Query Rewriting and Optimisation with Database Dependencies in *Ontop*

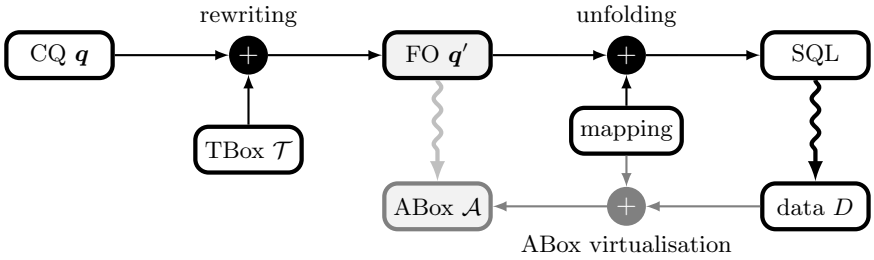
Mariano Rodríguez-Muro[1], Roman Kontchakov[2] and Michael Zakharyaschev[2]

[1] Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
[2] Department of Computer Science and Information Systems, Birkbeck, University of London, U.K.

Abstract. We present technologies that underpin the OBDA system *Ontop* and take full advantage of storing data in relational databases. We discuss the theoretical foundations of *Ontop*, including the tree-witness query rewriting, $\mathcal{T}$-mappings and optimisations based on database integrity constraints and SQL features.

1 Introduction

Ontology-based data access (OBDA) [18] is regarded as a key ingredient for the new generation of information systems. In the OBDA paradigm, an ontology defines a high-level global schema and provides a vocabulary for user queries, thus isolating the user from the details of the data source structure (which can be a relational database, a triple store, a datalog engine, etc.). The OBDA system transforms user queries into the vocabulary of the data and then delegates the actual query evaluation to the data sources. In this paper, we concentrate on OBDA for ontologies formulated in OWL 2 QL, a profile of OWL 2 specifically tailored to support rewriting of conjunctive queries (CQs) over ontologies into first-order (FO) queries. A standard architecture of such an OBDA system over relational data sources can be represented as follows:



The user is given an OWL 2 QL TBox $\mathcal{T}$ and can formulate CQs $\boldsymbol{q}(\boldsymbol{x})$ in the signature of $\mathcal{T}$. The system rewrites $\boldsymbol{q}$ and $\mathcal{T}$ into an FO-query $\boldsymbol{q}'(\boldsymbol{x})$, called a *rewriting of* $\boldsymbol{q}$ *and* $\mathcal{T}$, such that $(\mathcal{T},\mathcal{A}) \models \boldsymbol{q}(\boldsymbol{a})$ iff $\mathcal{A} \models \boldsymbol{q}'(\boldsymbol{a})$, for any ABox $\mathcal{A}$ and any tuple $\boldsymbol{a}$ of individuals in $\mathcal{A}$. A number of different rewriting techniques have been proposed and implemented for OWL 2 QL (PerfectRef [18], Presto/Prexto [24, 23], Rapid [4], the tree-witness rewriting [11]) and its extensions ([12], Nyaya [7], Requiem/Blackout [16, 17], Clipper [5]).

The rewriting $\boldsymbol{q}'$ is formulated in the signature of $\mathcal{T}$ and has to be further transformed into the vocabulary of the data source D before being evaluated. For instance, $\boldsymbol{q}'$ can be *unfolded* into an SQL query by means of a GAV mapping $\mathcal{M}$ relating the signature of $\mathcal{T}$ to the vocabulary of D. Strangely enough, mappings and unfoldings have largely been ignored by query rewriting algorithms (with Mastro-I [18] being an exception), partly because the data was assumed to be given as an ABox (say, as a universal table in a database or as a triple store). We consider the query transformation process as consisting of two steps—query rewriting and unfolding—and argue that this brings practical benefits (even in the case of seemingly trivial mappings for universal tables or triple stores).

The performance of first OBDA systems based on the architecture above was marred by large rewritings that could not be processed by RDBMSs, which led the OBDA community to intensive investigations of rewriting techniques and optimisations. There are 3 main reasons for large CQ rewritings and unfoldings:

(E) Sub-queries of $\boldsymbol{q}$ with existentially quantified variables can be folded in many different ways to match the canonical models of possible $(\mathcal{T}, \mathcal{A})$, all of which must be reflected in the rewriting $\boldsymbol{q}'$.

(H) The concepts and roles for atoms in $\boldsymbol{q}$ can have many sub-concepts and sub-roles according to $\mathcal{T}$, which also have to be included in the rewriting $\boldsymbol{q}'$.

(M) The mapping $\mathcal{M}$ can have multiple definitions of the ontology terms, which may result in an exponential blowup when $\boldsymbol{q}'$ is unfolded into a (most suitable for RDBMSs) union of Select-Project-Join queries.

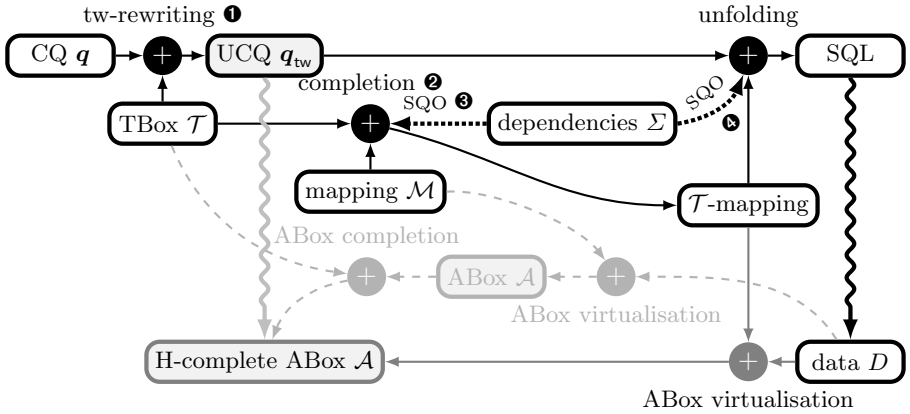
In fact, most of the proposed rewriting techniques try to tame (E): various optimisations are used in unification strategies to reduce the size of UCQs, with conjunctive query containment as the last resort. Presto [24] and the tree-witness rewriting [11] use non-recursive datalog to deal with (H); this, however, is of little help if a further transformation to a UCQ is required. The combined approach [13, 15] constructs finite representations of (in general) infinite canonical models of $(\mathcal{T}, \mathcal{A})$ thereby totally removing (H). It also solves (E) for TBoxes without role inclusions; otherwise, rewritings can still be of exponential size, or the filtering procedure [15] may have to run exponentially many times.

