Implementing Controlled Query Evaluation in OBDA

Divya Baura[1,*], Diego Calvanese[1,2] and Lorenzo Marconi[3]

[1]Umeå Universitet, Umeå, Sweden
[2]Free University of Bozen-Bolzano, Bolzano, Italy
[3]Sapienza Università di Roma , Rome, Italy

Abstract

In the *Ontology Based Data Access* (OBDA) framework, users access a relational data source by querying a domain ontology, whose classes and properties are connected to the data via declarative mappings. OBDA is adopted for data management in various sectors, notably healthcare, where confidentiality of information is a key concern that requires data to be properly protected from unauthorized accesses. *Controlled Query Evaluation* (CQE) is a framework for privacy-preserving query answering in the presence of an ontology. In CQE, *policies* are used to represent the information that should be kept confidential, and the aim is to devise from policy specifications suitable *censors* that enforce data protection. Therefore, it is desirable to integrate CQE in OBDA to obtain a robust privacy-aware data management framework. This has been done in the recently proposed *Policy-Protected OBDA* (PPOBDA) framework, which ensures the integration of CQE within OBDA by embedding policies into mappings. In this paper, we present an open-source solution that implements PPOBDA and a simplified algorithm for policy embedding, compared to previously proposed ones. This facilitates the adoption of PPOBDA using any OBDA query engine capable of translating SPARQL queries into SQL. In our implementation, we rely on Ontop, a state-of-the-art open-source OBDA tool.

Keywords

Ontology Based Data Access, Controlled Query Evaluation, Policy-Protected OBDA, Privacy, Ontop

1. Introduction

Within the Ontology-Based Data Access (OBDA) framework [1, 2], illustrated in the left part of Figure 1, an ontology encapsulates relevant domain knowledge and provides to users a vocabulary of classes and properties over which they can formulate queries. In OBDA, domain knowledge is typically expressed in OWL 2 QL, a profile (i.e., fragment) of the Web Ontology Language OWL 2, standardized by the W3C [3], while the actual data is stored in a relational source, which is linked to the ontology through declarative mappings, expressed in the R2RML mapping language [4]. Intuitively, mapping assertion specify how to populate the classes and properties of the ontology by means of data retrieved through SQL queries over the source.

Guaranteeing the privacy of information represents a challenge across all data management systems, especially in those cases where sensitive data, such as medical records, need to be

Proceedings of the Joint Ontology Workshops (JOWO) - Episode X: The Tukker Zomer of Ontology, and satellite events co-located with the 14th International Conference on Formal Ontology in Information Systems (FOIS 2024), July 15-19, 2024, Enschede, The Netherlands.

*Corresponding author.
divya.baura@umu.se (D. Baura); diego.calvanese@unibz.it (D. Calvanese); marconi@diag.uniroma1.it (L. Marconi)

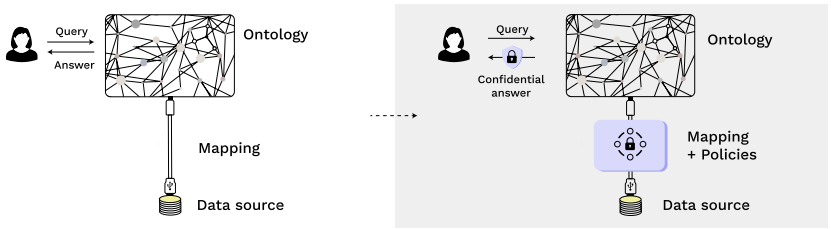

Figure 1: OBDA vs Policy-Protected OBDA

manipulated. This holds in particular for OBDA, where the main objective is to efficiently and effectively answer queries posed by users over the ontology, but transferring the necessary data from the underlying data source requires suitable privacy preserving methods that prevent any unauthorized disclosure of sensitive data. Recently, *Controlled Query Evaluation* (CQE) has emerged as a promising privacy-preserving framework for query answering in the presence of ontologies [5, 6, 7]. In CQE, *policies* represent confidential information and a *censor* protects these policies from being violated, by suitably modifying query answers. Therefore, the inclusion of CQE within OBDA would help in improving data confidentiality, reducing risks of unauthorized data access, and ensuring compliance with regulatory privacy frameworks.

Building upon the foundational principles of CQE and of OBDA, the *Policy-Protected OBDA* (PPOBDA) framework was introduced in [8]. In PPOBDA, policies are denial assertions formulated as first-order logic (FOL) formulas. These policies are then integrated into mapping assertions, thus establishing new mappings that are policy-protected, i.e., they ensure that policies are not violated when answering queries over the ontology by accessing the underlying data source (cf. Fig. 1). This framework has showcased promising results, and its first implementation highlights its potential to enhance privacy preservation in the OBDA settings. However, this initial implementation of PPOBDA relied on closed-source software, specifically the Mastro system [9], as the underlying OBDA query engine, which represents a limitation towards the wide adoption of PPOBA and the experimentation with such a framework.

