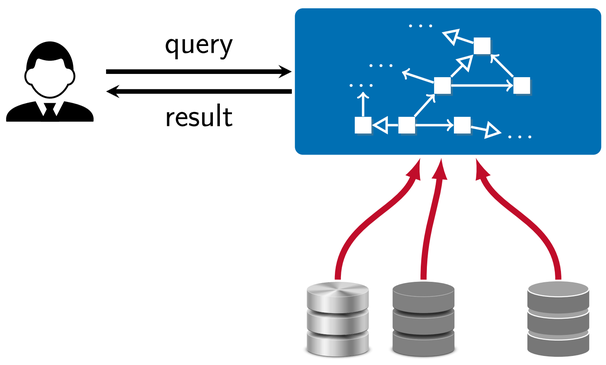
Beyond OWL 2 QL in OBDA: Rewritings and Approximations

Elena Botoeva, Diego Calvanese, Valerio Santarelli, Domenico F. Savo,
Alessandro Solimando, and Guohui Xiao

AAAI-16

Ontology Based Data Access – OBDA



Ontology

global vocabulary
conceptual view

Mappings

how to populate the ontology

Data Sources

external and heterogeneous

Logical transparency in accessing data:

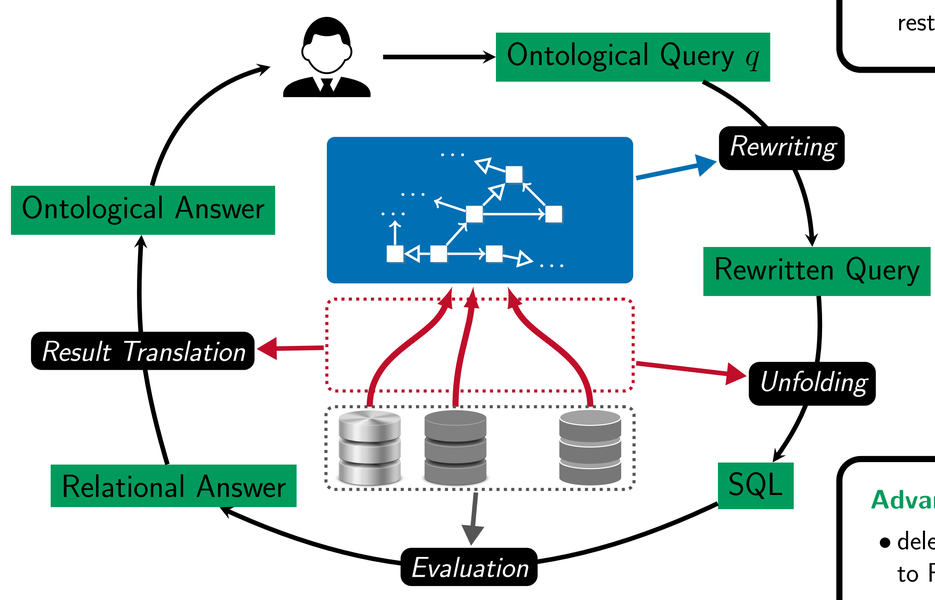
- does not know where and how **data** is stored;
- can only see a **conceptual view** of **data**.

An **OBDA specification** is a triple $\langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$, where

- $\mathcal{T}$ is a Description Logic TBox,
- $\mathcal{M}$ is a set of mapping assertions $\mathsf{SQL}_A(x) \rightsquigarrow A(x)$ and $\mathsf{SQL}_R(x, y) \rightsquigarrow R(x, y)$, and
- $\mathcal{S}$ is a relational schema.

Virtual Approach to OBDA

Query Answering is done by **Query Rewriting** into SQL.



Disadvantages:

- works only for first-order (FO) rewritable languages: the commonly adopted one is OWL 2 QL, which is too restrictive

Advantages:

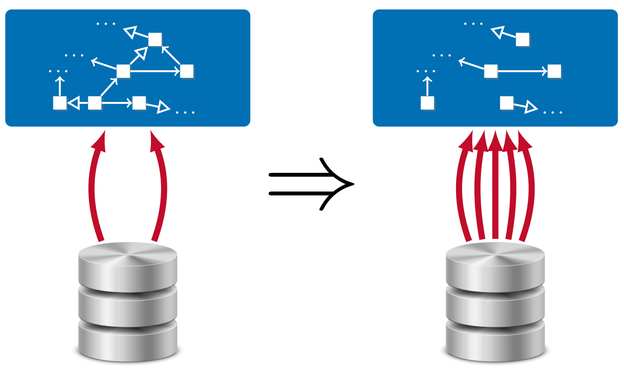
- delegates query evaluation to RDBMS
- avoids data materialization
- works well for Big Data

Beyond OWL 2 QL: Mappings to the Rescue

Problem We want to go beyond OWL 2 QL in Virtual OBDA.
However expressive ontologies are **not** FO-rewritable.

Solution Exploit the **mapping component** that makes use of arbitrary SQL queries.

We introduce a **framework** for rewriting and approximation of OBDA specifications.



Rewriting

The new specification is equivalent to the original one w.r.t. query answering (**query-inseparable**).

Approximation

The new specification is a sound approximation of the original one w.r.t. query answering.

