Ontology-Based RDF Integration of Heterogeneous Data
Maxime Buron, François Goasdoué, Ioana Manolescu, Marie-Laure Mugnier

To cite this version:
Maxime Buron, François Goasdoué, Ioana Manolescu, Marie-Laure Mugnier. Ontology-Based RDF Integration of Heterogeneous Data. EDBT 2020 - 23rd International Conference on Extending Database Technology, Mar 2020, Copenhagen, Denmark. pp.299-310, 10.5441/002/edbt.2020.27 . hal-02446427

HAL Id: hal-02446427
https://inria.hal.science/hal-02446427
Submitted on 20 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives| 4.0 International License
Ontology-Based RDF Integration of Heterogeneous Data

Maxime Buron\textsuperscript{1}  François Goasdoué\textsuperscript{2}  Ioana Manolescu\textsuperscript{1}  Marie-Laure Mugnier\textsuperscript{3}
\textsuperscript{1}Inria and LIX (UMR 7161, CNRS and Ecole polytechnique), France  \textsuperscript{2}Univ. Rennes, CNRS, IRISA, France  \textsuperscript{3}Univ. Montpellier, LIRMM, Inria, France

ABSTRACT

The proliferation of heterogeneous data sources in many application contexts brings an urgent need for expressive and efficient data integration mechanisms. There are strong advantages to using RDF graphs as the integration format: being schemaless, they allow for flexible integration of data from heterogeneous sources; RDF graphs can be interpreted with the help of an ontology, describing application semantics; last but not least, RDF enables joint querying of the data and the ontology.

To address this need, we formalize RDF Integration Systems (RIS), Ontology Based-Data Access mediators, that go beyond the state of the art in the ability to expose, integrate and flexibly query data from heterogeneous sources through GLAV (global-local-as-view) mappings. We devise several query answering strategies, based on an innovative integration of LAV view-based rewriting and a form of mapping saturation. Our experiments show that one of these strategies brings strong performance advantages, resulting from a balanced use of mapping saturation and query reformulation.

1 INTRODUCTION

The proliferation of digital data sources across all application domains brings a new urgency to the need for tools which allow to query heterogeneous data (relational, JSON, key-values, graphs etc.) in a flexible fashion. Traditional data integration systems fall into two classes: data warehousing, where all data source content is materialized in a single centralized source, respectively, mediation, where data remains in its original stores and all data can be queried through a single module called mediator. Data warehousing simplifies query evaluation, but requires potentially costly maintenance operations when the content of data sources changes; mediation does not suffer from these drawbacks, but requires more intricate query evaluation algorithms to distribute the work between the sources and the mediator.

Below, we classify prior mediator-based approaches according to two main dimensions, and illustrate this classification in Table 1. The simplest mappings define each element of the global schema, e.g., each relation (if the global schema is relational), as a view over the local schemas; this is known as Global-As-View, or GAV in short. In a GAV system, a query over the global (virtual) schema is easily transformed into a query over the local schemas, by unfolding each global schema relation, i.e., replacing it with its definition. In contrast, Local-As-View (LAV) mappings define elements of the local schemas as views over the global one. Query answering in this context requires rewriting the query with the views describing the local sources [31]. Global-Local-As-View (GLAV) data integration generalizes both GAV and LAV. A GLAV mapping pairs a query \( q_1 \) over one or several local schemas to a query \( q_2 \) over the global schema, having the same answer variables. The semantics is that, for each answer of \( q_1 \), the integration system exposes the data comprised in a corresponding answer of \( q_2 \). GLAV maximizes flexibility, or, equivalently, integration expressive power: unlike LAV, a GLAV mapping may expose only part of a given source’s data, and may combine data from several sources; unlike GAV, a GLAV mapping may include joins or complex expressions over

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
Model & GAV & GLAV \\
\hline
CQ & [40, 41, 43] & [18] \\
SPARQL & [16, 33, 44] & this work \\
\hline
\end{tabular}
\caption{Table 1: Outline of the positioning of our work.}
\end{table}

A second dimension is how the source (or local) schemas are connected to the global (integration) schema, using mappings [23]. There are three types of mappings, each corresponding to a column in Table 1. The simplest mappings define each element of the global schema, e.g., each relation (if the global schema is relational), as a view over the local schemas; this is known as Global-As-View, or GAV in short. In a GAV system, a query over the global (virtual) schema is easily transformed into a query over the local schemas, by unfolding each global schema relation, i.e., replacing it with its definition. In contrast, Local-As-View (LAV) mappings define elements of the local schemas as views over the global one. Query answering in this context requires rewriting the query with the views describing the local sources [31]. Global-Local-As-View (GLAV) data integration generalizes both GAV and LAV. A GLAV mapping pairs a query \( q_1 \) over one or several local schemas to a query \( q_2 \) over the global schema, having the same answer variables. The semantics is that, for each answer of \( q_1 \), the integration system exposes the data comprised in a corresponding answer of \( q_2 \). GLAV maximizes flexibility, or, equivalently, integration expressive power: unlike LAV, a GLAV mapping may expose only part of a given source’s data, and may combine data from several sources; unlike GAV, a GLAV mapping may include joins or complex expressions over
the global schema.

In this work, we study GLAV mediation supporting SPARQL queries over the data and the ontology. We pick GLAV for its highest expressive power, RDF for its wide adoption, and aim at querying the data and the ontology in order to fully benefit from the flexibility and expressivity of RDF. As Table 1 shows, our system is the first capable of integrating multiple data sources through GLAV mappings, for SPARQL querying over the data and the ontology; further, it supports heterogeneous data sources (of different data models). A benefit of our using GLAV is the ability to support a form of incomplete information, naturally present in RDF through the so-called blank nodes, in the virtual RDF graph exposed by the mediator (see Section 3.1).

Our closest competitors only support GAV mappings, even though some support more expressive ontologies and/or queries [16, 33, 44]. Formal OBDA frameworks based on GLAV mappings have been defined, e.g., [18], without concretely deployed systems. A technique for simulating GLAV mappings through GAV ones under certain conditions is suggested in [21], however this solution has many drawbacks; we defer a detailed discussion to Section 6.

Contributions and novelty

The contributions we make in this work are as follows.

1. RIS Formalism

   We formally define RDF Integration Systems (RIS, in short). OBDA mediators capable of exposing data from heterogeneous sources of virtually any data model through GLAV mappings, under the form of an RDF graph endowed with an RDFS ontology. We formalize the problem of BGP (basic graph pattern) RDF query answering over the RDF data and ontology exposed in such systems.

2. Novel RIS query answering techniques

   We describe several RIS query answering methods based on transforming mappings into LA V view definitions, and on reducing query answering to rewriting it using views. Our first method combines known techniques; the other two methods are novel, and rely on a form of mapping saturation. We show that a smart decomposition of reasoning between offline precomputation and query time makes one of these methods much faster than the others.

   The paper is organized as follows. Section 2 recalls a set of preliminary notions we build upon. Then, Section 3 defines our RIS and formalizes RIS query answering. Section 4 describes RIS query answering methods. Section 5 presents our experiments, then we discuss related work and conclude.

2 PRELIMINARIES

We present the basics of the RDF graph data model (Section 2.1), of RDF entailment used to make explicit the implicit information RDF graphs encode (Section 2.2), and how RDF graphs can be queried using the widely-considered SPARQL Basic Graph Pattern queries (Section 2.3). Then, we recall two techniques, namely query reformulation (Section 2.4) and view-based query rewriting (Section 2.5), which will serve as building blocks for our query answering techniques.