In theory, (E) turns out to be incurable under the architecture above: there exist CQs and OWL 2 QL TBoxes for which any FO- (or non-recursive datalog) rewriting results in a superpolynomial (or exponential) blowup [10], which happens independently of the contribution of (H) and (M); the polynomial rewriting of [8] hides this blowup behind the existential quantification over special constants. Fortunately, it seems that only (artificially) complex CQs and TBoxes trigger issues with (E). For real-world CQs and ontologies, the number of foldings in (E) appears to be very small and can be efficiently dealt with by suitable rewritings [11].

In this paper, we attack both (H) and (M) at *the same time* using two key observations. First, the schema and integrity constraints (dependencies), Σ, of the data source D together with the mapping $\mathcal{M}$ often provide valuable information about the class of possible ABoxes over which the user CQ is rewritten. (Note that these ABoxes are *virtual* representations of D and do not have to

be materialised.) For example, if we know that all our virtual ABoxes $\mathcal{A}$ are $\exists$-*complete* with respect to $\mathcal{T}$ (that is, contain witnesses for all concepts $\exists R$ in $\mathcal{T}$) then we do not face (E); if all $\mathcal{A}$ are *H-complete* (that is, $B(a) \in \mathcal{A}$ whenever $A(a) \in \mathcal{A}$ and $\mathcal{T} \models A \sqsubseteq B$, and similarly for roles) then (H) disappears. Second, we can make the virtual ABoxes H-complete by taking the composition of $\mathcal{T}$ and $\mathcal{M}$ as a new mapping. This composition, called a $\mathcal{T}$-*mapping* [20], can be simplified with the help of Σ and the features of the target query language before being used in the unfolding. As the simplifications use Σ, they preserve correct answers only over database instances satisfying Σ. (Even if the mappings are trivial and the ABox comes from a universal table or a triple store, it often has a certain structure and satisfies certain constraints, which could be taken into account to make query answering more efficient [9]).

These observations underpin the system *Ontop* (`ontop.inf.unibz.it`), which is implemented at the Free University of Bozen-Bolzano and available as a Protégé plugin, a SPARQL endpoint and OWLAPI and Sesame libraries. The process of query rewriting and unfolding in *Ontop* with all optimisations is shown in the picture below:



This architecture, which is our main contribution, will be discussed in detail in the remainder of the paper. Here we only emphasise the key ingredients:

- ❶ the tree-witness rewriting $\boldsymbol{q}_{\text{tw}}$ assumes the virtual ABoxes to be H-complete; it separates the topology of $\boldsymbol{q}$ from the taxonomy defined by $\mathcal{T}$, is fast in practice and produces short UCQs;
- ❷ the $\mathcal{T}$-mapping combines the system mapping $\mathcal{M}$ with the taxonomy of $\mathcal{T}$ to ensure H-completeness of virtual ABoxes;
- ❸ the $\mathcal{T}$-mapping is simplified using the Semantic Query Optimisation (SQO) technique and SQL features; the $\mathcal{T}$-mapping is constructed and optimised for the given $\mathcal{T}$ and Σ only once, and is used for unfolding all rewritings $\boldsymbol{q}_{\text{tw}}$;
- ❹ the unfolding algorithm uses SQO to produce small and efficient SQL queries.

Our experimental results [22, 21] (also `www.dcs.bbk.ac.uk/~roman/tw-rewriting`) show that when applied to real-world queries, ontologies and databases, *Ontop* automatically produces rewritings of reasonably high quality and its performance is comparable to that of traditional RDBMSs with hand-crafted queries.

2 OWL 2 QL and Databases

The language of OWL 2 QL contains *individual names* a_i, *concept names* A_i, and *role names* P_i ($i \geq 1$). *Roles* R and *basic concepts* B are defined by the grammar:

$$R \quad ::= \quad P_i \quad | \quad P_i^-, \qquad\qquad B \quad ::= \quad \bot \quad | \quad A_i \quad | \quad \exists R.$$

A *TBox*, $\mathcal{T}$, is a finite set of *inclusions* of the form

$$B_1 \sqsubseteq B_2, \qquad B_1 \sqsubseteq \exists R.B_2, \qquad B_1 \sqcap B_2 \sqsubseteq \bot, \qquad R_1 \sqsubseteq R_2, \qquad R_1 \sqcap R_2 \sqsubseteq \bot.$$

An *ABox*, $\mathcal{A}$, is a finite set of atoms of the form $A_k(a_i)$ or $P_k(a_i, a_j)$. The semantics for OWL 2 QL is defined in the usual way based on interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ [2]. The set of individual names in $\mathcal{A}$ is denoted by $\mathsf{ind}(\mathcal{A})$. Although ABoxes do not contain inverse roles, we write $P^-(a, b) \in \mathcal{A}$ if $P(b, a) \in \mathcal{A}$; also, we write $\exists R(a) \in \mathcal{A}$ if $R(a, b) \in \mathcal{A}$, for some b. We denote by $\sqsubseteq_{\mathcal{T}}$ the subsumption relation induced by $\mathcal{T}$ and write $S_1 \sqsubseteq_{\mathcal{T}} S_2$ if $\mathcal{T} \models S_1 \sqsubseteq S_2$, where S_1 and S_2 both are either basic concepts or roles.

A *conjunctive query* (CQ) $\boldsymbol{q}(\boldsymbol{x})$ is a first-order formula $\exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$, where φ is a conjunction of atoms of the form $A_k(t_1)$ or $P_k(t_1, t_2)$, and each t_i is a *term* (an individual or a variable in $\boldsymbol{x}$ or $\boldsymbol{y}$). We often use the datalog notation for CQs, writing $\boldsymbol{q}(\boldsymbol{x}) \leftarrow \varphi(\boldsymbol{x}, \boldsymbol{y})$ (without the existential quantifiers), and call $\boldsymbol{q}$ the *head* and φ the *body* of the rule. The variables in $\boldsymbol{x}$ are called *answer variables*. A tuple $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$ is a *certain answer* to $\boldsymbol{q}(\boldsymbol{x})$ over $(\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \boldsymbol{q}(\boldsymbol{a})$ for all models $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$; in this case we write $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}(\boldsymbol{a})$. We sometimes identify $\boldsymbol{q}$ with the set of its atoms, and set $R(x, y) = P(x, y)$ if $R = P$, and $R(x, y) = P(y, x)$ if $R = P^-$.

