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 manner. 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 their 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. Note that we also include in this table theoretical frameworks that did not necessarily lead to implementations.

A first dimension concerns the data model and query language provided by the mediator to its applications.

(i) The earliest goal of a mediator system was to mimic a single, integrated database. Thus the mediator supports one data model and its query language, e.g., relational and SQL, or XML and XPath/XQuery. More recent polystore systems support side-by-side different (data model, query language) pairs. These database-style mediators appear in the row we label DB in Table 1.

(ii) Mediators studied in knowledge representation and management research provide a view of the data sources as a set of classes and relationships, also endowed with a set of semantic constraints, or ontology. In such systems, users formulate conjunctive (relational) queries; answering them involves not only evaluation over the data (as done in DB mediators), but also reasoning on the data with the help of ontologies. This mediation approach is also commonly termed Ontology-Based Data Access (OBDA) \cite{41}, with ontologies expressed in Description Logics (DL, in short). Work following this approach are listed in the row we label CQ in Table 1.

(iii) RDF \cite{47} is naturally suited as an integration model, thanks to its flexibility, its wide adoption in the Open Data community, its close relationship with ontology languages such as RDFS and OWL, and the presence of its associated standard SPARQL query language. Accordingly, several mediators from the CQ group have been extended to support RDF as an integration model and SPARQL query answering. However, while SPARQL allows querying the data together with the ontology, e.g., “find the properties of node n, as well the classes to which the values of these properties belong”, a DL-based mediation approach shares with all logic-based query languages, e.g., Datalog, SQL etc., the inability to do so. RDF mediators which support SPARQL but limited to querying the data only (not the ontology) appear in the row we label SPARQL-data in Table 1.

(iv) Recent RDF mediators lift this limitation to support joint querying of the data and ontology; we list them in the SPARQL row in Table 1.

A second dimension is how the source (or local) schemas are connected to the global (integration) schema, using mappings \cite{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 \cite{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

<table>
<thead>
<tr>
<th>Model</th>
<th>Mappings</th>
<th>DB</th>
<th>CQ</th>
<th>SPARQL-data</th>
<th>SPARQL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GAV</td>
<td>[22, 24, 27]</td>
<td>[4, 5, 22, 38]</td>
<td>[19]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CQ</td>
<td>[40, 41, 43]</td>
<td>[1, 28, 30, 36]</td>
<td>[18]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPARQL</td>
<td>[11, 17, 32, 34]</td>
<td>[45]</td>
<td>[21]</td>
<td>this work</td>
</tr>
</tbody>
</table>

Table 1: Outline of the positioning of our work.
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 LAV 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 \( (\mathcal{I} \cup \mathcal{B}) \times (\mathcal{L} \cup \mathcal{F} \cup \mathcal{B}) \), 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 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>
<thead>
<tr>
<th>Schema triples</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclass</td>
<td>((s, \prec_c, o))</td>
</tr>
<tr>
<td>Subproperty</td>
<td>((s, \prec_p, o))</td>
</tr>
<tr>
<td>Domain typing</td>
<td>((s, \leftarrow, d))</td>
</tr>
<tr>
<td>Range typing</td>
<td>((s, \leftarrow, r))</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)) such ( p \not\in {r, \prec_c, \prec_p, \leftarrow, \leftarrow, \rightarrow})</td>
</tr>
</tbody>
</table>

Table 2: RDF triples.
A popular RDF query dialect consists of entailment rules. An entailment rule has the form body(head), where body(head) is an RDF graph, respectively called body and head of the rule r. In this work, we consider RDF entailment rules 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 rdfs5 states: whenever in an RDF graph, a property p1 is a subproperty of a property p2, and further p2 is a subproperty of p3 (body of rdfs5), it follows that p1 is a subproperty of p2 and a resource has the value o for p1, then s also has o as a value for p2. The triples (p1, s, o) and (s, p2, o) 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 R as partitioned into two subsets: the rules R_c lead to implicit schema triples, while rules R_a lead to implicit data triples. The direct entailment of an RDF graph G with a set of RDF entailment rules R, denoted by C^G_c, is the set of implicit triples resulting from rule applications that use solely the explicit triples of G. For instance, the rule rdfs9 applied to the graph G_NatComp, as comprises (NatComp, _hs, .NatComp), (_hs, r, .NatComp), leads to the implicit triple (_hs, r, .Comp). This triple belongs to C^G_NatComp (hence also to C^G_NatComp), 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 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^G_c for i \geq 0. The saturation of G w.r.t. R, denoted C^G, is G_n for n the smallest integer such that G_n = G_{n+1}.