2.1 RDF Graphs

We consider three pairwise disjoint sets of values: \( \mathcal{I} \) of IRIs (resource identifiers), \( \mathcal{L} \) of literals (constants) and \( \mathcal{B} \) of blank nodes modeling unknown IRIs or literals, a.k.a. labelled nulls [3, 29]. A (well-formed) triple belongs to \( (\mathcal{I} \cup \mathcal{L}) \cup (\mathcal{B} \cup \mathcal{I} \cup \mathcal{L}) \), and an RDF graph \( G \) is a set of (well-formed) triples. A triple \( (s, p, o) \) states that its subject \( s \) has the property \( p \) with the object value \( o \) [47]. We denote by \( \text{Val}(G) \) the set of all values (IRIs, blank nodes and literals) occurring in an RDF graph \( G \), and by \( \text{Bl}(G) \) its

<table>
<thead>
<tr>
<th>Schema triples</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclass</td>
<td>( (s, \prec_{cc}, o) )</td>
</tr>
<tr>
<td>Subproperty</td>
<td>( (s, \prec_{sp}, o) )</td>
</tr>
<tr>
<td>Domain typing</td>
<td>( (s, \rightarrow_{d}, o) )</td>
</tr>
<tr>
<td>Range typing</td>
<td>( (s, \rightarrow_{r}, o) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data triples</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class fact</td>
<td>( (s, r, o) )</td>
</tr>
<tr>
<td>Property fact</td>
<td>( (s, p, o) ) s.t. ( p \notin { r, \prec_{cc}, \prec_{sp}, \rightarrow_{d}, \rightarrow_{r} } )</td>
</tr>
</tbody>
</table>

Table 2: RDF triples.

set of blank nodes. In triples, we use \( _b \) (possibly with indices) to denote blank nodes, and quoted strings to denote literals.

Within an RDF graph, we distinguish data triples from schema ones. The former describe data (either attach a type, or a class, to a resource, or state the value of a certain data property of a resource). The latter state ontological constraints using RDF Schema (RDFS), which relate classes and properties: subclass (specialization relation between types), subproperty (specialization of a binary relation), typing of the domain (first attribute) of a property, respectively, range (typing of the second attribute) of a property. Table 2 introduces short notations we adopt for these schema properties.

From now on, we denote by \( \mathcal{F}_d \) the reserved IRIs from the RDF standard, e.g., the properties \( \prec_{cc}, \prec_{sp}, \rightarrow_{d}, \rightarrow_{r} \), shown in Table 2. The rest of the IRIs are application-dependent classes and properties, which are said user-defined and denoted by \( \mathcal{F}_{\text{user}} \).

Hence, \( \mathcal{F}_{\text{user}} = \mathcal{F} \setminus \mathcal{F}_d \).

We will consider RDF graphs with RDFS ontologies made of schema triples of the four above flavours. More precisely:

Definition 2.1 (RDFS ontology). An ontology triple is a schema triple whose subject and object are user-defined IRIs from \( \mathcal{F}_{\text{user}} \). An RDFS ontology (or ontology in short) is a set of ontology triples. Ontology \( O \) is the ontology of an RDF graph \( G \) if \( O \) is the set of schema triples of \( G \).

Above, ontology triples are not allowed over blank nodes. This is only to simplify the presentation; we could have allowed them, and handled them as in [29]. More importantly, we forbid ontology triples from altering the common semantics of RDF itself. For instance, we do not allow \( (\rightarrow_{d}, \prec_{sp}, \rightarrow_{r}) \), which would impose that the range of every property shares all the types of the property’s domain! This second restriction can be seen as common-sense; it underlies most ontological formalisms, in particular description logics [8] thus the W3C’s Web Ontology Language (OWL), Datalog+ [15] and existential rules [39], etc.

Example 2.2 (Running example, based on [12]).

Consider the following RDF graph:

\[
\text{Gex} = \{(\text{worksFor}, \rightarrow_{d}, \text{Person}), (\text{worksFor}, \rightarrow_{r}, \text{Org}),
(\text{:PubAdmin}, \prec_{cc}, \text{:Org}), (\text{:Comp}, \prec_{cc}, \text{:Org}),
(\text{:NatComp}, \prec_{cc}, \text{:Comp}), (\text{hiredBy}, \prec_{sp}, \text{worksFor}),
(\text{CEOOf}, \prec_{sp}, \text{worksFor}), (\text{CEOOf}, \rightarrow_{r}, \text{Comp}),
(\text{g1}, \text{CEOOf}, \_, \text{b}) \}, (\_, \text{b}, \text{r}, \text{NatComp}),
(\text{g2}, \text{hiredBy}, \_, \text{a}) \}, (\text{a}, \text{r}, \text{:PubAdmin})\}
\]

The ontology of \( \text{Gex} \), i.e., the first eight schema triples, states that people work for organizations, some of which are public administrations or companies. Further, national companies are a kind of companies. Being hired by or being CEO of an organization are two ways of working for it; in the latter case, this organization is a company. The facts of \( \text{Gex} \), i.e., the four remaining data triples, state that \( g_1 \) is CEO of some unknown company represented by the blank node \( _b \), which is a national company, and \( g_2 \) is hired by the public administration \( \_a \).
2.2 RDF Entailment Rules

An entailment rule \( r \) has the form body(\( r \)) \( \rightarrow \) head(\( r \)), where body(\( r \)) and head(\( r \)) are RDF graphs, respectively called body and head of the rule \( r \). In this work, we consider the RDF entailment rules \( \mathcal{R} \) shown in Table 3, which are the most frequently used; in the table, all values except RDF reserved IRIs are blank nodes. For instance, rule rdf5 states: whenever in an RDF graph, a property \( p_1 \) is a subproperty of a property \( p_2 \), then \( s \) also has \( o \) as a value for \( p_1 \). The triples \( (p_1, s, p_2) \) and \( (s, o, p_2) \) in the above examples are called implicit, i.e., they hold in a graph thanks to the entailment rules, even if they may not be explicitly present in the graph. Following [12], we view \( \mathcal{R} \) as partitioned into two subsets: the rules \( \mathcal{R}_c \) lead to implicit schema triples, while rules \( \mathcal{R}_a \) lead to implicit data triples. The direct entailment of an RDF graph \( G \) with a set of RDF entailment rules \( \mathcal{R} \), denoted by \( C_{\mathcal{R}} G \), is the set of implicit triples resulting from rule applications that use solely the explicit triples of \( G \). For instance, the rule rdf9 applies to the graph \( G_{\text{ex}} \), which comprises \( (\text{NatComp}, s_{\text{ex}}, \text{Comp}), (\_b_c, r, \text{NatComp}) \), leads to the implicit triple \( (\_b_c, r, \text{Comp}) \). This triple belongs to \( C_{\mathcal{R}_a} G_{\text{ex}} \) (hence also to \( C_{\mathcal{R}_c} G_{\text{ex}} \)). The saturation of an RDF graph allows materializing its semantics, by iteratively augmenting it with the triples it entails using entailment rules, until reaching a fixpoint; this process is finite [48]. Formally:

Definition 2.3 (RDF graph saturation). Let \( G \) be an RDF graph and \( \mathcal{R} \) a set of entailment rules. We recursively define a sequence \( (G_i)_{i \in \mathbb{N}} \) of RDF graphs as follows: \( G_0 = G \), and \( G_{i+1} = G_i \cup C_{\mathcal{R}} G_i \) for \( i \geq 0 \). The saturation of \( G \) w.r.t. \( \mathcal{R} \), denoted \( G^\mathcal{R} \), is \( G_n \) for \( n \) the smallest integer such that \( G_n = G_{n+1} \).

Example 2.4 (Saturation). The saturation of \( G_{\text{ex}} \) w.r.t. the set \( \mathcal{R} \) of RDFs entailment rules shown in Table 3 is attained after the following two saturation steps:

\[
\text{(Gex)}_1 = G_{\text{ex}} \cup
\{ (\text{NatComp}, s_{\text{ex}}, \text{Org}), (\text{hiredBy}, s_{\text{ex}}, \text{Person}), (\text{hiredBy}, s_{\text{ex}}, r, \text{Org}), (\text{CEOOf}, s_{\text{ex}}, \text{Person}), (\text{CEOOf}, s_{\text{ex}}, r, \text{Org}), (p_1, \text{worksFor}, _b_c), (_b_c, r, \text{Comp}), (p_2, \text{worksFor}, a), (a, r, \text{Org})\}
\]

\[
\text{(Gex)}_2 = (\text{Gex)}_1 \cup
\{ (p_1, r, \text{Person}), (p_2, r, \text{Person}), (_b_c, r, \text{Org})\}
\]

2.3 Basic Graph Pattern Queries

A popular RDF query dialect consists of basic graph pattern queries, or BGPOs, in short. Let \( Y \) be a set of variable symbols, disjoint from \( \mathcal{I} \cup \mathcal{R} \cup \mathcal{L} \). A basic graph pattern (BGP) is a set of triple patterns (triples in short) belonging to \( (\mathcal{I} \cup \mathcal{R} \cup \mathcal{L}) \times (\mathcal{I} \cup \mathcal{R} \cup \mathcal{L}) \). For a BGP \( P \), we denote by \( \text{Var}(P) \) the set of variables occurring in \( P \), by \( \text{Bl}(P) \) its set of blank nodes, and by \( \text{Val}(P) \) its set of values (IRIs, blank nodes, literals and variables).