As explained in the introduction, we assume that the data comes from a relational database rather than an ABox. We view databases [1] as triples $(\boldsymbol{R}, \Sigma, \boldsymbol{I})$, where $\boldsymbol{R}$ is a database schema, containing predicate symbols (with their arity) for both stored database relations and views (together with their definitions in terms of stored relations), Σ is a set of integrity constraints over $\boldsymbol{R}$ (in the form of inclusion and functional dependencies), and $\boldsymbol{I}$ is a data instance over $\boldsymbol{R}$ (satisfying Σ). The vocabularies of $\boldsymbol{R}$ and $\mathcal{T}$ are linked together by means of mappings given by a domain expert or extracted (semi-)automatically. A *mapping*, $\mathcal{M}$, from $\boldsymbol{R}$ to $\mathcal{T}$ is a set of (GAV) rules of the form

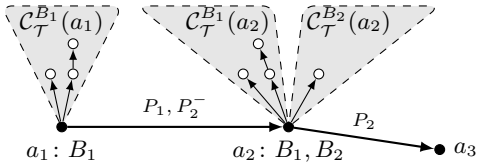
$$S(\boldsymbol{x}) \leftarrow \varphi(\boldsymbol{x}, \boldsymbol{z}),$$

where S is a concept name or a role in $\mathcal{T}$ and $\varphi(\boldsymbol{x}, \boldsymbol{z})$ a conjunction of atoms with stored relations and views from $\boldsymbol{R}$ and a *filter*, that is, a Boolean combination of built-in predicates such as $=$ and $<$. (Note that, by including views in the schema, we can express any SQL query in mappings.) Given a mapping $\mathcal{M}$ from $\boldsymbol{R}$ to $\mathcal{T}$, the ground atoms $S(\boldsymbol{a})$, for $S(\boldsymbol{x}) \leftarrow \varphi(\boldsymbol{x}, \boldsymbol{z})$ in $\mathcal{M}$ and $\boldsymbol{I} \models \exists \boldsymbol{z}\, \varphi(\boldsymbol{a}, \boldsymbol{z})$, comprise the ABox, $\mathcal{A}_{\boldsymbol{I},\mathcal{M}}$, which is called the *virtual ABox* for $\mathcal{T}$ and $\mathcal{M}$ over $\boldsymbol{I}$. We can now define *certain answers* to a CQ $\boldsymbol{q}$ over a TBox $\mathcal{T}$ and a database $(\boldsymbol{R}, \Sigma, \boldsymbol{I})$ linked by a mapping $\mathcal{M}$ as certain answers to $\boldsymbol{q}$ over $(\mathcal{T}, \mathcal{A}_{\boldsymbol{I},\mathcal{M}})$.

3 The Tree-Witness Rewriting over H-complete ABoxes

In the rewriting used in *Ontop*, we assume that the ABoxes $\mathcal{A}$ are *H-complete with respect to* $\mathcal{T}$ in the sense that $S_2(\boldsymbol{a}) \in \mathcal{A}$ whenever $S_1(\boldsymbol{a}) \in \mathcal{A}$ and $S_1 \sqsubseteq_{\mathcal{T}} S_2$. The issue of completing ABoxes will be discussed in Sec. 4.

Let $\boldsymbol{q}(\boldsymbol{x}) = \exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$. As is well-known, for any ABox $\mathcal{A}$, there is a *canonical model* $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ of $(\mathcal{T}, \mathcal{A})$ such that, for all $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}(\boldsymbol{a})$ iff $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models \boldsymbol{q}(\boldsymbol{a})$ iff there is a homomorphism $h\colon \boldsymbol{q}(\boldsymbol{a}) \to \mathcal{C}_{\mathcal{T},\mathcal{A}}$. The domain of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ consists of two parts: $\mathsf{ind}(\mathcal{A})$ and the witnesses introduced by the $\exists$ quantifiers in $\mathcal{T}$. We assume that every $a \in \mathsf{ind}(\mathcal{A})$ with $B(a) \in \mathcal{A}$ roots a (possibly infinite) subtree $\mathcal{C}_{\mathcal{T}}^{B}(a)$ of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$, which may intersect another such subtree only on their common root (each $\mathcal{C}_{\mathcal{T}}^{B}(a)$ is isomorphic over the signature of $\mathcal{T}$ to the canonical model of $(\mathcal{T} \cup \{A \sqsubseteq B\}, \{A(a)\})$, for a fresh concept name A).



Each homomorphism $h\colon \boldsymbol{q}(\boldsymbol{a}) \to \mathcal{C}_{\mathcal{T},\mathcal{A}}$ splits $\boldsymbol{q}$ into a sub-query mapped by h to $\mathsf{ind}(\mathcal{A})$ and a subquery mapped to the trees $\mathcal{C}_{\mathcal{T}}^{B}(a)$, for $B(a) \in \mathcal{A}$. We can think of a rewriting of $\boldsymbol{q}$ and $\mathcal{T}$ as listing possible splits of $\boldsymbol{q}$ into such subqueries. We first characterise the subqueries of $\boldsymbol{q}$ that can be mapped to subtrees $\mathcal{C}_{\mathcal{T}}^{B}(a)$. Let $\mathfrak{t} = (\mathfrak{t}_\mathsf{r}, \mathfrak{t}_\mathsf{i})$ be a pair of disjoint sets of terms in $\boldsymbol{q}$ such that $\mathfrak{t}_\mathsf{i} \neq \emptyset$ and $\mathfrak{t}_\mathsf{i} \subseteq \boldsymbol{y}$. Consider the subset $\boldsymbol{q}_\mathfrak{t}$ of $\boldsymbol{q}$ comprising atoms with terms in $\mathfrak{t}_\mathsf{r} \cup \mathfrak{t}_\mathsf{i}$ but not entirely in $\mathfrak{t}_\mathsf{r}$:

$$\boldsymbol{q}_\mathfrak{t} = \big\{ S(\boldsymbol{z}) \in \boldsymbol{q} \quad | \quad \boldsymbol{z} \subseteq \mathfrak{t}_\mathsf{r} \cup \mathfrak{t}_\mathsf{i} \text{ and } \boldsymbol{z} \not\subseteq \mathfrak{t}_\mathsf{r} \big\}.$$