To address this limitation, our ongoing research implements PPOBDA using the open-source OBDA engine Ontop [10, 11], by relying on its query rewriting functionalities for realizing most of the functions needed for PPOBDA. In this paper, we present the technical challenges that we faced in the implementation process. We also propose a simplified algorithm for embedding policies into mappings, with respect to the one presented in [8].

2. Preliminaries

In this section, we present the technical preliminaries that are necessary to understand the remaining part of the paper. Specifically, we introduced Description Logics, the logical-based formalism providing the underpinning for the OWL 2 language, and present the formalization of OBDA, and the query language used to express both queries in OBDA and policies in CQE. Finally, we introduce the open-source system Ontop, on which we rely for query reformulation.

We make use of countably infinite pairwise disjoint alphabets Σ_R of relation names, Σ_O of unary and binary ontology predicates, and Σ_C of constants.

2.1. Description Logics

Description Logics (DLs) [12] are a family of logics widely adopted in knowledge representation and reasoning. A DL can be considered as a decidable fragment of FOL, typically restricted to unary and binary predicates, respectively called *concepts* and *roles*. Concepts denote sets of objects, while roles denote binary relationships between objects. In a DL, knowledge is represented in a *knowledge base* (or *ontology*) $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, which consists of two components: a TBox $\mathcal{T}$, used to represent intensional knowledge, and an ABox $\mathcal{A}$, used to represent facts about individual objects. In OBDA, ontologies are typically expressed in some variant of *DL-Lite*, which is a family of lightweight DLs that are specifically tailored towards efficient data access. Specifically, we adopt *DL-Lite*$_\mathcal{R}$ [13], which provides the logical underpinning for the OWL 2 QL profile of OWL 2 [3]. In *DL-Lite*$_\mathcal{R}$, a TBox $\mathcal{T}$ consists of axioms of the form:

$B_1 \sqsubseteq B_2$ (concept inclusion) $\qquad B_1 \sqsubseteq \neg B_2$ (concept disjointness)

$Q_1 \sqsubseteq Q_2$ (role inclusion) $\qquad Q_1 \sqsubseteq \neg Q_2$ (role disjointness)

Here and in the following, we use Q (possibly with a subscript) to denote either an atomic role (i.e., a role name $R/2 \in \Sigma_O$), or an inverse role R^-, and B (possibly with a subscript) to denote either an atomic concept (i.e., a concept name $C/1 \in \Sigma_O$) or an unqualified existential restriction, which is a concept of the form $\exists Q$. An ABox $\mathcal{A}$ is a finite set of ground atoms, comprising assertions of the form $C(c)$ or $R(c, c')$, where $C/1, R/2 \in \Sigma_O$, and $c, c' \in \Sigma_C$.

The semantics of *DL-Lite*$_\mathcal{R}$ is given in terms of FOL interpretations, where an *interpretation* $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$, consists of a domain $\Delta^\mathcal{I}$, and a function $\cdot^\mathcal{I}$ that assigns to each concept name $C \in \Sigma_O$ a set $C^\mathcal{I} \subseteq \Delta^\mathcal{I}$ of objects, to each role name $R \in \Sigma_O$ a binary relation $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, and to each constant $c \in \Sigma_C$ a domain element $c^\mathcal{I} \in \Delta^\mathcal{I}$. We make the *unique name assumption*, i.e., $c \neq c'$ implies $c^\mathcal{I} \neq c'^\mathcal{I}$. Concept and role expressions are interpreted as follows:

$(R^-)^\mathcal{I} = \{(o', o) \mid (o, o') \in R^\mathcal{I}\}$ $\qquad \neg B^\mathcal{I} = \Delta^\mathcal{I} \setminus B^\mathcal{I}$

$(\exists Q)^\mathcal{I} = \{o \mid \exists o' \in \Delta^\mathcal{I} \text{ s.t. } (o, o') \in Q^\mathcal{I}\}$ $\qquad \neg Q^\mathcal{I} = (\Delta^\mathcal{I} \times \Delta^\mathcal{I}) \setminus Q^\mathcal{I}$

We say that $\mathcal{I}$ *satisfies* a concept/role inclusion/disjointness $E_1 \sqsubseteq E_2$, if $E_1^\mathcal{I} \subseteq E_2^\mathcal{I}$, and it satisfies an ABox assertion $C(c)$ (resp., $R(c, c')$) if $c^\mathcal{I} \in C^\mathcal{I}$ (resp., $(c^\mathcal{I}, c'^\mathcal{I}) \in R^\mathcal{I}$). An interpretation that satisfies all axioms and assertions in $\mathcal{K}$ (resp., $\mathcal{T}$, $\mathcal{A}$) is called a *model* of $\mathcal{K}$ (resp., $\mathcal{T}$, $\mathcal{A}$).