Definition 2.5 (BGP Queries). A BGP query \( q \) is of the form \( q(x) \leftarrow P \), where \( P \) is a BGP (also denoted by \( \text{body}(q) \)), and \( x \subseteq \text{Var}(P) \) are the answer variables of \( q \).

To ease the presentation, and without loss of generality, we consider BGPOs without blank nodes, as it is well-known that these can be replaced by non-answer variables [46].

For query answering based on query reformulation (see Section 2.4), it is convenient to slightly generalize BGPOs into partially instantiated BGPOs [12, 29]. Starting from a BGPO \( q \), partial instantiation may replace some variables with values from \( \mathcal{I} \cup \mathcal{R} \cup \mathcal{L} \), as specified by a substitution \( \sigma \). Due to \( \sigma \), and in contrast with standard BGPOs, some answer variables of the resulting query \( q_\sigma \) can be bound:

Example 2.6 (Partially instantiated BGPO). Consider the BGPO asking for who is working for which kind of company \( q(x, y) \leftarrow (x, \text{worksFor}, z), (z, r, y), (y, t_{\text{ex}}, \text{Comp}) \) and the substitution \( \sigma = (x \mapsto p_1) \). The corresponding partially instantiated BGPO is: \( q(p_1, y) \leftarrow (p_1, \text{worksFor}, z), (z, r, y), (y, t_{\text{ex}}, \text{Comp}) \). In it, the first answer variable has been bound to \( p_1 \).

For simplicity, below we use the term "query" to designate either a standard BGPO or a partially instantiated BGPO.

The semantics of a BGPO on an RDF graph is defined through standard homomorphisms from the query body to the queried graph. We recall that a homomorphism from a BGP \( P \) to an RDF graph \( G \) is a function \( \phi \) from \( \text{Val}(P) \) to \( \text{Val}(G) \) such that for any triple \( (s, p, o) \in P \), the triple \( (\phi(s), \phi(p), \phi(o)) \) is in \( G \), with \( \phi \) the identity on IRIs and literals. Next, we distinguish query evaluation, whose result is just based on the explicit triples of the graph, i.e., on BGP-to-RDF graph homomorphisms, from query answering that also accounts for the implicit graph triples, resulting from entailment. Formally:

Definition 2.7 (Evaluation and answering). The answer set to a BGPO \( q \) on an RDF graph \( G \) w.r.t. a set \( \mathcal{R} \) of RDF entailment rules is: \( q(G, \mathcal{R}) = \{ x \mid \phi \text{ homomorphism from body}(q) \text{ to } C_{\mathcal{R}} G \} \). If \( \mathcal{R} = \emptyset \) and true when \( q(G, \emptyset) = \{ \} \), i.e., the answer to \( q \) is the empty tuple.

The evaluation of \( q \) on \( G \), denoted \( q(G, \emptyset) \) or \( q(G) \) in short, is obtained from homomorphisms from body(\( q \)) to \( G \) alone (not \( G^\mathcal{R} \)). It can be seen as a particular case of query answering when \( \mathcal{R} = \emptyset \).

Example 2.8 (Evaluation and answering). Consider again the BGPO \( q \) from the preceding example. Its evaluation on \( G_{\text{ex}} \) is empty because \( G_{\text{ex}} \) has no explicit \( \text{worksFor} \) assertion, while its answer set on \( G_{\text{ex}} \) w.r.t. \( \mathcal{R} \) is \( \{ (p_1, \text{NatComp}) \} \) because \( p_1 \) being CEO of \( _b_c \), \( p_1 \) implicitly works for it, and \( _b_c \) is explicitly a company of the particular type \( \text{NatComp} \).

The above notions and notations naturally extend to unions of (partially instantiated) BGPOs, or UBPGOs in short.

We end this section by pointing out that many RDF data management systems use saturation-based query answering, which directly follows the definition of query answering: an RDF graph \( G \) is first saturated with the set \( \mathcal{R} \) of entailment rules, so that the
answer set to an incoming query \( q \) is obtained through query evaluation as \( q(\Omega^R) \).

2.4 Query Reformulation

Reformulation-based query answering is an alternative technique to the widely adopted saturation-based query answering. It consists in reformulating a query using \( \mathcal{R} \), so that evaluating the reformulated query on \( G \) yields the answer set to the original query on \( G \) w.r.t. \( \mathcal{R} \). Intuitively, reformulation injects the ontological knowledge into the query, just as saturation injects it into the RDF graph. We rely here on the very recent algorithm from [12], which takes into account all the entailment rules from Table 3. The process is decomposed into two steps according to the partition of \( \mathcal{R} \) into \( \mathcal{R}_u \) and \( \mathcal{R}_c \).

(i) The first step reformulates a BGPO \( q \) w.r.t. an RDFS ontology \( O \) and the set of rules \( \mathcal{R}_u \) into a BGPO, say \( Q_c \), which is guaranteed to contain ontology triples. Intuitively, this step generates new BGPOs obtained from \( q \) by instantiating variables that query the ontology with all their possible bindings; for instance, \( y \) in a query triple \((y, \prec, \text{Comp})\) is bound to the IRIs of all explicit and implicit subclasses of \text{Comp} in \( O \). This step, alone, is sound and complete w.r.t. \( \mathcal{R}_c \) for query answering, i.e., for any graph \( G \) with ontology \( O \), \( q(G, \mathcal{R}_c) = Q_c(G) \).

Example 2.9 (Two-step reformulation). Consider the query \( q(x, y) \leftarrow (x, \text{worksFor}, z), (z, \tau, y), (y, \prec, \text{Comp}) \) from the preceding example and the ontology \( O \) in Example 2.2. The first reformulation step instantiates the triple \((y, \prec, \text{Comp})\) on \( O \), leading to:

\[
Q_c = q(x, \text{NatComp}) \leftarrow (x, \text{worksFor}, z), (z, \tau, \text{NatComp}).
\]

Then, \( Q_c \) is reformulated into \( Q_c.a = q(x, \text{NatComp}) \leftarrow (x, \text{worksFor}, z), (z, \tau, \text{NatComp}) \cup q(x, \text{NatComp}) \leftarrow (x, \text{hiredby}, z), (z, \tau, \text{NatComp}) \cup q(x, \text{NatComp}) \leftarrow (x, \text{ceoOf}, z), (z, \tau, \text{NatComp}) \) by specializing \text{worksFor} according to its subproperties in \( O \). It can be checked that \( Q_{c, a} = (G_{c, a}) = q(G_{c, a}, \mathcal{R}) = q(G_{c, a}) \) obtained here from the third BGPO in \( Q_{c, a} \).

2.5 Query Rewriting-based Data Integration

We recall now the basics of relational view-based query rewriting (Section 2.5.1), which has been extensively studied [23, 31]. Then we present a generalization of the notion of views as mappings [35] (Section 2.5.2).

2.5.1 View-based (LAV) Data Integration. An integration system \( \mathcal{I} \) is made of a global schema \( S \) (a set of relations) and a set \( \mathcal{V} \) of views. An instance of \( \mathcal{I} \) assigns a set of tuples to each relation of \( S \) and to each view of \( \mathcal{V} \). The data stored in a view is called its extension. Further, to each view \( V \) is associated a query \( V(q) \leftarrow \psi(q) \) over the global schema \( S \), specifying how its data fits into \( S \). Accordingly, this framework is called local-as-view (LAV) data integration. For instance, let \( S \) consist of three relations \text{Emp}(\text{id}, \text{name}, \text{did}), \text{Dept}(\text{id}, \text{did}, \text{country}), \text{Salary}(\text{id}, \text{amount}) \), where \( \text{id}, \text{did} \) and \( \text{did} \) are respectively identifiers for employees, departments and companies.

