for any query and OBDA specification given in expressive languages, could check the data complexity of query answering, identify a suitable type of rewriting and compute it. However, it still remains to be seen whether such algorithms are feasible in practice and whether simpler (syntactic) sufficient conditions of FO-rewritability exist even for practically interesting classes of OMQs [Kaminski *et al.*, 2016; Hansen and Lutz, 2017]. Another challenge is utilising mappings and DB integrity constraints, which restrict the class of possible data instances and thereby can drastically simplify rewritings.

Streaming Data. Recent years have witnessed a huge increase in the amount of streaming data that needs to be processed in real-time, also in connection to the growth of the Internet of Things. This affects data management technologies in general, but also Semantic Web technologies, which need to be extended so as to deal with streaming data efficiently; cf. the W3C RDF Stream Processing Community Group[9]. OBDA can provide the currently missing link between streams of raw (generally non-RDF) data and its high-level view in terms of RDF triples, which would allow interoperability with the Semantic Web infrastructure. Calbimonte *et al.* [2010] describe an early proposal in this direction; Kharlamov *et al.* [2017] make first steps in developing analytics-aware temporal OBDA over streaming data.

Data Analytics. OBDA can be used to provide access to numerical data, representing, e.g., time, temperature or speed, using standard ontologies for these specific domains. Such structures are useful in formulating typical analytical tasks at a higher level of abstraction. This calls for extending the OBDA paradigm to the different types of numerical data, so as to support not only data access but also data analytics.

[9] `http://www.w3.org/community/rsp`

Acknowledgements

We thank the reviewers for their suggestions. This work was supported by the OBATS project at the Free Univ. of Bozen-Bolzano and by the Euregio (EGTC) IPN12 project KAOS.

References

[Abiteboul *et al.*, 1995] S. Abiteboul, R. Hull, V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.

[Ahmetaj *et al.*, 2016] S. Ahmetaj, W. Fischl, M. Kröll, R. Pichler, M. Simkus, S. Skritek. The challenge of optional matching in SPARQL. In *FoIKS*, vol. 9616 of *LNCS*, 2016.

[Antonioli *et al.*, 2014] N. Antonioli et al. Ontology-based data management for the Italian public debt. In *FOIS*, vol. 267 of *FAIA*, IOS Press, 2014.

[Artale *et al.*, 2009] A. Artale, D. Calvanese, R. Kontchakov, M. Zakharyaschev. The DL-Lite family and relations. *JAIR*, 36, 2009.

[Artale *et al.*, 2017] A. Artale, R. Kontchakov, A. Kovtunova, V. Ryzhikov, F. Wolter, M. Zakharyaschev. Ontology-mediated query answering over temporal data: A survey (invited talk). In *TIME*, vol. 90 of *LIPIcs*, 2017.

[Baader *et al.*, 2015] F. Baader, S. Borgwardt, M. Lippmann. Temporal query entailment in the description logic SHQ. *J. Web Semantics*, 33, 2015.

[Baader *et al.*, 2017] F. Baader, I. Horrocks, C. Lutz, U. Sattler. *An Introduction to Description Logic*. CUP, 2017.

[Bereta and Koubarakis, 2016] K. Bereta, M. Koubarakis. Ontop of geospatial databases. In *ISWC*, vol. 9981 of *LNCS*, 2016.

[Bienvenu and Ortiz, 2015] M. Bienvenu, M. Ortiz. Ontology-mediated query answering with data-tractable description logics. In *RW*, vol. 9203 of *LNCS*, 2015.

[Bienvenu and Rosati, 2016] M. Bienvenu, R. Rosati. Query-based comparison of mappings in ontology-based data access. In *KR*, 2016.

[Bienvenu *et al.*, 2014] M. Bienvenu, B. ten Cate, C. Lutz, F. Wolter. Ontology-based data access: A study through Disjunctive Datalog, CSP, and MMSNP. *ACM TODS*, 39(4), 2014.

[Bienvenu *et al.*, 2018] M. Bienvenu, S. Kikot, R. Kontchakov, V. Podolskii, M. Zakharyaschev. Ontology-mediated queries: Combined complexity and succinctness of rewritings via circuit complexity. *JACM*, 2018.

[Borgwardt *et al.*, 2015] S. Borgwardt, M. Lippmann, V. Thost. Temporalizing rewritable query languages over knowledge bases. *J. Web Semantics*, 33, 2015.

[Botoeva *et al.*, 2016] E. Botoeva, D. Calvanese, V. Santarelli, D. F. Savo, A. Solimando, G. Xiao. Beyond OWL 2 QL in OBDA: Rewritings and approximations. In *AAAI*, 2016.

[Brandt *et al.*, 2017] S. Brandt, E. G. Kalaycı, R. Kontchakov, V. Ryzhikov, G. Xiao, M. Zakharyaschev. Ontology-based data access with a Horn fragment of Metric Temporal Logic. In *AAAI*, 2017.

[Bursztyn *et al.*, 2015] D. Bursztyn, F. Goasdoué, I. Manolescu. Reformulation-based query answering in RDF: Alternatives and performance. *PVLDB*, 8(12), 2015.