We say that $\mathfrak{t}$ is *generated* by a basic concept B if there is a homomorphism $h\colon \boldsymbol{q}_\mathfrak{t} \to \mathcal{C}_{\mathcal{T}}^{B}(a)$, for some a, such that $h^{-1}(a) = \mathfrak{t}_\mathsf{r}$. We call $\mathfrak{t}$ a *tree witness for* $\boldsymbol{q}$ *and* $\mathcal{T}$ if $\mathfrak{t}$ is generated by some B, $\boldsymbol{q}_\mathfrak{t}$ is connected and contains all atoms of $\boldsymbol{q}$ with at least one variable from $\mathfrak{t}_\mathsf{i}$. (The last condition reflects the fact that if a homomorphism from $\boldsymbol{q}(\boldsymbol{a})$ sends a variable y of an atom $P(y,t) \in \boldsymbol{q}$ to a non-root point of a subtree $\mathcal{C}_{\mathcal{T}}^{B}(a)$ of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ then the other term t must be sent to the same subtree $\mathcal{C}_{\mathcal{T}}^{B}(a)$.) The terms in $\mathfrak{t}_\mathsf{r}$ (if any) are called *roots* and the variables in $\mathfrak{t}_\mathsf{i}$ the *interior* of $\mathfrak{t}$. Assuming that $\mathfrak{t}_\mathsf{r} = \{t_1, \dots, t_k\}$, $k \geq 0$, we associate with a tree witness $\mathfrak{t}$ a k-ary predicate $\mathsf{tw}_\mathfrak{t}$ defined by the following set of rules:

$$\mathsf{tw}_\mathfrak{t}(x, \dots, x) \leftarrow B(x), \qquad \text{if } \mathfrak{t} \text{ is generated by } B, \tag{1}$$

where $B(x) = A(x)$ if $B = A$, $B(x) = P(x, \_)$ if $B = \exists P$, $B(x) = P(\_, x)$ if $B = \exists P^-$, and $\_$ denotes an anonymous existentially quantified variable (since the ABox is H-complete, we take only those *basic concepts* B generating $\mathfrak{t}$ that are maximal with respect to $\sqsubseteq_{\mathcal{T}}$). If $\mathfrak{t}_\mathsf{r} \neq \emptyset$ then (1) makes all the arguments of $\mathsf{tw}_\mathfrak{t}$ equal, thus complying with $h^{-1}(a) = \mathfrak{t}_\mathsf{r}$. Otherwise, $\mathfrak{t}_\mathsf{r} = \emptyset$ and so, $\mathsf{tw}_\mathfrak{t}$ is a propositional variable and x is existentially quantified in the body of (1). As the arguments of $\mathsf{tw}_\mathfrak{t}$ play identical roles, we can write $\mathsf{tw}_\mathfrak{t}(\mathfrak{t}_\mathsf{r})$ without specifying any order on the set $\mathfrak{t}_\mathsf{r}$.

Tree witnesses $\mathfrak{t}$ and $\mathfrak{t}'$ are *consistent* if they intersect only on their common roots: $(\mathfrak{t}_\mathsf{r} \cup \mathfrak{t}_\mathsf{i}) \cap (\mathfrak{t}'_\mathsf{r} \cup \mathfrak{t}'_\mathsf{i}) \subseteq \mathfrak{t}_\mathsf{r} \cap \mathfrak{t}'_\mathsf{r}$. Each set Θ of pairwise consistent tree witnesses determines a subquery $\boldsymbol{q}_\Theta$ of $\boldsymbol{q}$ that comprises all atoms of $\boldsymbol{q}_\mathfrak{t}$, for $\mathfrak{t} \in \Theta$. The subquery $\boldsymbol{q}_\Theta$ is to be mapped to the $\mathcal{C}^B_\mathcal{T}(a)$, whereas the remainder $\boldsymbol{q} \setminus \boldsymbol{q}_\Theta$, obtained by removing the atoms of $\boldsymbol{q}_\Theta$ from $\boldsymbol{q}$, is mapped to $\mathsf{ind}(\mathcal{A})$. Thus, the *tree-witness rewriting* $\boldsymbol{q}_\mathsf{tw}$ is defined by the rules for the $\mathsf{tw}_\mathfrak{t}$ together with the following:

$$\boldsymbol{q}_\mathsf{tw}(\boldsymbol{x}) \leftarrow (\boldsymbol{q} \setminus \boldsymbol{q}_\Theta) \;\wedge \bigwedge_{\mathfrak{t} \in \Theta} \mathsf{tw}_\mathfrak{t}(\mathfrak{t}_\mathsf{r}), \qquad \text{for consistent } \Theta. \tag{2}$$

Theorem 1. *For any H-complete (with respect to $\mathcal{T}$) ABox $\mathcal{A}$ and $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$, we have $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models \boldsymbol{q}(\boldsymbol{a})$ iff $\mathcal{A} \models \boldsymbol{q}_\mathsf{tw}(\boldsymbol{a})$.*

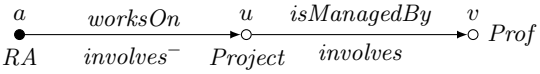
As defined, the rewriting $\boldsymbol{q}_\mathsf{tw}$ is a non-recursive datalog query. In what follows, however, we regard $\boldsymbol{q}_\mathsf{tw}$ as a UCQ obtained by replacing occurrences of the $\mathsf{tw}_\mathfrak{t}$ by their definitions. As an example, consider an ontology $\mathcal{T}$ with the axioms

$$\begin{array}{rr} \mathit{RA} \sqsubseteq \exists \mathit{worksOn}.\mathit{Project}, & \mathit{worksOn}^- \sqsubseteq \mathit{involves}, \\ \mathit{Project} \sqsubseteq \exists \mathit{isManagedBy}.\mathit{Prof}, & \mathit{isManagedBy} \sqsubseteq \mathit{involves} \end{array}$$

and the CQ asking to find those who work with professors:

$$\boldsymbol{q}(x) \leftarrow \mathit{worksOn}(x, y) \wedge \mathit{involves}(y, z) \wedge \mathit{Prof}(z).$$

$\mathcal{C}^{RA}_\mathcal{T}(a)$ looks as follows:



We have two tree witnesses generated by RA:

$$\begin{array}{ll} \mathfrak{t}^1 = (\{x\}, \{y, z\}), & \text{for } h(x) = a,\ h(y) = u,\ h(z) = v, \\ \mathfrak{t}^2 = (\{x, z\}, \{y\}), & \text{for } h(x) = h(z) = a,\ h(y) = u, \end{array}$$

and one tree witness $\mathfrak{t}^3 = (\{y\}, \{z\})$ generated by $\mathit{Project}$. Thus, there are 4 sets of consistent tree witnesses: $\emptyset$, $\{\mathfrak{t}^1\}$, $\{\mathfrak{t}^2\}$, $\{\mathfrak{t}^3\}$, which give the following rewriting:

$$\boldsymbol{q}_\mathsf{tw}(x) \leftarrow \mathit{worksOn}(x, y), \mathit{involves}(y, z), \mathit{Prof}(z), \tag{3}$$
$$\boldsymbol{q}_\mathsf{tw}(x) \leftarrow \mathit{RA}(x), \tag{4}$$
$$\boldsymbol{q}_\mathsf{tw}(x) \leftarrow \mathit{RA}(x), \mathit{Prof}(x), \tag{5}$$
$$\boldsymbol{q}_\mathsf{tw}(x) \leftarrow \mathit{worksOn}(x, y), \mathit{Project}(y). \tag{6}$$