Consider the views \( V_1(\text{id}, \text{name}, \text{country}) \leftarrow \text{Emp}(\text{id}, \text{name}, \text{did}) \), \( \text{Dept}(\text{id}, \text{"IBM"}, \text{country}) \) providing the names of IBM employees and where they work, and \( V_2(\text{id}, \text{amount}) \leftarrow \text{Emp}(\text{id}, \text{name}, \text{"R&D"}) \), \( \text{Salary}(\text{id}, \text{amount}) \), which indicates the salaries of employees in R&D departments. Typically, no single view is expected to bring all information of a given kind; for instance, \( V_1 \) brings some IBM employees, but other views may bring others, e.g., \( V_2 \), possibly overlapping with \( V_1 \); this is called the “Open World Assumption” (OWA).

In an OWA setting, we are interested in certain answers [31], i.e., those that are sure to be part of the query result, knowing the data present in the views. Such answers can be computed by rewriting a query over \( S \), into one over the views \( V \), evaluating the rewriting over the view extensions produces the answers.

Ideally, a rewriting should be equivalent to the query over \( S \), i.e., compute exactly the same answers. However, depending on the views and queries, such a rewriting may not exist. For instance, the query \( q(e, a) \leftarrow \text{Emp}(e, n, d), \text{Dept}(d, c, "France") \), \( \text{Salary}(e, a) \) does not have an equivalent rewriting using \( V_1 \) and \( V_2 \), because \( V_1 \) only provides IBM employees working in France, while \( V_2 \) only has salaries of employees of R&D departments. A maximally contained rewriting brings all the query answers that can be obtained through the given set of views; the rewriting may be not equivalent to \( q \) (but just contained in \( q \)). In our example, \( q(e, a) \leftarrow V_1(e, n, "France"), V_2(e, a) \) is a maximally contained rewriting of \( q \); it returns employees of French IBM R&D departments with their salary, clearly a subset of \( q \) answers.

A remarkable result holds for (unions of) conjunctive queries (UCQs), conjunctive views (views \( V \) such that the associated query \( V(q) \leftarrow \psi(q) \) is a CQ) and rewritings that are UCQs: any maximally contained rewriting computes exactly the certain answers [2]; we will build upon this result for answering queries in our RDF integration systems.

2.5.2 GLAV Data Integration. The above setting has been generalized to views that are not necessarily stored as such, but just queries over some underlying data source. For instance, assuming a data source \( D \) holds the relations \text{Person}(\text{id}, \text{name}) \) and \text{Contract}(\text{id}, \text{did}, \text{country}) \) (see Figure 1) with people and their work contracts at IBM, the view \( V_1 \) from the above example may be defined on \( D \) by the query \( V^D_1 \) over the \( D \) schema shown in the figure (note that \( V^D_1 \) hides the department from system \( I \)), \( V^D_1 \) provides the extension of \( V_1 \). Similarly, view \( V_2 \) may be defined as a query over some data source (or sources).

Figure 1: Example: view \( V_1 \) as a GLAV mapping.
Historically, two restrictions of GLAV mappings have been investigated. First, global-as-view (or GAV) mappings define global schema relations as views over the local schemas. Specifically, a GAV mapping \( q_1(x) \sim q_2(x) \), \( q_2 \) defines a single element of the global schema (hence body(q2) is restricted to a single atom if \( q_2 \) is a CQ, or a single triple pattern if \( q_2 \) is a BGPO and \( q_1 \) is the set of values occurring in \( E \). We can now define the RIS data triples induced by \( q_2 \) and an extent \( E \) of \( M \), the RIS data triples induced by \( M \) and \( E \) form an RDF graph defined as follows:

\[
G_E^M = \bigcup_{m=q_1(x) \sim q_2(x): m \in M} \{\text{bgp2rdf(body(q2)[x→i])} | \text{V}_m(i) \in E\}
\]

where

- body(q2)[x→i] is the BGP body(q2) in which the answer variables \( x \) are bound to the values in the tuple \( \text{V}_m(i) \), part of \( E \),
- bgp2rdf(\( \cdot \)) is a function that transforms a BGP into an RDF graph, by replacing each variable with a fresh blank node.

Observe that, because we use GLAV mappings, RIS data triples may include fresh blank nodes, as exemplified below; these correspond to the existential variables allowed in GLAV mappings, as discussed at the end of Section 2.5.2.

Example 3.4. Reusing the mappings from Example 3.2, let \( M = \{m_1, m_2\} \) and its extent \( E = \{\text{V}_m(p_1), \text{V}_m(p_2, a)\} \). The RIS data triples they lead to are:

\[
G_E^M = \{(p_1, \text{ceoOf}, _{-b}_c), (_{-b}_c, \tau, \text{NatComp}), (p_2, \text{hiredBy}, a), (c, a, \tau, \text{PubAdmin})\}
\]

In particular, the first and second triples contain the blank node \( _{-b}_c \), introduced by bgp2rdf instead of the variable \( y \) in the head (query \( q_2 \)) of \( m_1 \). Importantly, only non-answer variables in a mapping head lead to blank nodes introduced this way: by Def. 3.3, answer variables (here \( x \) for \( m_1 \) and \( x, y \) for \( m_2 \)) are replaced with values from \( \text{V}_m(i) \), thus from Val(E).

Finally, we define a RIS as a tuple \( S = (O, R, M, E) \) stating that \( S \) allows to access (query), with the reasoning power given by the set \( R \) of RDF entailment rules, the RDF graph comprising the ontology \( O \) and the data triples induced by the set of mappings \( M \) and their extent \( E \).

### 3.2 Query answering problem

The problem we consider is answering BGPOs in a RIS. We define certain answers in a RIS setting as follows:

**Definition 3.5 (Certain answer set).** The certain answer set of a BGPO \( q \) on a RIS \( S = (O, R, M, E) \) is:

\[
\text{cert(q, S)} = \{\phi(x) | \phi \text{ homomorphism from body(q) to (O\cup G^M_E)} \}
\]

where \( \phi(x) \) comprises only values from Val(E).

The certain answer set \( \text{cert(q, S)} \) is thus the subset of \( g(O \cup G^M_E, R) \) restricted to tuples fully built from source values, i.e., we exclude tuples with blank nodes introduced by the mappings (see Def. 3.3). Note, however, that blank nodes can be exploited to answer queries, as shown below.

**Example 3.6 (Certain answers).** Consider the RIS S made of the ontology \( O \) of \( G_{ex} \) in Example 2.2, the set \( R \) of entailment rules shown in Table 3, and the set of mappings \( M \) together with the extent \( E \) from Example 3.4.

Let \( g(q, y) \leftrightarrow (x, \text{worksFor}, y), (y, r, \text{Comp}) \) be the query asking "who works for which company", while the query \( q'(x) \leftrightarrow (x, \text{worksFor}, y), (y, r, \text{Comp}) \) asks "who works for some company". The only difference between them is that \( y \) is an answer variable in \( q \) and not in \( q' \). The certain answer set of \( q \) is \( \emptyset \)
which are entailed from: while the certain answer set of
The problem we study in the next section is:
The query
4 QUERY ANSWERING IN A RIS
Since we adopt a mediator-style approach, the RIS data triples
when evaluated on the RIS data triples (recall Section 2.4), provided that answers with blank nodes introduced by
these data triples are not materialized, the RDF query evaluation problem is in turn reduced to relational view-based query answering, by rewriting \( Q_{ca} \) using the RIS GLAV mappings \( M \) seen as LAV views (step (2)). This produces a relational rewriting \( q_{\text{rel}} \) over the mappings extension (step (3)), whose evaluation with a mediator query engine provides the desired certain answers (steps (4) and (5)).

Some reasoning at query time
The second strategy (detailed in Section 4.2) is a main contribution of this paper. First, it reduces the RIS query answering problem to saturation-based query answering by reformulating (step (1')) the query \( q \) based on \( O \) and \( R \) only (not \( R = R_e \cup R_a \) as above). The obtained reformulation \( Q_e \) thus yields the expected certain answer set when evaluated on the RIS data triples saturated with \( R_e \) (recall Section 2.4), again provided that the answers with blank nodes introduced by the \( \text{bgp2rdf} \) function are discarded (as above). Since these triples are not materialized in a RIS, hence cannot be saturated with \( R_e \), the saturation-based query answering problem is in turn reduced to relational view-based query answering, by rewriting \( Q_e \) using the RIS GLAV mappings saturated \( O \) and \( R_e \), seen as LAV views. These saturated mappings, denoted \( M^{a,O} \), are obtained (step (A)) from the original ones by adding to their head queries (\( q_2 \)) all the explicit and implicit RIS data triples they model w.r.t. \( O \) and \( R_e \). Then, the partially reformulated query \( Q_e \) is rewritten using \( M^{a,O} \) (step (2')) and the resulting query (step (3)) is evaluated as in the first strategy (steps (4) and (5)). Importantly, mappings are saturated offline, and need to be updated only when some mapping changes. This limits both the reasoning effort at query time and the complexity of the reformulated query to rewrite, hence the rewriting time needed to obtain a rewriting \( q_{\text{rel}} \) over the data sources, as our experiments show (Section 5).