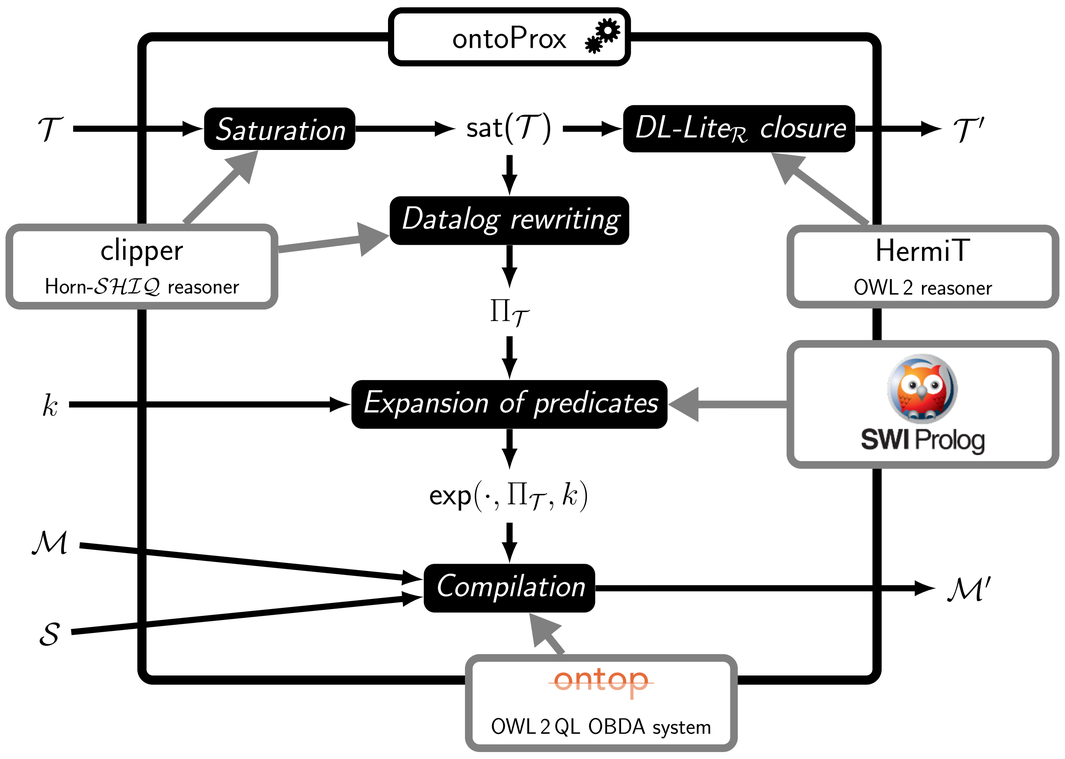
$\mathcal{T} = \{\ A \sqcap B \sqsubseteq C\ \}$
$\mathcal{M} = \{\ \mathsf{SQL}_A(x) \rightsquigarrow A(x),\ \mathsf{SQL}_B(x) \rightsquigarrow B(x)\ \}$
$\Rightarrow$
$\mathcal{T}' = \{\ \}$
$\mathcal{M}' = \{\ \mathsf{SQL}_A(x) \rightsquigarrow A(x),\ \mathsf{SQL}_B(x) \rightsquigarrow B(x),\ \mathsf{SQL}_A(x) \wedge \mathsf{SQL}_B(x) \rightsquigarrow C(x)\ \}$

$\mathcal{T} = \{\ \exists R.A \sqsubseteq C\ \}$
$\mathcal{M} = \{\ \mathsf{SQL}_A(x) \rightsquigarrow A(x),\ \mathsf{SQL}_R(x, y) \rightsquigarrow R(x, y)\ \}$
$\Rightarrow$
$\mathcal{T}' = \{\ \}$
$\mathcal{M}' = \{\ \mathsf{SQL}_A(x) \rightsquigarrow A(x),\ \mathsf{SQL}_R(x, y) \rightsquigarrow R(x, y),\ \mathsf{SQL}_R(x, y) \wedge \mathsf{SQL}_A(y) \rightsquigarrow C(x)\ \}$

$\mathcal{T} = \{\ \exists R.A \sqsubseteq A\ \}$
$\mathcal{M} = \{\ \mathsf{SQL}_A(x) \rightsquigarrow A(x),\ \mathsf{SQL}_R(x, y) \rightsquigarrow R(x, y)\ \}$
$\Rightarrow$
$\mathcal{T}' = \{\ \}$
$\mathcal{M}' = \{\ \mathsf{SQL}_A(x) \rightsquigarrow A(x),\ \mathsf{SQL}_R(x, y) \rightsquigarrow R(x, y),\ \mathsf{SQL}_R(x, y) \wedge \mathsf{SQL}_A(y) \rightsquigarrow A(x),\ \mathsf{SQL}_R(x, y) \wedge \mathsf{SQL}_R(y, z) \wedge \mathsf{SQL}_A(z) \rightsquigarrow A(x),\ \mathsf{SQL}_R(x, y) \wedge \mathsf{SQL}_R(y, z) \wedge \mathsf{SQL}_R(z, w) \wedge \mathsf{SQL}_A(w) \rightsquigarrow A(x)\ \}$

ontoProx

We have developed an algorithm for computing approximations and implemented it in a prototype system called **ontoProx**.



It takes as input a Horn-$\mathcal{SHIQ}$ TBox $\mathcal{T}$, a mapping $\mathcal{M}$ and an integer k, and produces a *DL-Lite*$_\mathcal{R}$ TBox $\mathcal{T}'$ and an extended mapping $\mathcal{M}'$.

Evaluation

We evaluated **ontoProx** over synthetic and real OBDA instances against

- the default **ontop** behavior,
- local semantic approximation (LSA),
- global semantic approximation (GSA), and
- **clipper** over materialized ABoxes.

The evaluation showed that we are able to obtain more answers using our approach (in fact, complete, whenever that could be verified by clipper).

http://ontop.inf.unibz.it
https://github.com/ontop/ontoprox
https://github.com/ghxiao/clipper

unibz KRDB Sapienza Università di Roma Università degli Studi di Genova
ontop Optique