Ontology-Based RDF Integration of Heterogeneous Data

Maxime Buron[1] François Goasdoué[2] Ioana Manolescu[1] Marie-Laure Mugnier[3]
[1]Inria and LIX (UMR 7161, CNRS and Ecole polytechnique), France
[2]Univ. Rennes, CNRS, IRISA, France [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 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) [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.

		Mappings		
		GAV	**LAV**	**GLAV**
Model	**DB**	[22, 24, 27]	[4, 5, 22, 38]	[19]
	CQ	[40, 41, 43]	[1, 28, 30, 36]	[18]
	SPARQL-data	[11, 17, 32, 34]	[45]	[21]
	SPARQL	[16, 33, 44]		this work

Table 1: Outline of the positioning of our work.

(*iii*) RDF [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* [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 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: $\mathscr{I}$ of IRIs (resource identifiers), $\mathscr{L}$ of literals (constants) and $\mathscr{B}$ of blank nodes modeling unknown IRIs or literals, a.k.a. *labelled nulls* [3, 29]. A *(well-formed) triple* belongs to $(\mathscr{I} \cup \mathscr{B}) \times \mathscr{I} \times (\mathscr{L} \cup \mathscr{I} \cup \mathscr{B})$, and an *RDF graph* G is a set of (well-formed) triples. A triple $(\mathsf{s}, \mathsf{p}, \mathsf{o})$ states that its *subject* s has the *property* p with the *object* value o [47]. We denote by $\mathrm{Val}(G)$ the set of all values (IRIs, blank nodes and literals) occurring in an RDF graph G, and by $\mathrm{Bl}(G)$ its set of blank nodes. In triples, we use $\_{:}b$ (possibly with indices) to denote blank nodes, and quoted strings to denote literals.

Schema triples	Notation
Subclass	$(\mathsf{s}, \prec_{sc}, \mathsf{o})$
Subproperty	$(\mathsf{s}, \prec_{sp}, \mathsf{o})$
Domain typing	$(\mathsf{s}, \hookleftarrow_{d}, \mathsf{o})$
Range typing	$(\mathsf{s}, \hookrightarrow_{r}, \mathsf{o})$
Data triples	Notation
Class fact	$(\mathsf{s}, \tau, \mathsf{o})$
Property fact	$(\mathsf{s}, \mathsf{p}, \mathsf{o})$ s.t. $\mathsf{p} \notin \{\tau, \prec_{sc}, \prec_{sp}, \hookleftarrow_{d}, \hookrightarrow_{r}\}$

Table 2: RDF triples.

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 $\mathscr{I}_{\mathrm{rdf}}$ the *reserved* IRIs from the RDF standard, e.g., the properties τ, $\prec_{sc}$, $\prec_{sp}$, $\hookleftarrow_{d}$, $\hookrightarrow_{r}$ shown in Table 2. The rest of the IRIs are application-dependent classes and properties, which are said *user-defined* and denoted by $\mathscr{I}_{\mathrm{user}}$. Hence, $\mathscr{I}_{\mathrm{user}} = \mathscr{I} \setminus \mathscr{I}_{\mathrm{rdf}}$.

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 $\mathscr{I}_{\mathrm{user}}$. 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 $(\hookleftarrow_{d}, \prec_{sp}, \hookrightarrow_{r})$, 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:

$G_{\mathrm{ex}} = \{$(:worksFor, $\hookleftarrow_{d}$, :Person), (:worksFor, $\hookrightarrow_{r}$, :Org),
(:PubAdmin, $\prec_{sc}$, :Org), (:Comp, $\prec_{sc}$, :Org),
(:NatComp, $\prec_{sc}$, :Comp), (:hiredBy, $\prec_{sp}$, :worksFor)
(:ceoOf, $\prec_{sp}$, :worksFor), (:ceoOf, $\hookrightarrow_{r}$, :Comp),
(:p$_1$, :ceoOf, $\_{:}b_c$), ($\_{:}b_c$, τ, :NatComp),
(:p$_2$, :hiredBy, :a), (:a, τ, :PubAdmin)$\}$

The ontology of G_{ex}, 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 G_{ex}, i.e., the four remaining data triples, state that :p$_1$ is CEO of some unknown company represented by the blank node $\_{:}b_c$, which is a national company, and :p$_2$ is hired by the public administration :a.