No reasoning at query time
Finally, the third strategy (detailed in Section 4.3) reduces the RIS query answering problem directly to view-based query answering. Here, the mappings are saturated offline as above (step (A)), in order to model all explicit and implicit RIS data triples. Also, these mappings are complemented with a set of mappings, noted \( M^{c,O} \) (step (B)), comprising all the explicit and implicit RIS schema triples w.r.t. \( O \) and \( R_e \); since only \( R_e \) rules entail new schema triples (Table 3), \( O^{c,O} \) is actually equal to \( O^{R_e} \). This second set of mappings is also computed offline, and only needs to be updated when the ontology changes. A query \( q \) just needs to be rewritten based on the above mappings \( M^{a,O} \cup M^{c,O} \), seen as LAV views (step (2’)), in order to obtain, as above, a rewriting \( q_{\text{rel}} \) over the data sources (step (3’), followed by the evaluation steps (4’) and (5’)).

Before going into the technical details of the above strategies, we introduce a set of simple functions. The \( \text{bgp2ca} \) function transforms a BGP into a conjunction of atoms with ternary predicates \( T \) (standing for "triples") as follows: \( \text{bgp2ca}(\{(s_1, p_1, o_1), \ldots, (s_n, p_n, o_n)\}) = T(s_1, p_1, o_1) \land \cdots \land T(s_n, p_n, o_n) \). The \( \text{bgpq2cq} \) function transforms a BGPQ \( q(x) \leftarrow \text{body}(q) \) into a CQ \( q(x) \leftarrow \text{bgpq2cq}body(q)) \). Finally, the function \( \text{ubgpq2ucq} \) function transforms a UBGPQ \( \bigcup_{i=1}^{n} q_i(x_i) \) into a UCQ by applying the above \( \text{bgpq2cq} \) function to each of its \( q_i \).

4.1 Rewriting Fully-Reformulated Queries using Mappings as Views: \( \text{REW-CA} \)
Based on [12], the first step of this strategy, (1) in Figure 2, reformulates a query \( q \) w.r.t. \( O \) and \( R = R_e \cup R_a \) into a query \( Q_{ca} \). This allows obtaining the certain answers directly from the RIS data triples, and not from their saturation after they have been
augmented with $O$ (recall Definition 3.5). Indeed, the correctness of the reformulation ensures that the certain answers of $q$ on the RIS $S$ correspond precisely to those of $Q_{c,a}$ asked on $S$ when disregarding $O$ and $R$, as formally expressed in the next lemma. Of course, this still does not provide a concrete solution to obtain the desired certain answers using standard query evaluation, since the RIS data triples $G^M$ are not materialized.

**Lemma 4.1.** Let $S = (O, R, M, E)$ be a RIS, $q$ be a BGPQ and $Q_{c,a}$ its UBGQ reformulation w.r.t. $O, R = \mathcal{R}_c \cup \mathcal{R}_a$ using [12]. Then:

$$
cert(q, S) = cert(Q_{c,a}, (\emptyset, \emptyset, M, E))
$$

The proof of this and our following claims can be found in [13]. Recall that the RIS data triples are defined from the mappings $M$ by, for every mapping $m = q_1(x) \leadsto q_2(x) \in M$, (i) evaluating the mapping body $q_1(x)$ on the data source to produce its extension $\text{ext}(m) \in \mathcal{E}$, and then (ii) instantiating the mapping head $q_2(x)$ with its extension. At the same time, this is also how the instance of a data integration system based on LAV views and their extensions is defined in a relational setting (Section 2.5.1).

Based on this analogy, we recast the RIS query answering problem of the above Lemma, into a relational view-based query answering one. To this aim, we treat our mappings as LAV views:

**Definition 4.2 (Mappings as relational LAV views).** Let $m = \{q_1(x) \leadsto q_2(x)\}$ be a mapping. Its corresponding relational LAV view is: $V_m(x) \leftarrow bgp2ca(body(q_2))$.

**Example 4.3.** The relational LAV views corresponding to the mappings $m_1, m_2$ from Example 3.2 are:

- $V_{m_1}(x) \leftarrow T(x, \text{ceoOf}, y), T(y, r, \text{NatComp})$
- $V_{m_2}(y) \leftarrow T(x, \text{hiredBy}, y), (T(y, r, \text{PubAdmin})$

We denote the set of views derived from all the mappings $M$ by Views($M$). Crucially, the extent $\mathcal{E}$ of the mapping set $M$ is also an extent for the corresponding set of views Views($M$).

Based on the above Lemma 4.1, treating mappings and their extent as relational LAV views and their extent, and seeing UBGQs as (U)CQs with the help of the functions introduced in the beginning of Section 4, we reduce the RIS query answering problem to view-based query answering:

**Theorem 4.4 (REW-CA correctness).** Let $S = (O, R, M, E)$ be a RIS and $q$ be a BGPQ. Let $Q_{c,a}$ be the reformulation of $q$ w.r.t. $O$ and $R$ using [12]. Then:

$$
cert(q, S) = cert(\text{ubgq2acq}(Q_{c,a}), \text{Views}(M), E)
$$

where $cert(\text{ubgq2acq}(Q_{c,a}), \text{Views}(M), E)$ denotes the certain answer set of $\text{ubgq2acq}(Q_{c,a})$ over Views($M$) and $E$.

Importantly, this provides an effective solution to RIS query answering problem by using state-of-the-art view-based query rewriting techniques [31], in particular for step (2) in Figure 2.

**Example 4.5 (REW-CA query answering).** Consider again the RIS in Example 3.6 and the query $q(x, y) \leftarrow (x, y, z), (z, r, t), (y, \text{sp}, \text{worksFor}), (t, \text{ceo}, \text{Comp}), (x, \text{worksFor}, a), (a, r, \text{PubAdmin})$ asking "who works for some public administration, and what working relationship he/she has with some company". Its UBGQ reformulation, seen as a UQ, is shown in Figure 3. Its maximally-contained rewriting based on the views obtained from the RIS mappings is: $q_1(x, \text{ceoOf}) \leftarrow V_m(x), V_m(x, y)$ obtained from the second CQ in the above union. This becomes clear when the views are replaced by their bodies: $q_1(x, \text{ceoOf}) \leftarrow T(x, \text{ceoOf}, y), T(y, r, \text{NatComp}), T(x, \text{hiredBy}, y), T(y, r, \text{PubAdmin})$. Note

$$Q_{c,a} = q(x, \text{ceoOf}) \leftarrow T(x, \text{ceoOf}, z), T(z, r, \text{NatComp}), T(x, \text{worksFor}, a), (a, r, \text{PubAdmin})$$

$$\cup q(x, \text{ceoOf}) \leftarrow T(x, \text{ceoOf}, z), T(z, r, \text{NatComp})$$

$$\cup q(x, \text{ceoOf}) \leftarrow T(x, \text{hiredBy}, a), (a, r, \text{PubAdmin})$$

$$\cup q(x, \text{hiredBy}) \leftarrow T(x, \text{hiredBy}, z), T(z, r, \text{NatComp})$$

$$\cup q(x, \text{hiredBy}) \leftarrow T(x, \text{hiredBy}, z), T(z, r, \text{NatComp})$$

$$\cup q(x, \text{hiredBy}) \leftarrow T(x, \text{hiredBy}, z), T(z, r, \text{NatComp})$$

$$\cup q(x, \text{hiredBy}) \leftarrow T(x, \text{hiredBy}, z), T(z, r, \text{NatComp})$$

Using rewritings for the RIS and $Q_{c,a}$, the query $Q_{c,a}$ is rewritten to the following form (2) in Figure 2.

**Figure 3: Sample reformulation for Example 4.5.**

that the other CQs cannot be rewritten given the available views.