2.2. Ontology Based Data Access

In OBDA, users can query a data source through an ontology TBox, which is connected to the data source via declarative mappings [1, 2]. We formalize this through the notion of *OBDA specification*, which is a triple $\mathcal{O} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{T}$ denotes a *DL-Lite*$_\mathcal{R}$ TBox, $\mathcal{S}$ a relational database (DB) schema over the alphabet Σ_R, and $\mathcal{M}$ a mapping between $\mathcal{T}$ and $\mathcal{S}$. The *mapping* $\mathcal{M}$ is a finite set of mapping assertions from $\mathcal{S}$ to $\mathcal{T}$, where each *mapping assertion* m has form $\varphi(\vec{x}) \rightsquigarrow \psi(\mathsf{IRI}(\vec{x}))$. Here, $\varphi(x)$ denotes a FOL (or SQL) source query over $\mathcal{S}$ with answer variables $\vec{x}$. Instead, $\psi(\mathsf{IRI}(\vec{x}))$, called the *head* of m, is an ABox atom over the variables in $\vec{x}$ and so-called *IRI-templates* $\mathsf{IRI}(\vec{x})$. Each IRI-template $\mathsf{iri}(\vec{x})$ in $\mathsf{IRI}(\vec{x})$ is a term that concatenates string values and the answer variables in $\vec{x}$, and is used to construct object identifiers (i.e., IRIs) from the DB values returned by the source query $\varphi(\vec{x})$. A concrete mapping language that provides such form of mappings is R2RML, standardized by the W3C [4].

Given an OBDA specification $\mathcal{O} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ and a DB instance $\mathcal{D}$ for $\mathcal{S}$, the pair $\mathcal{J} = \langle \mathcal{O}, \mathcal{D} \rangle$ is called an *OBDA instance*. The *retrieved ABox* for $\mathcal{J}$, denoted $\mathsf{ret}(\mathcal{J})$, consists of all facts $\psi(\mathsf{IRI}(\vec{t}))$, where $\varphi(\vec{x}) \rightsquigarrow \psi(\mathsf{IRI}(\vec{x}))$ is a mapping assertion in $\mathcal{M}$, and $\vec{t} \in \mathsf{eval}(\varphi(\vec{x}), \mathcal{D})$ is a tuple of constants in the evaluation of the mapping source query over $\mathcal{D}$. Notice that the fact $\psi(\mathsf{IRI}(\vec{t}))$ contains IRIs of the form $\mathsf{iri}(\vec{t})$ constructed from the answer tuple $\vec{t}$ using the IRI template $\mathsf{iri}(\vec{x})$ in the mapping head. Hence, $\mathsf{ret}(\mathcal{J})$ is an ABox over constants in the set $\Delta_{\mathcal{J}}$ consisting of *(i)* all DB values in $\mathcal{D}$ and *(ii)* all possible IRIs $\mathsf{iri}(\vec{t})$ constructed from some tuple $\vec{t}$ of values in $\mathcal{D}$ and some IRI template $\mathsf{iri}(\vec{x})$ in some mapping assertion in $\mathcal{M}$. Then, a *model* of the OBDA instance $\mathcal{J}$ is defined as a model of the knowledge base $\langle \mathcal{T}, \mathsf{ret}(\mathcal{J}) \rangle$. We denote the set of models of $\mathcal{J}$ by $\mathsf{Mod}(\mathcal{J})$, and we say that $\mathcal{J}$ is *inconsistent* if $\mathsf{Mod}(\mathcal{J}) = \emptyset$. Moreover, $\mathcal{J} \models \alpha$, indicating that $\mathcal{J}$ entails a sentence α, holds if α is true in every model in $\mathsf{Mod}(\mathcal{J})$.

We make here the *standard name assumption*, i.e., given $\langle \mathcal{O}, \mathcal{D} \rangle$, we consider interpretations over a fixed domain Δ containing $\Delta_{\mathcal{J}}$ and such that all values in $\Delta_{\mathcal{J}}$ are interpreted as themselves. Notice that the standard name assumption implies the unique name assumption.