Example 2.4 (Saturation). The saturation of G_NatComp w.r.t. the set R of RDFs entailment rules shown in Table 3 is attained after the following two saturation steps:

(G_NatComp)_1 = (G_NatComp \cup
{[(\text{NatComp}, _hs, .Org)],
(\text{hiredBy}, _hs, .Person), (\text{hiredBy}, _hs, r, .Org),
(\text{ceoOf}, _hs, .Person), (\text{ceoOf}, _hs, r, .Org),
(p_1, \text{worksFor}, _hs), (_hs, r, .Comp),
(p_2, \text{worksFor}, a, (a, r, .Org))}

(G_NatComp)_2 = (G_NatComp) \cup
{[p_1, r, .Person], [p_2, r, .Person], (_hs, r, .Org)}}

The semantics of a BGP 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 Val(P) to 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 BGP q on an RDF graph G w.r.t. a set R of RDF entailment rules is: q(G, R) = \{\phi(x) | \phi \text{ homomorphism from body}(q) to C^G\}. If x = \emptyset, q is a Boolean query, in which case q is false when q(G, R) = \emptyset and true when q(G, R) = \{\}, 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 C^G). It can be seen as a particular case of query answering when R = \emptyset.

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

The above notions and notations naturally extend to unions of (partially instantiated) BGPQs, or UBGQs 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 R of entailment rules, so that the
answer set to an incoming query \( q \) is obtained through query evaluation as \( q(G^R) \).

### 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 \( \mathcal{R}_d \) and \( \mathcal{R}_c \).

(i) The first step reformulates a BGPO \( q \) w.r.t. an RDFS ontology \( O \) and the set of rules \( \mathcal{R}_d \) into a BGPO, say \( Q_c \), which is guaranteed not to contain ontological triples. Intuitively, this step generates new BGPOs obtained from \( q \) by instantiating variables that query the ontology with all their possible bindings; for instance, if \( y \) is in a query triple \((y, <s, -\text{comp})\) is bound to the IRI s 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 \( O \), \( q(G, \mathcal{R}_d) = Q_c(G) \).

(ii) The second step reformulates \( Q_c \) w.r.t. \( O \) and \( \mathcal{R}_d \), and outputs a BGPO, say \( Q_{c,a} \). This step, alone, is sound and complete w.r.t. \( \mathcal{R}_d \) for query answering, i.e., for any graph \( G \) with ontology \( O \), \( Q_c(G, \mathcal{R}_d) = Q_{c,a}(G) \). Furthermore, a key property is that \( q(G, \mathcal{R}) = Q_c(G, \mathcal{R}_d) \), i.e., only \( \mathcal{R}_d \) needs to be considered to answer \( Q_c \) with respect to the entire set of rules \( \mathcal{R} \). This is the fundamental reason why the successive application of these two reformulation steps leads to a sound and complete reformulation-based query answering technique: \( q(G, \mathcal{R}) = Q_{c,a}(G) \).

Example 2.9 (Two-step reformulation). Consider the query \( q(x, y) \leftarrow (x, \text{worksFor}, z), (z, \tau, y), (y, <\text{comp}) \) from the preceding example and the ontology \( O \) in Example 2.2. The first reformulation step instantiates the triple \((y, <\text{comp}) \) on \( O \), leading to: \( Q_c \). Then, \( Q_c \) is reformulated into \( Q_{c,a} \) as:

\[
q(x, \text{NatComp}) \leftarrow (x, \text{worksFor}, z), (z, \tau, \text{NatComp}) \cup q(x, \text{NatComp}) \leftarrow (x, \text{hiredby}, z), (z, \tau, \text{NatComp}) \cup q(x, \text{NatComp}) \leftarrow (x, \text{ceof}, z), (z, \tau, \text{NatComp})
\]

by specializing worksFor according to its subproperties in \( O \). It can be checked that \( Q_{c,a}(G_{CS}) = Q(G_{CS}, \mathcal{R}) = q(G_{CS}) \) = \{ (\psi_1, \text{NatComp}) \}, 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 \( I \) is made of a global schema \( S \) (a set of relations) and a set \( V \) of views. An instance of \( I \) assigns a set of tuples to each relation of \( S \) and to each view of \( V \). The data stored in a view is called its extension. Further, to each view \( V \) is associated a query \( \psi(V) \) - \( \psi \) 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 \( \text{Emp}(\text{id}, \text{name}, \text{did}), \text{Dept}(\text{did}, \text{cl}, \text{country}), \text{Salary}(\text{id}, \text{amount}) \), where \( \text{id}, \text{did} \) and \( \text{cl} \) are respectively identifiers for employees, departments and companies.

Consider the views \( V_1 \) and \( V_2 \) over the data sources and another query \( q_2 \) over the global schema, both with the same answer variables, e.g., \( q_1 = V_1^D \) and \( q_2 = V_2 \) above, is commonly called a LAV mapping (denoted \( q_1 \sim V_1^D \)).
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 \( \text{body}(q_2) \) is restricted to a single atom if \( q_2 \) is a CQ, or a single triple pattern if \( q_2 \) is a BGPOQ) and its variables are exactly \( x \). Second, local-as-view (LAV) mappings express elements of the local schema as views over the global schema, similarly to the views described in Section 2.5.1. Importantly, unlike GAV mappings, GLAV ones do not require all variables of \( q_2 \) to be answer variables (e.g., \( dID \) in \( V_1 \) in Figure 1); this makes integration more powerful. For example, suppose that \( 1 \) \( \{\text{"John Doe"}, \text{"France"}\} \) is an answer to \( V_1 \) above. Then, \( V_1 \) exposes this tuple in the global schema as: \( \text{Emp}(1, \"John Doe\"), \text{Dept}(x, \"IBM","France"), \) stating that John Doe works for a department \( x \) located in France. Here, \( x \) is an existential variable (called "labeled null" in [3]); the GLAV mapping states the existence of such a department in the global schema, even if its identifier is unknown (because it is not provided by \( V_1 \)). Therefore, John Doe is a certain answer to a query asking for all employees in IBM departments, based on the above GLAV mapping. This answer cannot be found using GAV mappings.