[Calbimonte *et al.*, 2010] J.-P. Calbimonte, Ó. Corcho, A. J. Gray. Enabling ontology-based access to streaming data sources. In *ISWC*, vol. 6496 of *LNCS*, 2010.

[Calvanese *et al.*, 2007] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *JAR*, 39(3), 2007.

[Calvanese *et al.*, 2008] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, R. Rosati, M. Ruzzi. Data integration through *DL-Lite*$_\mathcal{A}$ ontologies. In *SDKB*, vol. 4925 of *LNCS*, 2008.

[Calvanese *et al.*, 2011] D. Calvanese et al. The Mastro system for ontology-based data access. *Semantic Web*, 2(1), 2011.

[Calvanese *et al.*, 2012] D. Calvanese, G. De Giacomo, M. Lenzerini, M. Y. Vardi. Query processing under GLAV mappings for relational and graph databases. *PVLDB*, 6, 2012.

[Calvanese *et al.*, 2013] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati. Data complexity of query answering in description logics. *AIJ*, 195, 2013.

[Calvanese *et al.*, 2015] D. Calvanese, M. Giese, D. Hovland, M. Rezk. Ontology-based integration of cross-linked datasets. In *ISWC*, vol. 9366 of *LNCS*, 2015.

[Calvanese *et al.*, 2017] D. Calvanese et al. Ontop: Answering SPARQL queries over relational databases. *Semantic Web*, 8(3), 2017.

[Console and Lenzerini, 2014] M. Console, M. Lenzerini. Data quality in ontology-based data access: The case of consistency. In *AAAI*, 2014.

[De Giacomo *et al.*, 2017] G. De Giacomo, D. Lembo, X. Oriol, D. F. Savo, E. Teniente. Practical update management in ontology-based data access. In *ISWC*, 2017.

[De Giacomo *et al.*, 2018] G. De Giacomo et al. Using ontologies for semantic data integration. In *A Comprehensive Guide through the Italian Database Research over the Last 25 Years*, vol. 31 of *Studies in Big Data*. Springer, 2018.

[Di Pinto *et al.*, 2013] F. Di Pinto et al. Optimizing query rewriting in ontology-based data access. In *EDBT*, 2013.

[Gerasimova *et al.*, 2017] O. Gerasimova, S. Kikot, V. Podolskii, M. Zakharyaschev. On the data complexity of ontology-mediated queries with a covering axiom. In *DL*, vol. 1879 of *CEUR*, 2017.

[Glimm and Krötzsch, 2010] B. Glimm, M. Krötzsch. SPARQL beyond subgraph matching. In *ISWC*, 2010.

[Gottlob *et al.*, 2014] G. Gottlob, G. Orsi, A. Pieris. Query rewriting and optimization for ontological databases. *ACM TODS*, 39(3), 2014.

[Hansen and Lutz, 2017] P. Hansen, C. Lutz. Computing FO-rewritings in EL in practice: From atomic to conjunctive queries. In *ISWC*, vol. 10587 of *LNCS*, 2017.

[Hernich *et al.*, 2017] A. Hernich, C. Lutz, F. Papacchini, F. Wolter. Dichotomies in ontology-mediated querying with the guarded fragment. In *PODS*, 2017.

[Hovland *et al.*, 2016] D. Hovland, D. Lanti, M. Rezk, G. Xiao. OBDA constraints for effective query answering. In *RuleML*, vol. 9718 of *LNCS*, 2016.

[Hovland *et al.*, 2017] D. Hovland, R. Kontchakov, M. G. Skjæveland, A. Waaler, M. Zakharyaschev. Ontology-based data access to Slegge. In *ISWC*, vol. 10588 of *LNCS*, 2017.

[Jiménez-Ruiz *et al.*, 2015] E. Jiménez-Ruiz et al. BootOX: Practical mapping of RDBs to OWL 2. In *ISWC*, vol. 9367 of *LNCS*, 2015.

[Kaminski *et al.*, 2016] M. Kaminski, Y. Nenov, B. Cuenca Grau. Datalog rewritability of Disjunctive Datalog programs and non-Horn ontologies. *AIJ*, 236, 2016.

[Kharlamov *et al.*, 2017] E. Kharlamov et al. Semantic access to streaming and static data at Siemens. *J. Web Semantics*, 44, 2017.

[Klarman and Meyer, 2014] S. Klarman, T. Meyer. Querying temporal databases via OWL 2 QL. In *RR*, vol. 8741 of *LNCS*, 2014.

[König *et al.*, 2015] M. König, M. Leclère, M.-L. Mugnier, M. Thomazo. Sound, complete and minimal UCQ-rewriting for existential rules. *Semantic Web*, 6(5), 2015.

[Kontchakov *et al.*, 2014] R. Kontchakov, M. Rezk, M. Rodriguez-Muro, G. Xiao, M. Zakharyaschev. Answering SPARQL queries over databases under OWL 2 QL entailment regime. In *ISWC*, vol. 8796 of *LNCS*, 2014.