2.3. Queries

We consider queries, expressed as FOL formulas, over DB and OBDA instances. A query q over an OBDA instance $\mathcal{J}$ is formulated over the ontology predicates, and we are interested in the *certain answers* $\mathsf{cert}(q, \mathcal{J})$ to q, which are defined as those answers that hold over all models of $\mathcal{J}$, i.e., $\vec{t} \in \mathsf{cert}(q, \mathcal{J})$ iff $\mathcal{J} \models q(\vec{t})$. A *conjunctive query* (CQs) has the form $q(\vec{x}) = \exists \vec{y}.\alpha_1(\vec{x}, \vec{y}) \wedge \cdots \wedge \alpha_n(\vec{x}, \vec{y})$, where each α_i is a DB/ontology predicate. A *Boolean CQ* (BCQ) is a CQ $q()$ without answer variables, i.e., of zero arity, which returns either the empty tuple $\{()\}$ (i.e., *true*), or the empty set $\emptyset$ (i.e., *false*). A *ground atom* (GA) is a BCQ with only one atom and no variables. **CQ** denotes the language of BCQs, and **GA** the language of GAs.

2.4. The OBDA System Ontop

We make use here of the state-of-the-art open-source OBDA system Ontop [10, 11], which implements query answering over an OBDA instance, by adopting a query transformation approach and advanced query optimization techniques. Specifically, an Ontop installation operates over an OBDA specification $\mathcal{O} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{T}$ is an OWL 2 QL TBox, $\mathcal{M}$ an R2RML [4] mapping, and $\mathcal{S}$ a relational schema with constraints. It is able to efficiently answer SPARQL queries[1] [14] over the OBDA instance $\langle \mathcal{O}, \mathcal{D} \rangle$, where $\mathcal{D}$ is a DB instance for $\mathcal{S}$, according to the OWL 2 QL entailment regime [15]. Ontop is compliant with all relevant W3C standards, and supports all major commercial and free relational DB engines (and also several data federation tools, such as Denodo, Dremio, and Teiid, which expose a collection of heterogeneous data sources via a single relational DB schema).

Ontop implements query answering by rewriting, i.e., it computes the certain answers to SPARQL queries over the OBDA instance, by reformulating them into SQL queries expressed over $\mathcal{S}$, which then get executed by the underlying DB engine. To do so, Ontop first performs some off-line tasks, in which it pre-processes $\mathcal{T}$, $\mathcal{S}$, and $\mathcal{M}$ (e.g., by saturating $\mathcal{M}$ w.r.t. $\mathcal{T}$), in

[1] We can consider SPARQL as a concrete syntax for CQs, although SPARQL features additional constructs not present in CQs (e.g., `OPTIONAL` and aggregations), and adopts a different semantics for existentially quantified variables.

order to be then more efficient during query answering. At query answering time, it transforms a given SPARQL query q into a logically equivalent (w.r.t. $\mathcal{O}$) SQL query q_{SQL} and optimizes such query by taking into account mapping information and the DB constraints in S.

Ontop's toolkit encompass a Plugin for the widely adopted Protégé ontology editor, facilitating the development of OBDA specifications. Moreover, it allows for setting up a SPARQL endpoint, via a command line interface, with which users can interact by issuing queries via HTTP requests. In our work, we have used Ontop as a blackbox to exploits its query rewriting functionalities.

3. Encoding a Policy into a Mapping

To create a privacy-preserving OBDA setting, one can proceed in three ways: *(i)* the *ontology* can be modified by adjusting TBox axioms based on policy requirements [16]; *(ii)* the answers of *queries* can be modified through a censor, leading to Controlled Query Evaluation, which is an extensively explored topic [6, 17]; *(iii) mappings* can be modified according to policies, user authorization rights, or so that instances are anonymized. We follow here the third approach, on the one hand because of the promising outcomes demonstrated by the existing PPOBDA framework, e.g., for query evaluation over the NPD Benchmark [8], on the other hand because of the flexibility that manipulating mappings gives to introduce more privacy aspects.

To introduce formally the PPOBDA setting, we first define a *denial (assertion)* as a FOL sentence of the form $\forall\vec{x}.\varphi(\vec{x}) \rightarrow \bot$, such that $\exists\vec{x}.\varphi(\vec{x})$ is a BCQ. Given a set $\mathcal{O}$ of FOL sentences (e.g., a TBox) and a denial δ, we have that $\mathcal{O} \cup \{\delta\}$ is *consistent* if $\mathcal{O} \not\models \exists\vec{x}.\varphi(\vec{x})$. Following [8], we define a *PPOBDA specification* as a 4-tuple $\mathcal{E} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$, such that $\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ is an OBDA specification and $\mathcal{P}$ a *policy*, i.e., a finite set of denials over the signature of $\mathcal{T}$, such that $\mathcal{T} \cup \mathcal{P}$ is consistent. The semantics of a PPOBDA specification coincides with that of the underlying OBDA specification, and we naturally extend to PPOBDA also all other notions (source DB $\mathcal{D}$, instance $\langle \mathcal{E}, \mathcal{D} \rangle$, retrieved ABox $\mathsf{ret}(\langle \mathcal{E}, \mathcal{D} \rangle)$, and set $\mathsf{Mod}(\langle \mathcal{E}, \mathcal{D} \rangle)$ of models).