3 PROBLEM STATEMENT

In this section, we first formalize the notion of RDF integration problem (Section 3.1). Then, we state the associated query answering problem (Section 3.2), for which Section 4 provides solutions.

3.1 RDF integration system (RIS)

In an RDF integration system (RIS in short), data from heterogeneous sources, each of which may have its own data model and query language, is integrated into an RDF graph, consisting of an (RDFS) ontology and of data triples derived from the sources by means of GLAV-style mappings. Mappings allow (i) specifying the data made available from the sources, and (ii) organizing it according to the RIS ontology.

Definition 3.1 (RIS mappings and extensions). A RIS mapping \( m = q_1(x) \sim q_2(x) \) where \( q_1 \) and \( q_2 \) are two queries with the same answer variables, and \( q_2 \) is a BGPOQ whose body contains only triples of the forms:

- \( (s, p, o) \) where \( p \in \mathcal{F}_{\text{user}} \).
- \( (s, c, v) \) where \( c \in \mathcal{F}_{\text{user}} \).

The body of \( m \) is \( q_1 \), and its head is \( q_2 \). The extension of \( m \) is the set of tuples \( \text{ext}(m) = \{V_m(\delta(v_1), \ldots, \delta(v_n)) \mid \langle v_1, \ldots, v_n \rangle \in q_1(D)\} \), where \( q_1(D) \) is the answer set of \( q_1 \) on the data source \( D \) that \( m \) integrates and \( \delta \) is a function that maps source values to RDF values, i.e., IRIs, blank nodes and literals.

Intuitively, \( m \) specifies that the result of query \( q_1 \) on \( D \) transformed in RDF, i.e., the extension of \( m \), is exposed to the RIS as the result of the (BGPOQ) query \( q_2 \).

Example 3.2 (Mappings). Consider the two mappings:

- \( m_1 \) with head \( q_1(x) \leftarrow (x, \text{CEOOf}, y), (y, r, \text{NatComp}) \) and \( m_2 \) with head \( q_2(x, y) \leftarrow (x, \text{hiredBy}, y), (y, r, \text{PubAdmin}) \).

Suppose that the body of \( m_1 \) returns \( \{p_1\} \) as its results, and that the \( \delta \) function maps the value \( p_1 \) from the data source \( D_1 \) to the IRI \( p_1 \). Then, the extension of \( m_1 \) is: \( \text{ext}(m_1) = \{V_m(p_1)\} \).

Further, suppose that the body of \( m_2 \) returns \( \{p_2, a_2\} \), and that \( \delta \) maps the values \( p_2 \rightarrow a_2 \) from the data source \( D_2 \) to the IRIs \( p_2, a_2 \). Then, the extension of \( m_2 \) is: \( \text{ext}(m_2) = \{V_m(p_2, a_2)\} \).

Given a set of RIS mappings \( M \), the extent of \( M \) is the union of the mappings’ extensions, i.e., \( E = \bigcup_{m \in M} \text{ext}(m) \), and we denote by \( \text{Val}(E) \) the set of values occurring in \( E \). We can now define the RIS data triples induced by some mappings and an extent thereof. These are all the data which is exposed (can be queried) through a RIS.

Definition 3.3 (RIS data triples). Given a set \( M \) of RIS mappings and an extent \( E \) of \( M \), the RIS data triples induced by \( M \) and \( E \) form an RDF graph defined as follows:

\[
\mathcal{G}^M_E = \bigcup_{m=q_1(x) \sim q_2(x) \in M} \{\text{bgp2rdf}(\text{body}(q_2)(\langle x, \ldots, l \rangle)) \mid V_m(l) \in E\}
\]

where:

- \( \text{body}(q_2)(\langle x, \ldots, l \rangle) \) is the BGP body \( q_2(x) \) in which the answer variables \( x \) are bound to the values in the tuple \( V_m(l) \), part of \( E \);
- \( \text{bgp2rdf}(\cdot) \) 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 \( E = \{V_m(p_1), V_m(p_2, a)\} \). The RIS data triples they lead to are:

\[
\mathcal{G}^M_E = \{(p_1, \text{CEOOf}, _b_{-b}), (_b_{-b}, r, \text{NatComp}), (p_2, \text{hiredBy}, a), (a, r, \text{PubAdmin})\}
\]

In particular, the first and second triples contain the blank node \( _b_{-b} \), introduced by \( \text{bgp2rdf} \) 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 \( V_m(l) \), thus from \( \text{Val}(E) \)).