The size of the tree-witness rewriting depends on the number of consistent sets of tree witnesses for $\boldsymbol{q}$ and $\mathcal{T}$. There exist CQs and TBoxes whose shortest rewritings are exponential [10]. Observe, however, that to generate many tree witnesses, the CQ $\boldsymbol{q}$ must have many subqueries that can be homomorphically mapped to some $\mathcal{C}^B_\mathcal{T}(a)$, which requires both $\boldsymbol{q}$ and $\mathcal{C}^B_\mathcal{T}(a)$ to be quite sophisticated, with $\boldsymbol{q}$ 'mimicking' parts of $\mathcal{C}^B_\mathcal{T}(a)$ as in the example above. To the best of our knowledge, this does not occurs in real-world CQs and ontologies used for

OBDA. More often than not, they do not generate tree witnesses at all. It is also known [11, Theorem 21] that, if the CQ and ontology do not contain fragments as in the (artificial) example above, then the number of consistent sets of tree witnesses is polynomial. As a result, the UCQ rewriting $\boldsymbol{q}_{\mathsf{tw}}$ is typically small in practice, and so further optimisations can be performed quickly. There are two ways of optimising this UCQ. First, we can use a subsumption algorithm to remove redundant CQs from the union: for example, (4) subsumes (5), which can therefore be safely removed. Second, we can reduce the size of individual CQs in the union using the following observation: for any CQ $\boldsymbol{q}$ (viewed as a set of atoms),

$$\begin{aligned}
&\boldsymbol{q} \equiv_c \boldsymbol{q} \setminus \{A(x)\}, && \text{if } A'(x) \in \boldsymbol{q} \text{ and } A' \sqsubseteq_{\mathcal{T}} A \text{ and } A' \neq A,\\
&\boldsymbol{q} \equiv_c \boldsymbol{q} \setminus \{A(x)\}, && \text{if } R(x,y) \in \boldsymbol{q} \text{ and } \exists R \sqsubseteq_{\mathcal{T}} A,\\
&\boldsymbol{q} \equiv_c \boldsymbol{q} \setminus \{P(x,y)\}, && \text{if } R(x,y) \in \boldsymbol{q} \text{ and } R \sqsubseteq_{\mathcal{T}} P \text{ and } R \neq P,
\end{aligned}$$

where $\equiv_c$ reads 'has the same certain answers over *H-complete* ABoxes.' Surprisingly, such a simple optimisation, especially for $\exists R \sqsubseteq_{\mathcal{T}} A$, makes rewritings substantially shorter [24, 7].

4 From Rewritings over ABoxes to Database Queries

The rewriting $\boldsymbol{q}_{\mathsf{tw}}$ works only for H-complete ABoxes. Given a mapping $\mathcal{M}$ from the database schema $\boldsymbol{R}$ to $\mathcal{T}$, one can define H-complete (with respect to $\mathcal{T}$) ABoxes by taking the composition $\mathcal{M}^{\mathcal{T}}$ of $\mathcal{M}$ and the inclusions in $\mathcal{T}$:

$$\begin{aligned}
A(x) &\leftarrow \varphi(x, \boldsymbol{z}) && \text{if } A'(x) \leftarrow \varphi(x, \boldsymbol{z}) \in \mathcal{M} \text{ and } A' \sqsubseteq_{\mathcal{T}} A,\\
A(x_0) &\leftarrow \varphi(x_0, x_1, \boldsymbol{z}) && \text{if } R(x_0, x_1) \leftarrow \varphi(x_0, x_1, \boldsymbol{z}) \in \mathcal{M} \text{ and } \exists R \sqsubseteq_{\mathcal{T}} A,\\
P(x_0, x_1) &\leftarrow \varphi(x_0, x_1, \boldsymbol{z}), && \text{if } R(x_0, x_1) \leftarrow \varphi(x_0, x_1, \boldsymbol{z}) \in \mathcal{M} \text{ and } R \sqsubseteq_{\mathcal{T}} P.
\end{aligned}$$

(Recall that we identify $P^-(x_1, x_0)$ with $P(x_0, x_1)$, in particular, in the heads of the mapping rules.) Clearly, for any database instance $\boldsymbol{I}$, the virtual ABox $\mathcal{A}_{\boldsymbol{I},\mathcal{M}^{\mathcal{T}}}$ is H-complete with respect to $\mathcal{T}$. Thus, to compute answers to $\boldsymbol{q}$ over $\mathcal{T}$ and a virtual ABox $\mathcal{A}_{\boldsymbol{I},\mathcal{M}}$, it suffices to evaluate the tree-witness rewriting $\boldsymbol{q}_{\mathsf{tw}}$ over $\mathcal{A}_{\boldsymbol{I},\mathcal{M}^{\mathcal{T}}}$:

Theorem 2. *For any data instance $\boldsymbol{I}$ and any tuple $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A}_{\boldsymbol{I},\mathcal{M}})$, we have $(\mathcal{T}, \mathcal{A}_{\boldsymbol{I},\mathcal{M}}) \models \boldsymbol{q}(\boldsymbol{a})$ iff $\mathcal{A}_{\boldsymbol{I},\mathcal{M}^{\mathcal{T}}} \models \boldsymbol{q}_{\mathsf{tw}}(\boldsymbol{a})$.*

Most OBDA systems first construct rewritings over arbitrary ABoxes and only then unfold them, using mappings, into unions of Select-Project-Join queries, which are evaluated by an RDBMS. By Theorem 2, the same result can be obtained by unfolding rewritings over H-complete ABoxes with the help of the composition $\mathcal{M}^{\mathcal{T}}$. However, in practice the resulting SQL query often turns out to be too large [19].

In *Ontop*, we also start with $\mathcal{M}^{\mathcal{T}}$. But before applying it to unfold $\boldsymbol{q}_{\mathsf{tw}}$, we simplify and reduce the size of $\mathcal{M}^{\mathcal{T}}$ using the database integrity constraints Σ.