For a query language $\mathcal{L}$ (e.g., **CQ** or **GA**), let $\mathcal{L}(\mathcal{T})$ be the restriction of $\mathcal{L}$ to the predicates in $\mathcal{T}$, and $\mathcal{L}_\mathcal{D}$ the formulas in $\mathcal{L}$ mentioning only constants in $\mathcal{D}$. An *optimal censor* for $\mathcal{E}$ in query language $\mathcal{L}$ is a function $\mathsf{cens}(\cdot)$ that, for each source DB $\mathcal{D}$ for $\mathcal{E}$, returns a set $\mathsf{cens}(\mathcal{D}) \subseteq \mathcal{L}_\mathcal{D}$ such that *(i)* $\langle \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, \mathcal{D} \rangle \models \varphi$, for each $\varphi \in \mathsf{cens}(\mathcal{D})$, and *(ii)* $\mathcal{T} \cup \mathcal{P} \cup \mathsf{cens}(\mathcal{D})$ is consistent. $\mathcal{L}$ is called the *censor language*. We are interested in *optimal* censors, i.e., that return as much formulas as possible. The set of all optimal censors in $\mathcal{L}$ for a PPOBDA specification $\mathcal{E}$ is denoted $\mathcal{L}\text{-}\mathsf{OptCens}_\mathcal{E}$.

To obtain a notion of censor that allows for embedding a policy into the mapping, [8] have defined censors that approximate censors for $\mathcal{E}$ in **GA**, the language of ground atoms [7].

Definition 1 (IGA censor [8]). Given a PPOBDA specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$, the *intersection GA (IGA)* censor for $\mathcal{E}$ is the function $\mathsf{cens}_{\mathrm{IGA}}(\cdot)$ such that, for every DB instance $\mathcal{D}$ for $\mathcal{S}$, $\mathsf{cens}_{\mathrm{IGA}}(\mathcal{D}) = \bigcap_{\mathsf{cens} \in \mathbf{GA}\text{-}\mathsf{OptCens}_\mathcal{E}} \mathsf{cens}(\mathcal{D})$. ◁

Hence, an IGA censor, when applied to a DB instance $\mathcal{D}$ for source schema $\mathcal{S}$ of $\mathcal{E}$, returns the intersection of the sets of ground atoms computed by all optimal censors.

In [8], an algorithm, called PolicyEmbed is presented, that embeds a policy $\mathcal{P}$ into a OBDA mapping $\mathcal{M}$ and generates a new mapping $\mathcal{M}'$. Such mapping has the property that all answers

Algorithm 1: EncodeMapping

Input: a *DL-Lite*$_\mathcal{R}$ TBox $\mathcal{T}$, a mapping $\mathcal{M}$, a policy $\mathcal{P}$.
Output: a mapping $\mathcal{M}_0$.

Let $\mathcal{P}'$ be the expansion of the policy $\mathcal{P}$ w.r.t. $\mathcal{T}$;
$\mathcal{M}_0 \leftarrow \emptyset$;
for each atomic concept C **do**
  $\psi \leftarrow \mathsf{addConstraints}(C(x), \mathcal{P}')$;
  $\mathcal{M}' \leftarrow \mathcal{M}' \cup \{(\mathsf{unfold}(\mathsf{rewrite}(\psi, \mathcal{T}), \mathcal{M}) \rightsquigarrow C(x)\}$
for each atomic role R **do**
  $\psi \leftarrow \mathsf{addConstraints}(R(x, y), \mathcal{P}')$;
  $\mathcal{M}' \leftarrow \mathcal{M}' \cup \{(\mathsf{unfold}(\mathsf{rewrite}(\psi, \mathcal{T}), \mathcal{M}) \rightsquigarrow R(x, y)\}$
return $\mathcal{M}'$

returned to queries posed over the ontology and processed by an OBDA engine making use of $\mathcal{M}$' automatically comply to $\mathcal{P}$ according to an IGA censor. We present in Algorithm 1 EncodeMapping, a streamlined version of that algorithm that reduces the number of calls to a *DL-Lite*$_\mathcal{R}$ query rewriting procedure. We discuss its functioning on the following example:

$$\begin{array}{rcl} \mathcal{T} & = & \{\ \mathit{Reviewer} \sqsubseteq \mathit{Student}, \quad \exists \mathit{ReviewsProject}^- \sqsubseteq \mathit{Student}\ \}, \\ \mathcal{M} & = & \{\ m_1 : \exists y.\mathit{student}(x, y) \rightsquigarrow \mathit{Student}(x) \\ & & \quad m_2 : \mathit{student}(x, \texttt{'review'}) \rightsquigarrow \mathit{Reviewer}(x) \\ & & \quad m_3 : \mathit{project}(x, y) \rightsquigarrow \mathit{ReviewsProject}(x, y)\ \} \\ \mathcal{P} & = & \{\ \mathit{Reviewer}(x) \wedge \mathit{ReviewsProject}(x, y) \wedge \mathit{Student}(y) \rightarrow \bot\ \} \end{array}$$

Here, the TBox $\mathcal{T}$ specifies that reviewer students (concept $\mathit{Reviewer}$) are special kinds of students (concept $\mathit{Student}$), and that projects of students are being reviewed (inverse of role $\mathit{ReviewsProject}$). The policy says that the fact that a reviewer reviews a project of a student is a confidential information (to protect the information regarding who graded whom).

Following the definition of the IGA censor, the target is to remove the facts that belong to at least one minimal ABox $\mathcal{A}$ such that $\mathcal{T} \cup \mathcal{A} \cup \mathcal{P}$ is inconsistent. Identifying such facts is facilitated when we can analyze each denial independently. This requires that the policy P exhibits the following property: for every denial δ in $\mathcal{P}$, every minimal ABox $\mathcal{A}$ where $\delta \cup \mathcal{T} \cup \mathcal{A}$ is inconsistent must also be minimal when $\mathcal{P} \cup T \cup \mathcal{A}$ is inconsistent.

To achieve this, [8] introduces the concept of an *extended denial*, formulated as $\forall x.\varphi(x) \wedge \neg\pi(x) \rightarrow \bot$, where $\exists x.\varphi(x)$ forms a BCQ and $\pi(x)$ is a disjunction of conjunctions of equality atoms. The extended policy is a finite set of extended denials, enabling the transformation of the initial policy into an updated policy $\mathcal{P}$ satisfying the above mentioned property.

In Step 1, we use the perfectRef$(\mathcal{P}, \mathcal{T})$ algorithm of *DL-Lite*$_\mathcal{R}$ to expand policy $\mathcal{P}$ with respect to $\mathcal{T}$ [13]. The existing policy expands as follows:

$$\begin{array}{l} p_1 : \mathit{Reviewer}(x) \wedge \mathit{ReviewsProject}(x, y) \wedge \mathit{Student}(y) \rightarrow \bot \\ p_2 : \mathit{Reviewer}(x) \wedge \mathit{ReviewsProject}(x, y) \wedge \mathit{Reviewer}(y) \rightarrow \bot \\ p_3 : \mathit{Reviewer}(x) \wedge \mathit{ReviewsProject}(x, y) \rightarrow \bot \end{array}$$

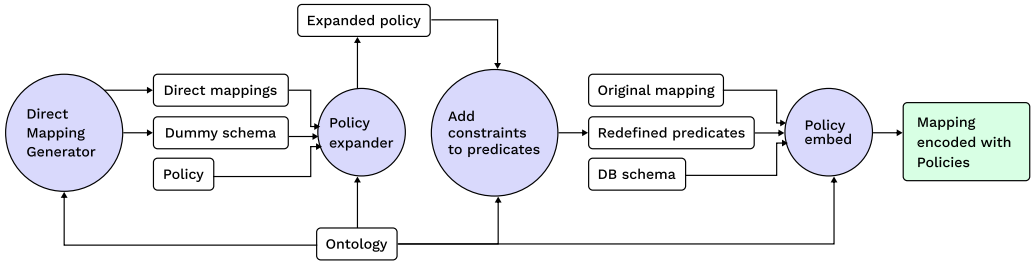

Figure 2: Workflow for PPOBDA implementation

Since p_3 implies p_1 and p_2, we discard p_1 and p_2 and $\mathcal{P}' = \{p_3\}$.

Then, one mapping assertion is constructed for each ontology predicate.