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 RDFS 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 BGPOQs 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 BGPOQ \( q \) on a RIS \( S = (O, R, M, E) \) is:

\[
\text{cert}(q, S) = \{\varphi(x) \mid \varphi \text{ homomorphism from body}(q) \text{ to } (O \cup \mathcal{G}^M_E) \text{ such that } \varphi(x) \text{ comprises only values from } \text{Val}(E)\}
\]

The certain answer set \( \text{cert}(q, S) \) is thus the subset of \( q(O \cup \mathcal{G}^M_E) \) 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 \( \text{Gex} \) 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 \( q(y, x) \leftarrow (x, \text{worksFor}, y), (y, r, \text{Comp}) \) be the query asking “who works for which company”, while the query \( q’(x) \leftarrow (x, \text{worksFor}, y), (y, r, \text{Comp}) \) 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 \( \emptyset \),
The obtained reformulated query \( Q_{c,a} \) using the RIS GLAV mappings \( M \) seen as LAV views (step (2)). This produces a relational rewriting \( q_{rew} \) 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 \( R_c \) only (not \( R = R_c \cup R_a \) as above). The obtained reformulation \( Q_c \) thus yields the expected certain answer set when evaluated on the RIS data triples saturated with \( R_a \) (recall Section 2.4), again provided that the answers with blank nodes introduced by the bgp2rdf function are discarded (as above). Since these triples are not materialized in a RIS, hence cannot be saturated with \( R_a \), the saturation-based query answering problem is in turn reduced to relational view-based query answering, by rewriting \( Q_c \), using the RIS GLAV mappings saturated \( O \) and \( R_a \), 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 implicit data triples they model w.r.t. \( O \) and \( R_a \). Then, the partially reformulated query \( Q_c \) 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 \( q_{rew} \) 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 \( M^{O,R} \) (step (B)), comprising all the explicit and implicit RIS schema triples w.r.t. \( O \) and \( R \), since only \( R_a \) rules entail new schema triples (Table 3), \( O^R \) is actually equal to \( O^R \). 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 \( M^{a,O} \cup M^{O,R} \), seen as LAV views (step (2')), in order to obtain, as above, a rewriting \( q_{rew} \) 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 \( bgp2cu \) function transforms a BGP into a conjunction of atoms with ternary predicates over the data sources.

**4 QUERY ANSWERING IN A RIS**

Since we adopt a mediator-style approach, the RIS data triples \( G_E \) are not materialised, hence the saturation of \( O \cup G_E^M \) cannot be computed to answer queries as defined above. Instead, queries are rewritten in terms of the remote heterogeneous sources, based on the RIS ontology \( O \), reasoning power \( R \) and mappings \( M \). We present three query answering strategies, which differ in how reasoning is incorporated: we may have all, some or no reasoning performed at query time, as outlined in Figure 2.

**All reasoning at query time** The first strategy will be detailed in Section 4.1. First, it reduces the RIS query answering problem to standard query evaluation in an RDF data management system, by reformulating (step (1) in Figure 2) the query \( q \) based on the RIS ontology \( O \) and entailment rules \( R = R_c \cup R_a \).

The obtained reformulated query \( Q_{c,a} \) thus yields the expected certain answers when evaluated on the RIS data triples (recall Section 2.4), provided that answers with blank nodes introduced by the bgp2rdf function are discarded (recall Section 3.2). Since these data triples are not materialized, the RDF query evaluation process is in turn reduced to relational view-based query answering, by rewriting \( Q_{c,a} \) using the RIS GLAV mappings \( M \) seen as LAV views (step (2)). This produces a relational rewriting \( q_{rew} \) over the mappings extension (step (3)), whose evaluation with a mediator query engine provides the desired certain answers (steps (4) and (5)).

**4.1 Rewriting Fully-Reformulated Queries using Mappings as Views: rew-ca**

Based on [12], the first step of this strategy, (1) in Figure 2, reformulates a query \( q \) w.r.t. \( O \) and \( R = R_c \cup R_a \) into a query \( Q_{c,a} \). 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 BGQ and $Q_{c,a}$ its UBGPQ reformulation w.r.t. $O, R = R' \cup R_a$ using [12]. Then:

$$\text{cert}(q, S) = \text{cert}(Q_{c,a}, \{(0, 0, M, E)\})$$

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 $m = q_1(x) \rightsquigarrow q_2(x) \in M$, (i) evaluating the mapping body $q_1(x)$ on the data source to produce its extension $\text{ext}(m) \in E$, and (ii) then 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 $m = q_1(x) \rightsquigarrow q_2(x)$ be a mapping. Its corresponding relational LAV view is:

$$V_m(x) \leftarrow \text{bgp2ca(body}(q_2)).$$

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

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

We denote the set of views derived from all the mappings $M$ by Views($M$). Crucially, the extent $E$ 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 BGQ. Let $Q_{c,a}$ be the reformulation of $q$ w.r.t. $O$ and $R$ using [12]. Then:

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

where cert(bgpa2cupQ($Q_{c,a}$), Views($M$), $E$) denotes the certain answer set of bgpqa2cupQ($Q_{c,a}$) 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 $q(x, y) \leftarrow (x, y, z), (x, r, t), (y, \neg \text{sp}, \neg \text{worksFor}), (t, \neg \text{ceo}, \text{Comp}), (x, \text{worksFor}, a), (a, a, r, \text{PubAdmin})$ asking "who works for some public administration, and what working relationship he/she has with some company". Its UBGPQ reformulation, seen as a UCQ, 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(y), \text{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 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: $E = \{V_m(q_1), V_m(q_2), \ldots\}$. However, if we add $V_m(q_1, a)$ to $E$, then cert($q, S) = \{q_1, \text{ceoOf}\}$.