A mapping $\mathcal{M}$ from $\boldsymbol{R}$ to $\mathcal{T}$ is called a *$\mathcal{T}$-mapping over Σ* if the ABox $\mathcal{A}_{\boldsymbol{I},\mathcal{M}}$ is H-complete with respect to $\mathcal{T}$, for any data instance $\boldsymbol{I}$ satisfying Σ. (The composition $\mathcal{M}^{\mathcal{T}}$ is a $\mathcal{T}$-mapping over any Σ.) *Ontop* transforms $\mathcal{M}^{\mathcal{T}}$ to a simpler $\mathcal{T}$-mapping using Σ and SQL features such as disjunctions in filter conditions.

To illustrate, we take a simplified IMDb[1], whose schema contains relations $\mathit{title}[m, t, y]$ with information about movies (ID, title, production year), and $\mathit{castinfo}[p, m, r]$ with information about movie casts (person ID, movie ID, person role), and an ontology MO,[2] describing the application domain in terms of, for example, concepts *mo:Movie* and *mo:Person*, and roles *mo:cast* and *mo:year*:

$$\mathit{mo{:}Movie} \equiv \exists \mathit{mo{:}title}, \qquad \mathit{mo{:}Movie} \sqsubseteq \exists \mathit{mo{:}year},$$
$$\mathit{mo{:}Movie} \equiv \exists \mathit{mo{:}cast}, \qquad \exists \mathit{mo{:}cast}^{-} \sqsubseteq \mathit{mo{:}Person}.$$

A mapping that relates the ontology terms to the database schema contains, for example, the following rules:

$$\mathit{mo{:}Movie}(m) \leftarrow \mathit{title}(m, t, y), \qquad \mathit{mo{:}cast}(m, p) \leftarrow \mathit{castinfo}(p, m, r) \tag{7}$$
$$\mathit{mo{:}title}(m, t) \leftarrow \mathit{title}(m, t, y), \qquad \mathit{mo{:}Person}(p) \leftarrow \mathit{castinfo}(p, m, r) \tag{8}$$
$$\mathit{mo{:}year}(m, y) \leftarrow \mathit{title}(m, t, y). \tag{9}$$

4.1 Integrity Constraints for $\mathcal{T}$-mapping Optimisation

Suppose a $\mathcal{T}$-mapping $\mathcal{M}$ over Σ contains two rules $S(\boldsymbol{x}) \leftarrow \psi_i(\boldsymbol{x}, \boldsymbol{z})$, $i = 1, 2$. If one of these rules is more specific than the other, then it can be removed without any change in the virtual ABoxes produced from database instances. To discover such 'more specific' rules, we run the standard query containment check (see, e.g., [1]) taking account of the *inclusion dependencies* $\Sigma_{IND} \subseteq \Sigma$, that is, integrity constraints of the form

$$\forall \boldsymbol{x} \left(\exists \boldsymbol{y}_1 \, R_1(\boldsymbol{z}_1) \rightarrow \exists \boldsymbol{y}_2 \, R_2(\boldsymbol{z}_2) \right),$$

where each R_i is a database relation and each $\boldsymbol{z}_i$ contains all the variables from $\boldsymbol{x}$ and $\boldsymbol{y}_i$, possibly in a different order.

(opt1) *If* $\Sigma_{IND} \models \forall \boldsymbol{x} \left(\exists \boldsymbol{z} \, \psi_1(\boldsymbol{x}, \boldsymbol{z}) \rightarrow \exists \boldsymbol{z} \, \psi_2(\boldsymbol{x}, \boldsymbol{z}) \right)$,
then $\mathcal{M} \setminus \{ S(\boldsymbol{x}) \leftarrow \psi_1(\boldsymbol{x}, \boldsymbol{z}) \}$ *is a $\mathcal{T}$-mapping over Σ.*

For example, since $\exists \mathit{mo{:}cast} \sqsubseteq_{\mathcal{T}} \mathit{mo{:}Movie}$, the composition $\mathcal{M}^{\mathrm{MO}}$ of the mapping $\mathcal{M}$ defined by (7)–(9) and MO contains the following rules for *mo:Movie*:

$$\mathit{mo{:}Movie}(m) \leftarrow \mathit{title}(m, t, y),$$
$$\mathit{mo{:}Movie}(m) \leftarrow \mathit{castinfo}(p, m, r).$$

By **(opt1)**, the latter rule is redundant because IMDb contains the foreign key (inclusion dependency)

$$\forall m \left(\exists p, r \, \mathit{castinfo}(p, m, r) \rightarrow \exists t, y \, \mathit{title}(m, t, y) \right).$$

[1] `http://www.imdb.com/interfaces`
[2] `http://www.movieontology.org`

The $\mathcal{T}$-mapping optimisation **(opt1)** turns out to be very effective in practice. Its power comes from the fact that Σ describes integrity constraints of database instances, while $\mathcal{T}$ describes concepts and roles defined by the mapping over the same data. Essentially, both are theories about the same entities but in different languages. Thus, it is no wonder that many inclusions in $\mathcal{T}$ are consequences of Σ, which allows us to drastically reduce the size of $\mathcal{T}$-mappings.

4.2 Disjunctions in SQL

Another efficient way to reduce the size of a $\mathcal{T}$-mapping is to identify rules whose bodies are equivalent up to *filters with respect to constant values*. More precisely, let a $\mathcal{T}$-mapping $\mathcal{M}$ contain two rules $S(\boldsymbol{x}) \leftarrow \psi(\boldsymbol{x}, \boldsymbol{z}), \varphi_i(\boldsymbol{x}, \boldsymbol{z})$, for $i = 1, 2$, such that φ_1 and φ_2 are Boolean conditions constructed from built-in predicates (such as $=$ and $<$).

(opt2) *The result of replacing* $S(\boldsymbol{x}) \leftarrow \psi(\boldsymbol{x}, \boldsymbol{z}), \varphi_i(\boldsymbol{x}, \boldsymbol{z})$, *for* $i = 1, 2$, *in* $\mathcal{M}$ *by* $S(\boldsymbol{x}) \leftarrow \psi(\boldsymbol{x}, \boldsymbol{z}), (\varphi_1(\boldsymbol{x}, \boldsymbol{z}) \vee \varphi_2(\boldsymbol{x}, \boldsymbol{z}))$ *is a* $\mathcal{T}$*-mapping over* Σ.