With the current RIS, this rewriting yields an empty certain answer set to $q$, i.e., $cert(q, S) = \emptyset$, because the extent of the mappings, hence of the views, is: $\mathcal{E} = \{V_{m_1}(p_1), V_{m_2}(p_2, a)\}$. However, if we add $V_{m_3}(p_1, a)$ to $E$, then $cert(q, S) = \{\{p_1, \text{ceoOf}\}\}$.

4.2 **Rewriting Partially-Reformulated Queries using Saturated Mappings as Views: REW-C**

In contrast with the REW-CA strategy that performs all the reasoning w.r.t. $O$ and $R = \mathcal{R}_c \cup \mathcal{R}_a$ at query time, our second strategy called REW-C splits the reasoning work between offline preprocessing and query time.

The first step of this strategy, labeled (1) in Figure 2, reformulates a query $q$ using [12], but solely w.r.t. $O$. $\mathcal{R}_c$, producing a UBGQ denoted $Q_c$. From the correctness of this reformulation step, and the fact that only $\mathcal{R}_a$ needs to be considered to answer $Q_c$, with respect to the entire set of rules $R$ (recall Section 2.4), the certain answer set of $q$ asked on the RIS $S$ is exactly the certain answer set of $Q_c$ asked on $S$ when disregarding $\mathcal{R}_c$.

**Lemma 4.6.** Let $S = (O, R, M, E)$ be a RIS, $q$ be a BGPQ and $Q_c$ its reformulation w.r.t. $O, \mathcal{R}_c$ [12]. Then:

$$cert(q, S) = cert(Q_c, (O, \mathcal{R}_c, M, E))$$

In other words, the desired answer set could be obtained by evaluating $Q_c$ on the RIS data triples $G^M$ saturated by $\mathcal{R}_c$. Again, since the RIS data triples are not materialized, this does not provide a concrete solution. To account for the impact of the ontology $O$ and the entailment rules $R$ on these "virtual" data triples, we rely on $\text{BGQ saturation}$ [25]: given a BGPQ $q$, $O$ and $R$, the saturation $q^{R,O}$ is augmented with all the triples $q$ implicitly asks for, given the ontology $O$ and the rules $R$. BGQ saturation is exemplified below:

**Example 4.7 (BGQ saturation).** Consider the ontology $O$ of $G_a$ and the query $q(x) \leftarrow \langle x, \text{hiredBy}, y \rangle, (y, r, \text{NatComp})$ asking who has been hired by a national company. Its saturation w.r.t. $\mathcal{R}_a, O$ is:

$$q^{\mathcal{R}_a,O}(x) \leftarrow \text{body}(q), (x, \text{worksFor}, y), (x, r, \text{Person}), (y, r, \text{Comp}), (y, r, \text{Org})$$

We use BGQ saturation to saturate the RIS mapping heads w.r.t. $\mathcal{R}_a, O$, so that the saturated mappings together with $E$ model the saturated RIS data triples w.r.t. $\mathcal{R}_a, O$. To compute $q^{\mathcal{R}_a,O}$ we (1) saturate body($q$) $\cup$ $O$ using $\mathcal{R}_a$, then (2) add to the body of $q$ all triples thus inferred.

**Definition 4.8 (Mappings saturation).** The saturation of a set $M$ of RIS mappings w.r.t. entailment rules $\mathcal{R}_a$ and ontology $O$ is:

$$M^{a,O} = \bigcup_{m \in M} \{q_1(\bar{x}) \leadsto q_2^{R_a,O}(\bar{x}) \mid m = q_1(\bar{x}) \leadsto q_2(\bar{x})\}$$
We saturate mappings offline, and just need to update them when \( O \) or the mapping heads change.

**Example 4.9 (Saturated mappings).** Consider the RIS of Example 3.6, the mapping heads in \( M^{a,O} \) are (added implicit tripels are in blue):

\[ m_1 : q_2^{RIS}(x) \leftarrow (x, \text{ceoOf}, y), (y, r, \text{NatComp}) \]
\[ m_2 : q_2^{RIS}(x, y) \leftarrow (x, \text{hiredBy}, y), (y, r, \text{PubAdmin}) \]

From the above Lemma and the use of saturated RIS mappings instead of the original ones, we show:

**Lemma 4.10.** Let \( S = (O, R, M, E) \) be a RIS, \( q \) be a BGPQ and \( Q_c \) its reformulation w.r.t. \( O, R_c \) [12]. Then:

\[ \text{cert}(q, S) = \text{cert}(Q_c, (\emptyset, \emptyset, M^{a,O}, E)) \]

This result allows solving the RIS query answering problem by relational view-based query rewriting (step 2' in Figure 2):

**Theorem 4.11 (REW-C correctness).** Let \( S = (O, R, M, E) \) be a RIS, \( q \) be a BGPQ and \( Q_c \) its reformulation w.r.t. \( O, R_c \). Then:

\[ \text{cert}(q, S) = \text{cert}(ubqp2ucq(Q_c), \text{Views}(M^{a,O}, E)) \]

**Example 4.12 (REW-CA).** Consider again the RIS in Example 3.6 and the query \( q \) of Example 4.5. Its reformulation \( Q_c \) w.r.t. \( O, R_c \), seen as a UCQ is:

\[ q(x, \text{ceoOf}) \leftarrow T(x, \text{ceoOf}, z), (z, r, \text{NatComp}), \]
\[ (x, \text{worksFor}, a), T(z, \text{hiredBy}, z), (z, r, \text{NatComp}), \]
\[ (x, \text{worksFor}, a), (a, r, \text{PubAdmin}) \]

This reformulation is therefore rewritten using the RIS views as:

\[ q_f(x, \text{ceoOf}) \leftarrow V_m(x), V_m(x, y) \]

It is obtained from the first CQ in the above; the second one has no rewriting based on the available RIS views. We remark that this rewriting is equivalent to the one obtained in Example 4.5, hence yields the same answers.

### 4.3 Rewriting Queries using Saturated Mappings and Ontology Mappings as Views: rew

This strategy does not reason at query time at all. Instead, it rewrites a query \( q \) based on the saturated RIS mappings \( M^{a,O} \) as above, and on a specific set of ontology mappings we build to model the saturated RIS ontology as a data source:

**Definition 4.13 (Ontology mappings).** The set of ontology mappings for a RIS ontology \( O \) is:

\[ M_{O^{\text{C}}} = \bigcup_{c \in \{\text{ceoOf}, \text{worksFor} \}} \{m_x \mid m_x = q_2(s, o) \rightsquigarrow q_2(s, o) \} \]

with \( q_2(s, o) \leftarrow (s, o) \). The extension of an ontology mapping \( m_x \) is \( \text{ext}(m_x) = \{V_m(s, o) \mid (s, o) \in O^{\text{C}}\} \). The extent of \( M_{O^{\text{C}}} \) is denoted \( E_{O^{\text{C}}} \).

We compute ontology mappings offline, and only need to update them when the ontology changes. The ontology mapping extensions \( E_{O^{\text{C}}} \) store all the explicit and implicit RIS ontology truples (recall from Section 2.2 that only \( R_k \) lead to such truples).

Importantly, this leads to the observation that a query triple that refers to the ontology (schema) can be evaluated on the ontology mapping extensions alone. Formally:

\[ q(x, \text{ceoOf}) \leftarrow V_m(x), V_m(x, \text{ceoOf}, \text{worksFor}) \]
\[ \cup q(x, \text{ceoOf}) \leftarrow V_m(x), V_m(x, \text{ceoOf}, \text{NatComp}) \]
\[ \cup q(x, \text{worksFor}) \leftarrow V_m(x), V_m(x, \text{worksFor}, \text{worksFor}) \]
\[ \cup q(x, \text{worksFor}) \leftarrow V_m(x), V_m(x, \text{worksFor}, \text{NatComp}) \]
\[ \cup q(x, \text{worksFor}) \leftarrow V_m(x), V_m(x, \text{worksFor}, \text{PubAdmin}) \]
\[ \cup q(x, \text{worksFor}) \leftarrow V_m(x), V_m(x, \text{worksFor}, \text{PubAdmin}) \]

**Figure 4:** Sample rewriting for Example 4.17.

**Lemma 4.14.** Let \( S = (O, R, M, E) \) be a RIS and \( q \) be a BGPQ. Then:

\[ \text{cert}(q, S) = \text{cert}(q, (O, R_c, M_{O^{\text{C}}} \cup M, E_{O^{\text{C}}} \cup E)) \]