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

In constrast with the REW-CA strategy that performs all the reasoning w.r.t. $O$ and $R = R' \cup R_a$ 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. $O, R_c$, producing a UBGPQ denoted $Q_c$. From the correctness of this reformulation step, and the fact that only $R_a$ 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 $R_c$. Formally:

**Lemma 4.6.** Let $S = (O, R, M, E)$ be a RIS, $q$ be a BGQ and $Q_c$ its reformulation w.r.t. $O, R_c$ [12]. Then:

$$\text{cert}(q, S) = \text{cert}(Q_c, (O, R_a, M, E))$$

In other words, the desired answer set could be obtained by evaluating $Q_c$ on the RIS data triples $G^M$ saturated by $R_a$. 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 BGQPQ saturation [25]: given a BGQ $q$, $O$ and $R$, the saturation $q_{R,O}$ is augmented with all the triples $q$ implicitly asks for, given the ontology $O$ and the rules $R$. BGQPQ saturation is exemplified below:

**Example 4.7 (BGQPQ saturation).** Consider the ontology $O$ of $G_{ax}$ and the query $q(x) \leftarrow (x, \text{hiredBy}, y), (y, r, \text{NatComp})$ asking who has been hired by a national company. Its saturation w.r.t. $R_a, O$ is:

$$q_{R_a,O}(x) \leftarrow \text{body}(q(x), (x, \text{worksFor}, y), (y, r, \text{Person}), (y, r, \text{Comp}), (y, r, \text{Org}))$$

We use BGQPQ saturation to saturate the RIS mapping heads w.r.t. $R_a, O$, so that the saturated mappings together with $E$ model the saturated RIS data triples w.r.t. $R_a, O$. To compute $q_{R_a,O}$ we (1) saturate body($q$) $\cup O$ using $R_a$, 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 $R_a$ and ontology $O$ is:

$$M_{a,O} = \bigcup_{m \in M} \{q_1(x) \rightsquigarrow q_2, q_{R_a,O}(x) \mid m = q_1(x) \rightsquigarrow q_2(x)\}$$

In Figure 3: Sample reformulation for Example 4.5.
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 triplets are in blue):

\[
m_1 : \ \{(x, \text{CEOOf}, y) \mid (x, \text{Person}, y, r, \text{NatComp}) \}\ 
\leq\{(x, \text{worksFor}, y, (y, r, \text{Comp})) \\
\leq\{(x, r, \text{Person}), (y, r, \text{Org}) \}\}\n\]

\[
m_2 : \ \{(x, \text{hiredBy}, y) \mid (x, \text{hiredBy}, y, (y, r, \text{PubAdmin}) \}\ 
\leq\{(x, \text{worksFor}, y) \leq\{(x, r, \text{Org}) \\}
\]

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

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

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

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 \( S = \langle O, R, M, E \rangle \) be a RIS, \( q \) be a BGPQ and \( Q_c \) its reformulation w.r.t. \( O, R, E \). Then:

\[
cert(q, S) = cert(uqpq2ucq(Q_c), \text{Views}(M^{a,O}, E))
\]

**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. \( O, R, E \), seen as a UCQ, is:

\[
q(x, \text{CEOOf}) \leftrightsquigarrow T(x, \text{CEOOf}, z), (z, r, \text{NatComp}), (z, \text{worksFor}, a), (a, r, \text{PubAdmin})
\]

**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:

\[
M_{O^{\mathcal{E}}} = \bigcup_{x \in \{s, o\}} \{m_x \mid m_x = q_2(s, o) \rightarrow q_2(s, o)\}
\]

with \( q_2(s, o) \leftrightsquigarrow (s, o, \text{CEOOf}) \). The extension of an ontology mapping \( m_x \) is \( ext(m_x) = \{V_m(x, s, o) \mid (s, x, o) \in O^{\mathcal{E}}\} \). The extent of \( M_{O^{\mathcal{E}}} \) is denoted \( \mathcal{E}^{\mathcal{O}} \).