This optimisation deals with the rules introduced due to the so-called type (discriminating) attributes [6] in database schemas. For example, the mapping $\mathcal{M}$ for IMDb and MO contains six rules for sub-concepts of *mo:Person*:

$$\begin{gathered} \textit{mo:Actor}(p) \leftarrow \textit{castinfo}(c, p, m, r), (r = 1), \\ \dots \\ \textit{mo:Editor}(p) \leftarrow \textit{castinfo}(c, p, m, r), (r = 6). \end{gathered}$$

The composition $\mathcal{M}^{\text{MO}}$ contains six rules for *mo:Person* that differ only in the last condition $(r = k)$, for $k = 1, \dots, 6$, and **(opt2)** reduces them to a single one:

$$\textit{mo:Person}(p) \leftarrow \textit{castinfo}(c, p, m, r), \big((r = 1) \vee \dots \vee (r = 6)\big).$$

Such disjunctions lend themselves to efficient query evaluation by RDBMSs.

4.3 Semantic Index: $\mathcal{T}$-mappings over Materialised ABoxes

In addition to working with proper relational data sources, *Ontop* also supports ABox storage in the form of structureless *universal tables*: a binary relation $CA[\textit{id}, \textit{concept-id}]$ and a ternary relation $RA[\textit{id}_1, \textit{id}_2, \textit{role-id}]$ represent concept and role assertions. The universal tables give rise to trivial mappings and *Ontop* implements a technique, the *semantic index* [20], that takes advantage of SQL features in $\mathcal{T}$-mappings for this scenario. The key observation is that since the IDs in the universal tables CA and RA can be chosen by the system, each concept and role in the TBox $\mathcal{T}$ can be assigned a numeric *index* and a set of numeric *intervals* in such a way that the resulting $\mathcal{T}$-mapping contains simple SQL queries with interval filter conditions; cf. **(opt2)**. For example, in IMDb, we have

$$\textit{mo:Artist} \sqsubseteq \textit{mo:Person}, \quad \textit{mo:Director} \sqsubseteq \textit{mo:Person}, \quad \textit{mo:Actor} \sqsubseteq \textit{mo:Artist}$$

so we can choose index 1 and interval [1,1] for *mo:Actor*, 2 and [1,2] for *mo:Artist*, 3 and [3,3] for *mo:Director* and 6 and [1,6] for *mo:Person*. This will generate a $\mathcal{T}$-mapping with, for instance,

$$\begin{aligned} \textit{mo:Person}(p) &\leftarrow \textit{CA}(p, \textit{concept-id}), (1 \leq \textit{concept-id} \leq 6), \\ \textit{mo:Artist}(p) &\leftarrow \textit{CA}(p, \textit{concept-id}), (1 \leq \textit{concept-id} \leq 2). \end{aligned}$$

So, by choosing appropriate concept and role IDs, we, on the one hand, effectively construct H-complete ABoxes *without* the expensive forward chaining procedure (and the need to store large amounts of derived assertions). On the other hand, the semantic index $\mathcal{T}$-mappings are based on range expressions, which can be evaluated efficiently by RDBMSs using standard B-Tree indexes [6].

4.4 Unfolding with Semantic Query Optimisation (SQO)

To execute the rewriting $\boldsymbol{q}_{\mathsf{tw}}$ over an instance data, one can simply extend $\boldsymbol{q}_{\mathsf{tw}}$ with the definitions of ontology predicates provided by a $\mathcal{T}$-mapping. This would result in an SQL query with subqueries (for the mapping rules). In theory, it can be passed directly to an RDMBS. It is, however, known that RDBMSs are poor at estimating the cost and planning queries with complex subqueries. Instead, *Ontop* unfolds rewritings and $\mathcal{T}$-mappings into unions of Select-Project-Join queries, which are known to be optimal for execution by RDBMSs. The unfolding procedure [18] applies SLD-resolution to $\boldsymbol{q}_{\mathsf{tw}}$ and the $\mathcal{T}$-mapping, and returns those rules whose bodies contain only database atoms (cf. partial evaluation [14]).

Ontop applies SQO [3] to rules obtained at the intermediate steps of unfolding. In particular, this eliminates redundant self-Join operations caused by reification of database relations by means of concepts and roles. Recall [1] that, *functional dependencies* are integrity constraints of the form

$$\forall \boldsymbol{x} \, \forall u_1 \forall u_2 \, \big(\exists \boldsymbol{y} \, R(\boldsymbol{z}) \land \exists \boldsymbol{y} \, R(\boldsymbol{z}[u_1/u_2]) \to (u_1 = u_2) \big), \tag{10}$$

where R is a database relation, $\boldsymbol{z}$ contains u_1 and all the variables in $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}[u_1/u_2]$ is the result of replacing u_1 in $\boldsymbol{z}$ with u_2; $\boldsymbol{x}$ is called the *determinant* of the dependency.

(opt3) *If the determinants of a functional dependency* (10) *coincide in atoms* $R(\boldsymbol{z}_1)$ *and* $R(\boldsymbol{z}_2)$ *then any rule of the form* $\boldsymbol{q}(\boldsymbol{x}) \leftarrow \varphi(\boldsymbol{x}, \boldsymbol{y}), R(\boldsymbol{z}_1), R(\boldsymbol{z}_2)$ *can be equivalently replaced by the result of removing duplicate atoms from the body of* $\boldsymbol{q}(\boldsymbol{x}) \leftarrow \varphi(\boldsymbol{x}, \boldsymbol{y}), R(\boldsymbol{z}_1), R(\boldsymbol{z}_2[u_1/u_2])$.

Consider, for example, the CQ

$$\boldsymbol{q}(t, y) \leftarrow \textit{mo:Movie}(m), \textit{mo:title}(m, t), \textit{mo:year}(m, y), (y > 2010)$$

It has no tree witnesses, and so $\boldsymbol{q}_{\mathsf{tw}} = \boldsymbol{q}$. By straightforwardly applying the unfolding to $\boldsymbol{q}_{\mathsf{tw}}$ and the $\mathcal{T}$-mapping $\mathcal{M}$ defined by (7)–(9), we obtain the query

$$\boldsymbol{q}'_{\mathsf{tw}}(t, y) \leftarrow \textit{title}(m, t_0, y_0), \textit{title}(m, t, y_1), \textit{title}(m, t_2, y), (y > 2010),$$

which requires two (potentially) expensive JOIN operations. However, by using the primary key m of *title*:

$$\forall m\, \forall t_1 \forall t_2 \left(\exists y\, \mathit{title}(m, t_1, y) \wedge \exists y\, \mathit{title}(m, t_2, y) \rightarrow (t_1 = t_2)\right),$$
$$\forall m\, \forall y_1 \forall y_2 \left(\exists t\, \mathit{title}(m, t, y_1) \wedge \exists t\, \mathit{title}(m, t, y_2) \rightarrow (y_1 = y_2)\right)$$

(a functional dependency with determinant m), we reduce two JOIN operations in the first three atoms of $\boldsymbol{q}'_{\mathsf{tw}}$ to a single atom $\mathit{title}(m, t, y)$:

$$\boldsymbol{q}''_{\mathsf{tw}}(t, y) \leftarrow \mathit{title}(m, t, y), (y > 2010).$$

Note that these two JOIN operations were introduced to reconstruct the ternary relation from its reification by means of the roles *mo:title* and *mo:year*.