This lemma effectively "pushes" \( \text{REM} \) reasoning in the set of mappings (to which we add \( M_{O^{\text{C}}} \)) and the extent (to which we add \( E_{O^{\text{C}}} \)). Next, we rely (as we did for REW-CA) on mappings saturation with \( O, R_2 \) to also push \( R_2 \) reasoning in the mappings, leading to:

**Lemma 4.15.** Let \( S = (O, R, M, E) \) be a RIS and \( q \) be a BGPQ. Then:

\[ \text{cert}(q, S) = \text{cert}(q, (\emptyset, \emptyset, M_{O^{\text{C}}} \cup M^{a,O}, E_{O^{\text{C}}} \cup E)) \]

This allows to reduce RIS query answering to relational view-based query rewriting (step 2' in Figure 2):

**Theorem 4.16 (REW correctness).** Let \( S = (O, R, M, E) \) be a RIS and \( q \) be a BGPQ. Then:

\[ \text{cert}(q, S) = \text{cert}(ubqp2ucq(q), \text{Views}(M_{O^{\text{C}}} \cup M^{a,O}, E_{O^{\text{C}}} \cup E)) \]

**Example 4.17 (REW).** Consider again the RIS in Example 3.6 and the query \( q \) of Example 4.5 seen as a CQ:

\[ q(x, y) \leftarrow T(x, y, z), T(z, r, t), T(y, \text{sp}, \text{worksFor}) \]
\[ T(t, \text{sp}, \text{Comp}), T(x, \text{worksFor}, a), \]
\[ T(a, \text{r}, \text{PubAdmin}) \]

Its maximally-contained rewriting \( q_{\text{rew}} \) based on the views obtained from the RIS saturated mappings and ontology mappings appears in Figure 4. This rewriting is much larger than the ones of the two preceding techniques: this is due to the ontology mappings. If we assume that \( E \) also contains \( V_m(s, (p_1, a)) \), as we did in
We now describe our experiments with RIS query answering. The rewritings produced by Graal are unfolded into queries on the based mediator (or polystore) system, capable both of pushing, we use Tatooine [4, 10], a Java-heterogeneous data sources also provides the RDF query reformulation algorithm [12].

In addition to our strategies based on query rewriting, we in-clude in our comparison a simple alternative strategy, based in short) Then, MAT query answering amounts to query eval-uation on the saturated materialization. Therefore, MAT query answering can be seen as a lower bound for query answering through other strategies.

5 EXPERIMENTAL EVALUATION
We now describe our experiments with RIS query answering. In addition to our strategies based on query rewriting, we in-clude in our comparison a simple alternative strategy, based on materialization and denoted MAT. Offline (before answering queries), this strategy materializes the RIS data triples and saturates them with the rule set R. The materialization is stored and saturated in an RDF data management system (RDFDB, in short). Then, MAT query answering answers to query eval-uation on the saturated materialization. Therefore, MAT query answering can be seen as a lower bound for query answering through other strategies.

5.1 Experimental settings

Software Our platform is developed in Java 1.8, as follows. Our RDFDB is OntoSQL2, a Java platform providing efficient RDF storage, saturation, and query evaluation on top of an RDBMS [14, 29], relying on Postgres v9.6. To save space, OntoSQL encodes IRIs and literals into integers, and a dictionary table which allows going from one to the other. It stores all resources of a certain type in a one-attribute table, and all (subject, object) pairs for each property (including RDFs schema properties) in a table; the tables are indexed. OntoSQL is used in the MAT strategy, and it also provides the RDF query reformulation algorithm [12].

We rely on the Graal engine [9] for view-based query re-writ-ting. Graal is a Java toolkit dedicated to query answering algo-rithms in knowledge bases with existential rules (a.k.a. tuple-generating dependencies). Since the relational view \( v_m(x) \leftarrow bp2ca\text{body}(q_2) \) corresponding to a GLAV mapping m (recall Def. 4.2) can be seen as a specific existential rule of the form \( v_m(x) \rightarrow bp2ca\text{body}(y_2) \)), the query reformulation algorithm of Graal can be used to rewrite the UCQ translation of a BGPOQ with respect to a set of RIS mappings. To execute queries against heterogeneous data sources, we use Tatooine [4, 10], a Java-based mediator (or polystore) system, capable both of pushing queries in underlying data sources and (unlike other polystores, e.g., [24]) of evaluating joins within the mediator engine. Query rewritings produced by Graal are unfolded into queries on the data sources (using the q2_i parts of the mappings, see Section 2.5.2) and passed to Tatooine. We implemented the RIS query answering methods described here in Java 1.8 on top of these tools.

Hardware We used servers with 2.7 GHz Intel Core i7 processors and 160 GB of RAM, running CentOs Linux 7.5.

5.2 Experimental scenarios

RDF Integration Systems (RIS) used Our first interest was to study scalability of RIS query answering, in particular in the relational setting studied in many prior works. To achieve this, we used the BSBM benchmark relational data generator3 to build databases consisting of 10 relations named producer, product, offer, review etc. Using two different benchmark scale factors, we obtained a data source DS1 of 154.054 tuples across the relations, respectively, DS2 of 7.843.660 tuples; both are stored in Postgres. We used two RDFs ontologies O1 respectively O2, containing, first, subclass hierarchies of 151 (resp. 201) product types, which come with DS1 respectively DS2. To O1 and O2, we add a natural RDFs ontology for BSBM composed of 26 classes and 36 properties, used in 40 subclass, 32 subproperty, 42 domain and 16 range statements.

Relational-sources RIS We devised two sets M1, M2 of 307, respectively. 3863 mappings, which expose the relational data from DS1, respectively DS2 as RDF graphs. The relatively high number of mappings is because: (i) each product type (of which there are many, and their number scales up with the BSBM data size) appears in the head of a mapping, enabling fine-grained and high-coverage exposure of the data in the integration graph; (ii) we also generated more complex GLAV mappings, partially exposing the results of join queries over the BSBM data; interestingly, these mappings expose incomplete knowledge, in the style of Example 3.4. The mapping sets lead to the RIS graphs of 2.0 - 10^6, respectively, 108 - 10^6 triples. Their saturated versions comprise respectively 3.4 - 10^6 and 185 - 10^6 triples. Our first two RIS are thus: \( S_1 = \langle O_1, R, M_1, E_1 \rangle \) and \( S_2 = \langle O_2, R, M_2, E_2 \rangle \), where \( E_i \) for \( i \in \{1, 2\} \) the extents resulting from DS1 and M1.

Heterogeneous-sources RIS Second, going beyond relational-sources OBDA [16, 17, 44], our architecture extends to heterogeneous data sources. For that, we converted a third (33%) of DS1, DS2 into JSON documents, and stored them into MongoDB, leading to the JSON source data denoted DS1, DS2, the relational sources DSr,1, DSr,2, the remaining (relational) data. Conceptually, for \( i \in \{1, 2\} \), the extension based on DSr,1 and extension based on DSr,1 form a partition of \( E_i \). We devise a set of JSON-to-RDF mappings to expose DSr,1 and DSr,2 into RDF, and denote \( M_3 \), the set of mappings exposing DSr,1 and DSr,2, together, as an RDF graph; similarly, the mappings \( M_4 \) expose DSr,2 and DSr,2 as RDF. Our last two RIS are thus: \( S_3 = \langle O_1, R, M_3, E_3 \rangle \) and \( S_4 = \langle O_2, R, M_2, E_4 \rangle \), where \( E_3 \) the ex-tent of \( M_3 \) based on DSr,1 and DSr,1, while \( E_4 \) is the extent of \( M_4 \) based on DSr,2 and DSr,2. The RIS data and ontology triples of \( S_3 \) and \( S_4 \) are identical; thus, the difference between these two RIS is only due to the heterogeneity of their underlying data sources. The same holds for \( S_2 \) and \( S_1 \).