We compute ontology mappings offline, and only need to update them when the ontology changes. The ontology mapping extensions \( \mathcal{E}^{\mathcal{O}} \) store all the explicit and implicit RIS ontology triples (recall from Section 2.2 that only \( R_e \) lead to such triples). 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:

\[
q(x, \text{CEOOf}) \leftrightsquigarrow \{V_m(x, \text{CEOOf}, \text{worksFor}), V_m(x, \text{NatComp}, \text{Comp}), V_m(x, a)\}
\]

\[
\cup q(x, \text{CEOOf}) \leftrightsquigarrow \{V_m(x, \text{CEOOf}, \text{worksFor}), V_m(x, \text{NatComp}, \text{Comp}), V_m(x, a)\}
\]

\[
\cup q(x, \text{CEOOf}) \leftrightsquigarrow \{V_m(x, \text{CEOOf}, \text{worksFor}), V_m(x, \text{NatComp}, \text{Comp}), V_m(x, a)\}
\]

\[
\cup q(x, \text{worksFor}) \leftrightsquigarrow \{V_m(x, \text{worksFor}, \text{worksFor}), V_m(x, \text{NatComp}, \text{Comp}), V_m(x, a)\}
\]

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

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

\[
cert(q, S) = cert(q, \langle O, R, M_{O^{\mathcal{E}}} \cup M, E^{\mathcal{O}} \cup E \rangle)
\]

This lemma effectively "pushes" \( R_e \) reasoning in the set of mappings (to which we add \( M_{O^{\mathcal{E}}} \)) and the extent (to which we add \( \mathcal{E}^{\mathcal{O}} \)). Next, we rely (as we did forREW-CA) on mappings saturation with \( O, R_e \) to also push \( R_e \) reasoning in the mappings, leading to:

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

\[
cert(q, S) = cert(q, \langle O, R_e, M_{O^{\mathcal{E}}} \cup M, E^{\mathcal{O}} \cup E \rangle)
\]

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

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

\[
cert(q, S) = cert(bqpq2ucq(q), \text{Views}(M_{O^{\mathcal{E}}} \cup M^{a,O}, E^{\mathcal{O}} \cup E))
\]

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

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

Its maximally-contained rewriting \( q_{rew} \) 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 \( V_m(x, a) \), as we did in...
We rely on the Graal engine [9] for the evaluation of the first CQ in the UCQ rewriting; the other CQs yield empty results because some required $\leq_{sc}$ or $\leq_{sp}$ contraints are not found in the views built from the RIS ontology mappings.

**How do our strategies compare?** Since they are all correct, they lead to the same RIS certain answer set, however they do not necessarily compute the same view-based rewritings. Indeed, $\text{REW}$ considers the additional set $M_{OC}$ of ontology mappings. Hence, for queries over the ontology, i.e., featuring in a property position $\leq_{sc}$, $\leq_{sp}$, $\leq_{df}$, $\leq_{tr}$, or a variable, a $\text{REW}$ rewriting is larger than a $\text{REW-CA}$ or $\text{REW-C}$ rewriting and, to be answered, requires the additional ontology source. In contrast, $\text{REW-CA}$ and $\text{REW-C}$ yield logically equivalent rewritings; we minimize them both to avoid possible redundancies, thus they become identical (up to variable renaming). Hence, $\text{REW-CA}$ and $\text{REW-C}$ do not differ in how these rewritings are evaluated. Instead, they differ in how the rewritings are computed, or, equivalently, on the distribution of the reasoning effort on the data and mappings, across various query answering stages. As our experiments show, given the computational complexity of view-based query rewriting [42], this difference has a significant impact on their performance.

5 EXPERIMENTAL EVALUATION

We now describe our experiments with RIS query answering. In addition to our strategies based on query rewriting, we include 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 amounts to query evaluation 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 PostgreSQL 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 rewriting. Graal is a Java toolkit dedicated to query answering algorithms in knowledge bases with existential rules (a.k.a. tuple-generating dependencies). Since the relational view $V_m(x) \leftarrow bgp2ca(body(q_2))$ corresponding to a GLAV mapping $m$ (recall Def. 4.2) can be seen as a specific existential rule of the form $V_m(x) \leftarrow bgp2ca(body(q_2))$, the query reformulation algorithm of Graal can be used to rewrite the UCQ translation of a BGPQ 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 $q_i$ 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, value etc. Using two different benchmark scale factors, we obtained a data source $DS_1$ of 154,054 tuples across the relations, respectively, $DS_2$ of 7,843,660 tuples; both are stored in Postgres. We used two RDFS ontologies $O_1$ respectively $O_2$, containing, first, subclass hierarchies of 151 (resp. 2011) product types, which come with $DS_1$; respectively, $DS_2$. To $O_1$ and $O_2$, 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 $M_1$, $M_2$ of 307, respectively, 3,863 mappings, which expose the relational data from $DS_1$, respectively, $DS_2$ 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 $2.0 \cdot 10^6$, respectively, $108 \cdot 10^6$ triples. Their saturated versions comprise respectively $3.4 \cdot 10^6$ and $185 \cdot 10^6$ 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 $i \in \{1, 2\}$ are the extents resulting from $DS_i$ and $M_i$.