The role of SQO in OBDA systems appears to be much more prominent than in conventional RDBMSs, where it was initially proposed to optimise SQL queries. While some of SQO techniques reached industrial RDBMSs, it never had a strong impact on the database community because it is costly compared to statistics- and heuristics-based methods, and because most SQL queries are written by highly-skilled experts (and so are nearly optimal anyway). In OBDA scenarios, in contrast, SQL queries are generated automatically, and so SQO becomes the only tool to avoid redundant and expensive JOIN operations [25].

4.5 Why Does It Work?

The techniques above prove to be extremely efficient in practice. Moreover, they often automatically produce queries that are similar to those written by human experts. To understand why, we briefly review the process of designing database applications [6]. It starts with conceptual modelling which describes the application domain in such formalisms as ER, UML or ORM. The conceptual model gives the vocabulary of the database and defines its semantics by means of hierarchies, cardinality restrictions, etc. The conceptual model is turned into a relational database by applying a series of *standard* procedures that encode the semantics of the model into a flat relational schema. These procedures include:

- amalgamating many-to-one and one-to-one attributes of an entity to a single n-ary relation with a primary key identifying the entity (e.g., *title* with *mo:title* and *mo:year*); cf. Section 4.4;
- using foreign keys over attribute columns when a column refers to the entity (e.g., *title* and *castinfo*); cf. Section 4.1;
- using type (discriminating) attributes to encode hierarchical information (e.g., *castinfo*); cf. Sections 4.2 and 4.3.

As this process is universal, the $\mathcal{T}$-mappings created for the resulting databases are dramatically simplified by the *Ontop* optimisations **(opt1)**–**(opt3)**, and the resulting UCQs are usually of acceptable size and can be executed efficiently by RDBMSs.

Acknowledgements. We thank the *Ontop* team—Timea Bagosi, Josef Hardi and Mindaugas Slusnys—at the Free University of Bozen-Bolzano for their help in developing the system and running experiments.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
3. Chakravarthy, U.S., Fishman, D.H., Minker, J.: Semantic query optimization in expert systems and database systems. Benjamin-Cummings Publishing Co., Inc. (1986)
4. Chortaras, A., Trivela, D., Stamou, G.: Optimized query rewriting for OWL 2 QL. In: Proc. of CADE-23. LNCS, vol. 6803, pp. 192–206. Springer (2011)
5. Eiter, T., Ortiz, M., Šimkus, M., Tran, T.K., Xiao, G.: Query rewriting for Horn-SHIQ plus rules. In: Proc. of AAAI 2012. AAAI Press (2012)
6. Elmasri, R., Navathe, S.: Fundamentals of Database Systems. Addison-Wesley, 6th edn. (2010)
7. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: Proc. of ICDE 2011. pp. 2–13. IEEE Computer Society (2011)
8. Gottlob, G., Schwentick, T.: Rewriting ontological queries into small nonrecursive datalog programs. In: Proc. of KR 2012. AAAI Press (2012)
9. Kementsietsidis, A., Bornea, M., Dolby, J., Srinivas, K., Dantressangle, P., Udrea, O., Bhattacharjee, B.: Building an efficient RDF store over a relational database. In: Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2013). ACM (2013)
10. Kikot, S., Kontchakov, R., Podolskii, V., Zakharyaschev, M.: Exponential lower bounds and separation for query rewriting. In: Proc. of ICALP 2012, Part II. LNCS, vol. 7392, pp. 263–274. Springer (2012)
11. Kikot, S., Kontchakov, R., Zakharyaschev, M.: Conjunctive query answering with OWL 2 QL. In: Proc. of KR 2012. AAAI Press (2012)
12. König, M., Leclère, M., Mugnier, M.L., Thomazo, M.: A sound and complete backward chaining algorithm for existential rules. In: Proc. of RR 2012. LNCS, vol. 7497, pp. 122–138. Springer (2012)
13. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: Proc. of KR (2010)
14. Lloyd, J., Shepherdson, J.: Partial Evaluation in Logic Programming. The Journal of Logic Programming 11(3-4), 217–242 (Oct 1991)
15. Lutz, C., Seylan, İ., Toman, D., Wolter, F.: The combined approach to OBDA: Taming role hierarchies using filters. In: Proc. of SSWS+HPCSW 2012. CEUR-WS, vol. 943 (2012)
16. Pérez-Urbina, H., Motik, B., Horrocks, I.: A comparison of query rewriting techniques for DL-lite. In: Proc. of DL 2009. CEUR-WS, vol. 477 (2009)
17. Pérez-Urbina, H., Rodríguez-Díaz, E., Grove, M., Konstantinidis, G., Sirin, E.: Evaluation of query rewriting approaches for OWL 2. In: Proc. of SSWS+HPCSW 2012. CEUR-WS, vol. 943 (2012)
18. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. Data Semantics 10, 133–173 (2008)
19. Rodríguez-Muro, M.: Tools and Techniques for Ontology Based Data Access in Lightweight Description Logics. Ph.D. thesis, KRDB Research Centre for Knowledge and Data, Free Univ. of Bozen-Bolzano (2010)

20. Rodríguez-Muro, M., Calvanese, D.: Dependencies: Making ontology based data access work. In: Proc. of AMW 2011. vol. 749. CEUR-WS.org (2011)
21. Rodríguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: OBDA with Ontop. In: Proc. of the OWL Reasoner Evaluation Workshop 2013 (ORE 2013). CEUR-WS (2013)
22. Rodríguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: Ontop at work. In: Proc. of OWL: Experiences and Directions Workshop 2013 (OWLED 2013). CEUR-WS (2013)
23. Rosati, R.: Prexto: Query rewriting under extensional constraints in DL-Lite. In: Proc. of EWSC 2012. LNCS, vol. 7295, pp. 360–374. Springer (2012)
24. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: Proc. of KR 2010. AAAI Press (2010)
25. Sequeda, J.F., Miranker, D.P.: Ultrawrap: SPARQL execution on relational data. Tech. Rep. TR-12-10, University of Texas at Austin. Department of Computer Science (2012)