Queries We devised a set of 28 BGP queries having from 1 to 11 triple patterns (5.5 on average), of varied selectivity (they re-turn between 2 and 330 · 10^6 results in \( S_1 \) and \( S_2 \) and between 2 and 4.4 · 10^6 results in \( S_2 \) and \( S_3 \)); 6 among them query the data and the ontology (recall Example 2.6), a capability which most

---

2https://ontosql.inria.fr

3https://downloads.sourceforge.net/project/bshmtools/bshmtools/bshmtools-0.2
competitor systems lack (see Section 6). Table 4 reports three query properties impacting query answering performance: the number of induced triples \(N_{\text{TGI}}\), the number of BGPOQ reformulations on the ontology \(\{Q_c, a\}\), ranging from 1 to 1225; this strongly determines the performance of answering such large union queries, recall Example 4.5), and its number of answers \(N_{\text{ANS}}\) on the two RIS groups \((S_1, S_3)\) and \((S_2, S_4)\). To further study the impact of the ontology on query evaluation complexity, we created \textit{query families} denoted \(Q_X, Q_X a\) etc. by replacing the classes and properties appearing in \(Q_X\) with their super classes or super properties in the ontology. In such a family, \(Q_X\) is the most selective, and queries are sorted in the increasing order of their number of reformulations. Our ontologies, mappings, queries, and experimental details are available online\(^4\).

### 5.3 Query answering performance

**REW inefficiency** We have conducted experiments\(^4\) using our six queries on ontological triples showing, as in Example 4.17 and Figure 4, an explosion of the size of the rewriting (number of CQs), compared to the rewriting produced by the two other approaches. \textit{On queries (also) over the ontology}, as explained in

---

\(^4\) Experiment web site: https://gitlab.inria.fr/mburon/ogc/blob/master/projects/het2onto-benchmark/bsbm/
As stated in Section 5.2, there is a scale factor of about 50 between $S_1$, $S_2$ on one hand, and $S_2$, $S_1$ on the other. Figures 5 and 6 show that the query answering times generally grow by less than 50, when moving from $S_1$ to $S_2$, and from $S_2$ to $S_1$. This is mostly due to the good scalability of PostgreSQL (in the all-relational RIS), Tatooine (itself building on PostgreSQL and MongoDB, in the heterogeneous RIS), and OntoSQL (for MAT). As discussed above, computation steps we implemented outside these systems are strongly impacted by the mappings, ontology and query; intelligently distributing the reasoning effort, as Rew-C does, avoids the heavy performance penalties that from which Rew-C and Rew-CA sometimes suffer.

### Impact of heterogeneity

Rew-CA and Rew-C incur a (modest) overhead when combining data from PostgreSQL and MongoDB (heterogeneous RIS) w.r.t. the relational-sources RIS. Part of this is due to the cost of marshalling data across system boundaries; the rest is due to imperfect optimization within Tatooine. Overall, the comparison demonstrates that RIS query answering is feasible and quite efficient even on heterogeneous data sources.

### 5.4 Experiment conclusion

In a setting where the data, ontology and mappings do not change, MAT is an efficient and robust query answering technique, at a rather high cost to materialize and saturate the RIS instance. In contrast, in a dynamic setting, Rew-C smartly combines partial reformulation and view-based query rewriting to efficiently compute query answers. The changes it requires when the ontology and mappings change (basically re-saturating mapping heads) are light and likely to be very fast. Thus, we conclude that Rew-C is the best query answering strategy for dynamic RIS.

### 6 RELATED WORK AND CONCLUSION

Ontologies have been used to integrate relational or heterogeneous data sources in mediators [49] with LAV views based on description logics [1, 37] or their combination with Datalog [28, 30]. Semantics have been used at the integration level since e.g., [20] for SGML and soon after for RDF [6, 7]; data is considered represented and stored in a flexible object-oriented model, thus no mappings are used.

Our work follows the OBDA paradigm introduced in [41]. This paradigm was conceived to enhance access to relational data by mappings to an ontology expressed in a dialect of the DL-Lite description logic family (typically DL-Liteq underpinning the OWL 2 QL profile of the W3C ontological language OWL 2). Mature DL-based systems include Mastro² [17] and Ontop⁶ [16, 43]. Another notable OBDA system, namely UltrawrapOBDA [44], is based on an extension of RDFS to inverse and transitive properties. All these systems rely on GAV mappings.

Compared to these, our main novelty is to handle GLAV mappings and provide query answering algorithms for the resulting novel RIS setting. Note that formal OBDA frameworks with GLAV mappings have long been defined, e.g., in [18], but not put into practice. Regarding the other components of OBDA, we consider a simpler ontological language than existing OBDA systems, but support BGQs on both data and ontological triples, a feature hardly found in these systems (an exception is [33]).

As explained in the introduction, GLAV mappings maximize the expressive power of the integration system. In particular, they allow to expose a form of incomplete information (recall Example 3.6). To some extent, GLAV mappings may be simulated by GAV mappings provided with so-called Skolem functions on answer variables, as suggested for instance in [21]. To illustrate, consider the GLAV mapping $m_1 = q_1(\overline{x}) \sim q_2(\overline{x})$ with head $q_2(\overline{x}) \leftarrow (\overline{x} . : \text{CEOOf} , q_1 (\overline{y} , r . : \text{NatComp})$ from Example 3.2. The non-answer variable $y$ could be replaced by a Skolem function $f(\overline{x})$, which would yield two GAV mappings, namely $m_1 = q_1(\overline{x}) \sim q_2(\overline{x})$ and $m_2 = q_1(\overline{x}) \sim q_2(\overline{x})$, with respective head $q_2(\overline{x}) \leftarrow (\overline{x} . : \text{CEOOf} , f(\overline{x}) , y . : \text{NatComp})$. Note that Skolem functions would have to produce syntactically correct RDF values in a materialization scenario. Still in a materialisation scenario, query answering would require some post-processing to prevent the values built by the Skolem functions to be accepted as answers, while in a query rewriting scenario functional values would also have to be dealt with in a special way, which in particular prevents to use off-the-shelf view-based query rewriting algorithms. Hence, value invention would be simulated here at the price of technically more complex mappings and processing. Second, the break-up of GLAV mappings into several GAV mappings would lead to higher conceptual complexity since intrinsically connected triples, as those associated with $(\overline{x} . : \text{CEOOf} , y)$ and $(\overline{x} , r . : \text{NatComp})$ in the example, could not be exposed together by a single mapping. Last but not least, query rewriting would be considerably slowed down and would produce highly redundant rewritings, as demonstrated in the seminal paper [42].

Our mapping saturation (Definition 4.8) is inspired by a query saturation technique introduced in [25] to compute least general generalizations of BGQs under RDFS background knowledge.

---

²http://obdasmultimappers.com/mastro/
⁶https://ontop.inf.unibz.it/
It can be seen as a generalization to GLAV mappings of the ℱ-mapping technique introduced in [43] and further developed in [44] to optimize query rewriting in a classical OBDA context. The ℱ-mapping technique consists of completing the original set of GAV mappings with new ones, encapsulating information inferred from the DL ontology. For instance, given a GAV mapping \( m = q_1(x) \rightarrow q_2(x) \) with \( C(x) \) a class, and a DL constraint specifying that \( C \) is a subclass of \( D \), a new mapping \( m^* = q_1(x) \rightarrow q_3(x) \) is created by composing \( m \) and the DL constraint. On this example, we would saturate the head of \( m \) into \( q_2(x) \leftarrow C(x) \land D(x) \), which is semantically equivalent to adding the mapping \( m^* \). However, when mappings are GLAV and not GAV, one cannot simply add new mappings. For instance, consider the GLAV mapping \( m_1 = q_1(x) \rightarrow q_2(x) \) with head \( q_2(x) \equiv \langle x, :CEOOf, y \rangle, (y, \tau, :NatComp) \); given the entailment rule rdf\text{+}s a and the ontological triple \((:NatComp, \tau<_{sc} C, :Comp)\), the saturation adds the triple \((y, \tau, :Comp)\) to the body of \( q_2 \); creating instead a new mapping of the form \( m_2 = q_1(x) \rightarrow q_3(x) \) with head \( q_3(x) \equiv \langle y, \tau, :Comp \rangle \) would be unsatisfactory as \( y \) in \( m_1 \) should correspond to the same object as \( y \) in \( m_2 \).

Our mapping saturation technique could be extended to more general entailment rules, in which the head of the rules may include blank nodes that are not in their body, possibly shared by several triples. This is part of our future research agenda.

Acknowledgements: This work is supported by the Inria Project Lab iCoda and the ANR project CQFD (ANR-18-CE23-0003).

REFERENCES