**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 $DS_1$, $DS_2$ into JSON documents, and stored them into MongoDB, leading to the JSON data sources denoted $DS_{1,j}$, $DS_{2,j}$; the relational sources $DS_{1,r}$, $DS_{2,r}$ store the remaining (relational) data. Conceptually, for $i \in \{1, 2\}$, the extension based on $DS_{1,r}$ and extension based on $DS_{2,r}$ form a partition of $E_i$. We devise a set of JSON-to-RDF mappings to expose $DS_{1,j}$ and $DS_{2,j}$ into RDF, and denote $M_3$ the set of mappings exposing $DS_{1,j}$ and $DS_{2,j}$, together, as an RDF graph; similarly, the mappings $M_4$ expose $DS_{1,r}$ and $DS_{2,r}$ as RDF. Our last two RIS are thus: $S_3 = \langle O_1, R, M_3, E_3 \rangle$ and $S_4 = \langle O_2, R, M_4, E_4 \rangle$, where $E_3$ is the extent of $M_3$ based on $DS_{1,r}$ and $DS_{2,r}$, while $E_4$ is the extent of $M_4$ based on $DS_{1,r}$ and $DS_{2,r}$. 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 return between 2 and $330 \cdot 10^6$ results in $S_1$ and $S_2$ between 2 and $4.4 \cdot 10^6$ results in $S_2$ and $S_4$); 6 among them query the data and the ontology (recall Example 2.6.2), a capability which most

---

²https://ontosql.inria.fr

³https://downloads.sourceforge.net/project/bshmtools/bshmtools/ bshmtools-G2
Table 4: Characteristics of the queries used in our experiments.

<table>
<thead>
<tr>
<th>RIS</th>
<th>Q01</th>
<th>Q01a</th>
<th>Q01b</th>
<th>Q02</th>
<th>Q02a</th>
<th>Q02b</th>
<th>Q02c</th>
<th>Q03</th>
<th>Q04</th>
<th>Q07</th>
<th>Q07a</th>
<th>Q09</th>
<th>Q10</th>
<th>Q13</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>N㎝T</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>S₁, S₂</td>
<td>N㎝A</td>
<td>21</td>
<td>21</td>
<td>175</td>
<td>21</td>
<td>49</td>
<td>147</td>
<td>1225</td>
<td>525</td>
<td>5</td>
<td>1</td>
<td>15</td>
<td>17</td>
<td>79</td>
</tr>
<tr>
<td>S₁</td>
<td>N㎝A</td>
<td>1722</td>
<td>4376</td>
<td>22738</td>
<td>16</td>
<td>56</td>
<td>174</td>
<td>1342</td>
<td>19</td>
<td>91</td>
<td>2</td>
<td>3</td>
<td>5617</td>
<td>9</td>
</tr>
<tr>
<td>S₂</td>
<td>A㎝A</td>
<td>21</td>
<td>175</td>
<td>1407</td>
<td>63</td>
<td>147</td>
<td>525</td>
<td>1225</td>
<td>4375</td>
<td>1</td>
<td>5</td>
<td>19</td>
<td>735</td>
<td>9550</td>
</tr>
<tr>
<td>S₂</td>
<td>N㎝A</td>
<td>15514</td>
<td>111793</td>
<td>86379</td>
<td>124</td>
<td>598</td>
<td>1058</td>
<td>1570</td>
<td>5</td>
<td>4487</td>
<td>2</td>
<td>3</td>
<td>299902</td>
<td>10</td>
</tr>
<tr>
<td>S₃</td>
<td>N㎝A</td>
<td>4416946</td>
<td>10049829</td>
<td>2998948</td>
<td>249004</td>
<td>39826</td>
<td>60834</td>
<td>904</td>
<td>7818</td>
<td>10486</td>
<td>51988</td>
<td>37176</td>
<td>1528</td>
<td>18588</td>
</tr>
<tr>
<td>S₃</td>
<td>N㎝A</td>
<td>330142</td>
<td>5628</td>
<td>5628</td>
<td>1</td>
<td>201</td>
<td>525</td>
<td>1225</td>
<td>4221</td>
<td>9350</td>
<td>40</td>
<td>520</td>
<td>192</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5: Query answering times on the smaller RIS S₁ (top, relational sources) and S₃ (bottom, heterogeneous sources).}

Figure 6: Query answering times on the larger RIS S₂ (top, relational sources) and S₃ (bottom, heterogeneous sources).

Section 4.3, we noted that the size of the rewriting produced by \( \text{REW} \) is larger (by a multiplicative factor of 29 to 74 in S₁ and S₃, and of 33 to 969 in S₂) than the rewritings of the two other strategies, which led to an explosion of the time spent minimizing the rewriting, and made \( \text{REW} \) overall unfeasible; the details of these tests can be found online\(^4\). On queries that do not carry over the ontology, \( \text{REW} \) produces the same rewritings as the other methods. Thus, we do not report further \( \text{REW} \) performance below.

**Query answering time comparison** Figure 5 depicts the query answering times, on the smaller RIS, of \( \text{REW-CA}, \text{REW-C} \) and \( \text{MAT} \). The size of (number of BGPOs in) the reformulation of each query w.r.t. \( \mathcal{R} \) (\( |Q_{\mathcal{R}}| \)) appears in parentheses after the query name, in the labels along the x axis. Given that S₁, S₂ have the same RIS data triples, the \( \text{MAT} \) strategy coincides among these two RIS. Figure 6 shows the corresponding times for the largest RIS S₂ and S₃; the same observations apply. Note the logarithmic time axes.

A first observation is that our query set is quite diverse; their answering times range from a few to more than \( 10^5 \) ms.