[Kontchakov *et al.*, 2016] R. Kontchakov, L. Pandolfo, L. Pulina, V. Ryzhikov, M. Zakharyaschev. Temporal and spatial OBDA with many-dimensional Halpern-Shoham logic. In *IJCAI*, 2016.

[Kostylev and Cuenca Grau, 2015] E. V. Kostylev, B. Cuenca Grau. Semantics of SPARQL under OWL 2 entailment regimes. In *DL*, vol. 1350 of *CEUR*, 2015.

[Krötzsch and Rudolph, 2011] M. Krötzsch, S. Rudolph. Extending decidable existential rules by joining acyclicity and guardedness. In *IJCAI*, 2011.

[Lanti *et al.*, 2015] D. Lanti, M. Rezk, G. Xiao, D. Calvanese. The NPD benchmark: Reality check for OBDA systems. In *EDBT*, 2015.

[Lanti *et al.*, 2017] D. Lanti, G. Xiao, D. Calvanese. Cost-driven ontology-based data access. In *ISWC*, vol. 10587 of *LNCS*, 2017.

[Lembo *et al.*, 2015] D. Lembo, J. Mora, R. Rosati, D. F. Savo, E. Thorstensen. Mapping analysis in ontology-based data access: Algorithms and complexity. In *ISWC*, vol. 9366 of *LNCS*, 2015.

[Lembo *et al.*, 2017] D. Lembo, R. Rosati, V. Santarelli, D. F. Savo, E. Thorstensen. Mapping repair in ontology-based data access evolving systems. In *IJCAI*, 2017.

[Lenzerini, 2002] M. Lenzerini. Data integration: A theoretical perspective. In *PODS*, 2002.

[López *et al.*, 2015] V. López, M. Stephenson, S. Kotoulas, P. Tommasi. Data access linking and integration with DALI: Building a safety net for an ocean of city data. In *ISWC*, vol. 9367 of *LNCS*, 2015.

[Lutz and Sabellek, 2017] C. Lutz, L. Sabellek. Ontology-mediated querying with the description logic EL: Trichotomy and Linear Datalog rewritability. In *IJCAI*, 2017.

[Lutz and Wolter, 2017] C. Lutz, F. Wolter. The data complexity of description logic ontologies. *LMCS*, 13, 2017.

[Lutz *et al.*, 2013] C. Lutz, I. Seylan, D. Toman, F. Wolter. The combined approach to OBDA: Taming role hierarchies using filters. In *ISWC*, vol. 8218 of *LNCS*, 2013.

[Mora and Corcho, 2013] J. Mora, O. Corcho. Engineering optimisations in query rewriting for OBDA. In *I-SEMANTICS*. ACM, 2013.

[Nikolaou *et al.*, 2017] C. Nikolaou, E. Kostylev, G. Konstantinidis, M. Kaminski, B. Cuenca Grau, I. Horrocks. The bag semantics of ontology-based data access. In *IJCAI*, 2017.

[Petersen *et al.*, 2017] N. Petersen et al. Realizing an RDF-based information model for a manufacturing company — A case study. In *ISWC*, vol. 10588 of *LNCS*, 2017.

[Poggi *et al.*, 2008] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati. Linking data to ontologies. *J. Data Semantics*, 10, 2008.

[Priyatna *et al.*, 2014] F. Priyatna, O. Corcho, J. F. Sequeda. Formalisation and experiences of R2RML-based SPARQL to SQL query translation using morph. In *WWW*, 2014.

[Rahimi *et al.*, 2014] A. Rahimi, S.-T. Liaw, J. Taggart, P. Ray, H. Yu. Validating an ontology-based algorithm to identify patients with Type 2 Diabetes Mellitus in electronic health records. *Int. J. Med. Inf.*, 83(10), 2014.

[Schaerf, 1993] A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *J. Intelligent Inf. Syst.*, 2, 1993.

[Sequeda and Miranker, 2013] J. F. Sequeda, D. P. Miranker. Ultrawrap: SPARQL execution on relational data. *J. Web Semantics*, 22, 2013.

[Sequeda *et al.*, 2011] J. F. Sequeda, S. H. Tirmizi, O. Corcho, D. P. Miranker. Survey of directly mapping SQL databases to the Semantic Web. *Knowl. Eng. Rev.*, 26, 2011.

[Sequeda *et al.*, 2014] J. F. Sequeda, M. Arenas, D. P. Miranker. OBDA: Query rewriting or materialization? In practice, both! In *ISWC*, vol. 8796 of *LNCS*, 2014.

[Vardi, 1982] M. Vardi. The complexity of relational query languages (extended abstract). In *STOC*, 1982.

[Xiao *et al.*, 2014] G. Xiao, M. Rezk, M. Rodriguez-Muro, D. Calvanese. Rules and ontology based data access. In *RR*, vol. 8741 of *LNCS*, 2014.

[Xiao *et al.*, 2018a] G. Xiao et al. Efficient ontology-based data integration with canonical IRIs. In *ESWC*, 2018.

[Xiao *et al.*, 2018b] G. Xiao, R. Kontchakov, B. Cogrel, D. Calvanese, E. Botoeva. Efficient Handling of SPARQL optional for OBDA. In *ISWC*, 2018.