At Step 4, for each concept C of the ontology, addConstraints$(C(x), \mathcal{P}')$ transforms $C(x)$ into a formula ensuring that the conditions leading to a policy violation cannot be satisfied, when retrieving data from the source through the mapping. Specifically, assume that all denials in $\mathcal{P}'$ containing an atom that unifies with $C(x)$ are $\forall x, \vec{y}_i.(C(x) \wedge \alpha_i(x, \vec{y}_i)) \rightarrow \bot$, for $i \in \{1, \ldots, k\}$. Then addConstraints$(C(x), \mathcal{P}')$ returns $C(x) \wedge \bigwedge_{1 \leq i \leq k} \neg \exists \vec{y}_i.\alpha_i(x, \vec{y}_i)$. Similarly, at Step 7, for each role R of the ontology. For instance, assume that $\mathcal{P}'$ contains the denials $\forall x.(C(x) \wedge D(x)) \rightarrow \bot$ and $\forall x, y.(C(x) \wedge R(x, y) \wedge E(y)) \rightarrow \bot$. Then, addConstraints$(C(x), \mathcal{P}')$ returns $C(x) \wedge \neg D(x) \wedge \neg \exists y.(R(x, y) \wedge E(y))$. For predicates that are not present in the policy $\mathcal{P}'$, we do not apply the transformation, as they pose no threat to the policy. In our running example, $Student(x)$ is not transformed, while

$$\begin{aligned} &\text{addConstraints}(Reviewer(x), \mathcal{P}') &&= Reviewer(x) \wedge \neg \exists y.ReviewsProject(x, y) \\ &\text{addConstraints}(ReviewsProject(x, y), \mathcal{P}') &&= ReviewsProject(x, y) \wedge \neg Reviewer(x) \end{aligned}$$

Steps 5 and 8 invoke rewrite$(\psi, \mathcal{T})$, which rewrites each of the transformed predicates w.r.t. the TBox $\mathcal{T}$. Notice that ψ might contain negated atoms, and for this step we rely on the capability of the query rewriting engine to correctly deal with SPARQL queries containing the `MINUS` operator. The resulting expression is then unfolded with respect to the original mapping $\mathcal{M}$ [1], to obtain the source query of the new mapping $\mathcal{M}'$ for the concept C or role R. For this, we again rely on the OBDA engine.

In our running example, EncodeMapping$(\mathcal{T}, \mathcal{M}, \mathcal{P})$ returns the following mapping $\mathcal{M}'$:

$$\begin{aligned} &m'_1 : \exists y.\textit{student}(x, y) \vee \textit{student}(x, \texttt{'review'}) \vee \exists y.\textit{project}(y, x) \rightsquigarrow Student(x) \\ &m'_2 : student(x, 'review') \wedge \neg \exists y.\textit{project}(x, y)) \rightsquigarrow Reviewer(x) \\ &m'_3 : \textit{project}(x, y) \wedge \neg \textit{student}(x, \texttt{'review'}) \rightsquigarrow ReviewsProject(x, y) \end{aligned}$$

4. Implementation in the Ontop System

To implement the EncodeMapping algorithm, we need to exploit the query answering capabilities of an OBDA engine, and specifically both query rewriting with respect to an OWL 2 QL TBox, and query unfolding with respect to R2RML mappings. The original implementation of the PPOBDA framework described in [8] relied on the Mastro system [9]. That tool is, however,

a proprietary software that is not openly accessible, therefore the implementation described in [8] is not available for experimentation and for possible extensions. For this reason, we have reimplemented the PPOBDA framework from scratch, by relying on an open-source OBDA engine, and specifically on the Ontop system [10, 11]. The workflow we followed for our implementation is depicted in Fig. 2.

The initial step of expanding the policy w.r.t. the TBox $\mathcal{T}$ relies on the query rewriting functionality. Since for optimization purposes, Ontop combines the steps of query rewriting and of unfolding w.r.t. the mapping, it requires the existence of a relational data source and of mappings even to perform only query rewriting. To address this, we have developed a *Direct Mapping Generator*, which simulates a data source with unary and binary relations corresponding directly to the concepts and roles names in the ontology, and establishes direct (one-to-one) mappings from this dummy DB to the ontology. The *Policy Expander* function exploits this setup to activate the Ontop query reformulation functionality over the denials in the policy $\mathcal{P}$ as queries to be expanded. Given that the mappings are one-to-one, they have no impact on the outcome of the reformulation.

The rewritten queries, serving as policies expanded w.r.t. the ontology, are then provided as input to the addConstraints algorithm, which redefines each predicate as a SPARQL query. These SPARQL queries, along with the ontology, the original mapping, and the DB schema are then processed by the *Policy Embed* function, which converts them into SQL queries using Ontop for query reformulation. Subsequently, a new mapping is created with these SQL queries as the source parts for the respective predicates, which appear in the target part of the mapping assertions. This process effectively generates mappings that embed the policy constraints.