As expected, query answering in \( \text{MAT} \) is the fastest in most cases, since it has no reasoning work to do at query answering time. However, it required, for S₁, S₂, 1.2 · 10^5 ms to build the materialization and 1.49 · 10^5 ms more to saturate it, whereas for S₃, these times are 14h46 (5.31 · 10^7 ms), respectively, 1h28 (5.28 · 10^6

---

\(^4\)Experiment web site: https://gitlab.inria.fr/mburon/ogg/blob/master/projects/het2onto-benchmark/bsbm
ms). Not only these are orders of magnitude more than all query answering times; recall also that materializing \( g^M_C \) requires maintaining it when the underlying data changes, and its saturation \( (g^M_C \cup O)^S \) needs a second level of maintenance. Thus, MAT is not practical when data sources change. We were surprised to see \( \text{REW-C} \) and \( \text{REW-CA} \) somehow faster than MAT for queries \( Q_{20} \) and \( Q_{24} \). Answering these queries through MAT within OntoSQL leads to producing many results that involve mapping-generated blank nodes, tuples which should not appear in our certain answers, as per Definition 3.5. We remove such tuples in post-processing mode, which leads to a performance overhead for MAT. \( \text{REW-C} \) and \( \text{REW-CA} \), in contrast, are answered by evaluating rewritings, and do not have to apply such a result pruning. It remains to be seen if this pruning could be pushed in an RDFDB; note that not all answers including blank nodes should be pruned, only those whose blank nodes are due to mappings.

In each scenario, we observe that \( \text{REW-C} \) is faster or takes as long as \( \text{REW-CA} \). Since the two approaches produce the same rewritings, the difference is due to steps before the step (3) in Figure 2. It turns out it is due to the rewriting time, which in turn strongly depends on the size of the reformulation it receives as input. In \( \text{REW-C} \), the reformulations w.r.t. \( R_C \) are of size 1 (no union, just one BGP) for queries on data triples only, and never exceed 64 in \( S_1 \) and \( S_3 \) and 200 in \( S_2 \) and \( S_4 \), whereas, in \( \text{REW-CA} \) the reformulation sizes are much larger. \( \text{REW-C} \) is most often faster than \( \text{REW-CA} \), by up to two orders of magnitude e.g., for \( Q_{20a}, Q_{19} \) and \( Q_{20a} \) on \( S_2 \), the latter two on \( S_1 \) etc. One order of magnitude speed-up is noticeable even on the smaller RIS \( S_1, S_2 \) (Figure 5) for \( Q_{20a} \). As a consequence, \( \text{REW-C} \) completes successfully in all scenarios we study, whereas \( \text{REW-CA} \) fails to complete for many queries with timeout set to 10min (missing yellow bars in Figure 6), in close correlation with the increased number of reformulations.

**Scaling in the data size** As stated in Section 5.2, there is a scale factor of about 50 between \( S_1 \), \( S_3 \), on one hand, and \( S_2 \), \( S_4 \) 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 \( S_3 \) to \( S_4 \). 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 \( \text{REW-C} \) does, avoids the heavy performance penalties that from which \( \text{REW-CA} \) and \( \text{REW} \) sometimes suffer.

**Impact of heterogeneity** \( \text{REW-CA} \) and \( \text{REW-C} \) 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, \( \text{REW-C} \) 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 \( \text{REW-C} \) 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 Mastro\(^6\) [17] and Ontop\(^6\) [16, 43]. Another notable OBDA system, namely Ultrawrap\(^\text{OBDA} \) [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 BGPQs 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 \( m_1 = q_1(x) \sim q_2(x) \) with head \( q_2(x) \leftarrow (x, \text{ceoOf}, q_1(y, \tau, \text{NatComp}) \) from Example 3.2. The non-answer variable \( y \) could be replaced by a Skolem function \( f(x) \), which would yield two GAV mappings, namely \( m_1 = q_1(x) \sim q_2(x) \) and \( m_1 = q_1(x) \sim q_2(x) \), with respective head \( q_2(x) \leftarrow (x, \text{ceoOf}, f(x)) \) and \( q_2(x) \leftarrow (f(x), \tau, \text{NatComp}) \). 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 \( (x, \text{ceoOf}, y) \) and \( (y, \tau, \text{NatComp}) \) 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 BGPQs under RDFS background knowledge.

\(^6\)http://obdasytems.com/mastro/

\(^6\)https://ontop.inf.unibz.it/
Acknowledgements: This work is supported by the Inria Project Lab iCoda and the ANR project CQFD (ANR-18-CE23-0003).

REFERENCES


[9] Ioana Manolescu, Daniela Florescu, and Donald Kossmann. 2001. Answering Queries Using Views over Heterogeneous Information Sources Using Source Descriptions. In PODS.


[28] Ioana Manolescu, Daniela Florescu, and Donald Kossmann. 2001. Answering Queries Using Views over Heterogeneous Data Sources. In VLDB.


[34] Ioana Manolescu, Daniela Florescu, and Donald Kossmann. 2001. Answering XML Queries on Heterogeneous Data Sources. In VLDB.


[40] Ioana Manolescu, Daniela Florescu, and Donald Kossmann. 2001. Answering XML Queries on Heterogeneous Data Sources. In VLDB.