5. Conclusions

We have described an ongoing research effort that aims at extending the OBDA framework so as to incorporate privacy policies expressed as denials, in line with the Controlled Query Evaluation (CQE) approach. Following an approach in the literature, we have provided an initial implementation in a prototype system that builds on the open-source OBDA system Ontop. Our implementation is available as an open-source project[2]. We rely on our prototype implementation in our ongoing research, which aims at analyzing privacy requirements in real-world scenarios, and at assessing the adequacy of the CQE approach to capture them. We also plan to investigate how the approach we have presented for embedding policies into mappings impacts performance of query evaluation in OBDA, specifically when compared to the original approach proposed in [8].

Acknowledgments

This work has been partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, by the Province of Bolzano and DFG through the project D2G2 (DFG grant n. 500249124), and by the HEU project CyclOps (under GA n. 101135513).

[2] https://github.com/divyabaura/PPOBDA-with-Ontop

References

[1] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, Linking data to ontologies, J. on Data Semantics 10 (2008). doi:10.1007/978-3-540-77688-8_5.

[2] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, M. Zakharyaschev, Ontology-based data access: A survey, in: Proc. IJCAI, IJCAI Org., 2018, pp. 5511–5519. doi:10.24963/ijcai.2018/777.

[3] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, C. Lutz, OWL 2 Web Ontology Language Profiles (Second Edition), W3C Recommendation, W3C, 2012.

[4] S. Das, S. Sundara, R. Cyganiak, R2RML: RDB to RDF Mapping Language, W3C Recommendation, W3C, 2012. Available at http://www.w3.org/TR/r2rml/.

[5] P. A. Bonatti, L. Sauro, A confidentiality model for ontologies, in: Proc. ISWC, volume 8218 of *LNCS*, Springer, 2013, pp. 17–32. doi:10.1007/978-3-642-41335-3_2.

[6] B. C. Grau, E. Kharlamov, E. V. Kostylev, D. Zheleznyakov, Controlled query evaluation for Datalog and OWL 2 Profile ontologies, in: Proc. IJCAI, AAAI Press, 2015, pp. 2883–2889.

[7] D. Lembo, R. Rosati, D. F. Savo, Revisiting controlled query evaluation in description logics, in: Proc. IJCAI, IJCAI Org., 2019, pp. 1786–1792. doi:10.24963/IJCAI.2019/247.

[8] G. Cima, D. Lembo, L. Marconi, R. Rosati, D. F. Savo, Controlled query evaluation in ontology-based data access, in: Proc. ISWC, volume 12506 of *LNCS*, Springer, 2020, pp. 128–146. doi:10.1007/978-3-030-62419-4_8.

[9] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, D. F. Savo, The Mastro system for ontology-based data access, Semantic Web J. 2 (2011) 43–53. doi:10.3233/SW-2011-0029.

[10] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, G. Xiao, Ontop: Answering SPARQL queries over relational databases, Semantic Web J. 8 (2017) 471–487. doi:10.3233/SW-160217.

[11] G. Xiao, D. Lanti, R. Kontchakov, S. Komla-Ebri, E. Güzel-Kalayci, L. Ding, J. Corman, B. Cogrel, D. Calvanese, E. Botoeva, The virtual knowledge graph system Ontop, in: Proc. ISWC, volume 12507 of *LNCS*, Springer, 2020, pp. 259–277. doi:10.1007/978-3-030-62466-8_17.

[12] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation and Applications, CUP, 2003.

[13] 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 (2007) 385–429. doi:10.1007/s10817-007-9078-x.

[14] S. Harris, A. Seaborne, SPARQL 1.1 Query Language, W3C Recommendation, W3C, 2013. Available at http://www.w3.org/TR/sparql11-query.

[15] B. Glimm, C. Ogbuji, SPARQL 1.1 Entailment Regimes, W3C Recommendation, W3C, 2013. Available at http://www.w3.org/TR/sparql11-entailment/.

[16] F. Baader, F. Kriegel, A. Nuradiansyah, R. Peñaloza, Repairing Description Logic Ontologies by Weakening Axioms, CoRR Technical Report arXiv:1808.00248, arXiv.org, 2018.

[17] P. A. Bonatti, G. Cima, D. Lembo, L. Marconi, R. Rosati, L. Sauro, D. F. Savo, Controlled query evaluation in OWL 2 QL: A "longest honeymoon" approach, in: Proc. ISWC, volume 13489 of *LNCS*, Springer, 2022, pp. 428–444. doi:10.1007/978-3-031-19433-7_25.