	Rule [48]	Entailment rule	
$\mathcal{R}$	`rdfs5`	$(\mathsf{p}_1, \prec_{sp}, \mathsf{p}_2), (\mathsf{p}_2, \prec_{sp}, \mathsf{p}_3) \rightarrow (\mathsf{p}_1, \prec_{sp}, \mathsf{p}_3)$	$\mathcal{R}_c$
	`rdfs11`	$(\mathsf{s}, \prec_{sc}, \mathsf{o}), (\mathsf{o}, \prec_{sc}, \mathsf{o}_1) \rightarrow (\mathsf{s}, \prec_{sc}, \mathsf{o}_1)$	
	`ext1`	$(\mathsf{p}, \hookleftarrow_d, \mathsf{o}), (\mathsf{o}, \prec_{sc}, \mathsf{o}_1) \rightarrow (\mathsf{p}, \hookleftarrow_d, \mathsf{o}_1)$	
	`ext2`	$(\mathsf{p}, \hookrightarrow_r, \mathsf{o}), (\mathsf{o}, \prec_{sc}, \mathsf{o}_1) \rightarrow (\mathsf{p}, \hookrightarrow_r, \mathsf{o}_1)$	
	`ext3`	$(\mathsf{p}, \prec_{sp}, \mathsf{p}_1), (\mathsf{p}_1, \hookleftarrow_d, \mathsf{o}) \rightarrow (\mathsf{p}, \hookleftarrow_d, \mathsf{o})$	
	`ext4`	$(\mathsf{p}, \prec_{sp}, \mathsf{p}_1), (\mathsf{p}_1, \hookrightarrow_r, \mathsf{o}) \rightarrow (\mathsf{p}, \hookrightarrow_r, \mathsf{o})$	
	`rdfs2`	$(\mathsf{p}, \hookleftarrow_d, \mathsf{o}), (\mathsf{s}_1, \mathsf{p}, \mathsf{o}_1) \rightarrow (\mathsf{s}_1, \tau, \mathsf{o})$	$\mathcal{R}_a$
	`rdfs3`	$(\mathsf{p}, \hookrightarrow_r, \mathsf{o}), (\mathsf{s}_1, \mathsf{p}, \mathsf{o}_1) \rightarrow (\mathsf{o}_1, \tau, \mathsf{o})$	
	`rdfs7`	$(\mathsf{p}_1, \prec_{sp}, \mathsf{p}_2), (\mathsf{s}, \mathsf{p}_1, \mathsf{o}) \rightarrow (\mathsf{s}, \mathsf{p}_2, \mathsf{o})$	
	`rdfs9`	$(\mathsf{s}, \prec_{sc}, \mathsf{o}), (\mathsf{s}_1, \tau, \mathsf{s}) \rightarrow (\mathsf{s}_1, \tau, \mathsf{o})$	

Table 3: Sample RDFS entailment rules.

2.2 RDF Entailment Rules

An *entailment rule* r has the form $\text{body}(r) \rightarrow \text{head}(r)$, where $\text{body}(r)$ and $\text{head}(r)$ are RDF graphs, respectively called *body* and *head* of the rule r. In this work, we consider the **RDFS 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 `rdfs5` reads: whenever in an RDF graph, a property p_1 is a subproperty of a property p_2, and further p_2 is a subproperty of p_3 (body of `rdfs5`), it follows that p_1 is a subproperty of p_3 (head of `rdfs5`). Similarly, rule `rdfs7` states that if p_1 is a subproperty of p_2 and a resource s has the value o for p_1, then s also has o as a value for p_2. The triples $(\mathsf{p}_1, \prec_{sp}, \mathsf{p}_3)$ and $(\mathsf{s}, \mathsf{p}_2, \mathsf{o})$ 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 **explicit**ly 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[1]. The **direct entailment** of an RDF graph G with a set of RDF entailment rules $\mathcal{R}$, denoted by $C_{G,\mathcal{R}}$, 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_{ex}, which comprises (:NatComp, $\prec_{sc}$, :Comp), (_:b_c, τ, :NatComp), leads to the implicit triple (_:b_c, τ, :Comp). This triple belongs to $C_{G_{\text{ex}},\mathcal{R}_a}$ (hence also to $C_{G_{\text{ex}},\mathcal{R}}$). 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 $G_{i+1} = G_i \cup C_{G_i,\mathcal{R}}$ 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_{ex} w.r.t. the set $\mathcal{R}$ of RDFS entailment rules shown in Table 3 is attained after the following *two* saturation steps:

$(G_{\text{ex}})_1 = G_{\text{ex}} \cup$
{(:NatComp, $\prec_{sc}$, :Org),
(:hiredBy, $\hookleftarrow_d$, :Person), (:hiredBy, $\hookrightarrow_r$, :Org),
(:ceoOf, $\hookleftarrow_d$, :Person), (:ceoOf, $\hookrightarrow_r$, :Org),
(:p$_1$, :worksFor, _:b_c), (_:b_c, τ, :Comp),
(:p$_2$, :worksFor, :a), (:a, τ, :Org)}
$(G_{\text{ex}})_2 = (G_{\text{ex}})_1 \cup$
{(:p$_1$, τ, :Person), (:p$_2$, τ, :Person), (_:b_c, τ, :Org)}

2.3 Basic Graph Pattern Queries

A popular RDF query dialect consists of **basic graph pattern queries**, or **BGPQs**, in short. Let $\mathscr{V}$ be a set of variable symbols, disjoint from $\mathscr{I} \cup \mathscr{B} \cup \mathscr{L}$. A basic graph pattern (BGP) is a set of *triple patterns* (triples in short) belonging to $(\mathscr{I} \cup \mathscr{B} \cup \mathscr{V}) \times (\mathscr{I} \cup \mathscr{V}) \times (\mathscr{I} \cup \mathscr{B} \cup \mathscr{L} \cup \mathscr{V})$. For a BGP P, we denote by 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 $q(\bar{x}) \leftarrow P$, where P is a BGP (also denoted by $\text{body}(q)$), and $\bar{x} \subseteq \text{Var(P)}$ are the answer variables of q.

To ease the presentation, and without loss of generality, we consider BGPQs *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 BGPQs into **partially instantiated BGPQs** [12, 29]. Starting from a BGPQ q, partial instantiation may replace *some* variables with values from $\mathscr{I} \cup \mathscr{L} \cup \mathscr{B}$, as specified by a substitution σ. Due to σ, and in contrast with standard BGPQs, some answer variables of the resulting query q_σ can be bound:

Example 2.6 (Partially instantiated BGPQ). Consider the BGPQ asking for *who is working for which kind of company* $q(x, y) \leftarrow (x, \text{:worksFor}, z), (z, \tau, y), (y, \prec_{sc}, \text{:Comp})$ and the substitution $\sigma = \{x \mapsto \text{:p}_1\}$. The corresponding partially instantiated BGPQ is: $q(\text{:p}_1, y) \leftarrow (\text{:p}_1, \text{:worksFor}, z), (z, \tau, y), (y, \prec_{sc}, \text{:Comp})$. 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 BGPQ or a partially instantiated BPGQ.
The semantics of a BGPQ 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 φ from $\text{Val}(P)$ to $\text{Val}(G)$ such that for any triple $(s, p, o) \in P$, the triple $(\varphi(s), \varphi(p), \varphi(o))$ is in G, with φ 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 BGPQ q on an RDF graph G w.r.t. a set $\mathcal{R}$ of RDF entailment rules is: $q(G, \mathcal{R}) = \{\varphi(\bar{x}) \mid \varphi \textit{ homomorphism from } \text{body}(q) \textit{ to } G^{\mathcal{R}}\}$. If $\bar{x} = \emptyset$, q is a *Boolean* query, in which case q is false when $q(G, \mathcal{R}) = \emptyset$ and true when $q(G, \mathcal{R}) = \{\langle\rangle\}$, 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 $\text{body}(q)$ 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 BGPQ q from the preceding example. Its evaluation on G_{ex} is empty because G_{ex} has no explicit :worksFor assertion, while its answer set on G_{ex} w.r.t. $\mathcal{R}$ is $\{\langle\text{:p}_1, \text{:NatComp}\rangle\}$ 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 :NatComp.

The above notions and notations naturally extend to *unions* of (partially instantiated) BGPQs, or **UBGPQ**s 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

[1] In the notations $\mathcal{R}_c$ and $\mathcal{R}_a$, c and a respectively stand for "constraint triples" (called schema triples here) and "assertion triples" (data triples).

answer set to an incoming query q is obtained through query evaluation as $q(G^{\mathcal{R}})$.

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 $\mathcal{R}_a$ and $\mathcal{R}_c$.

(*i*) The first step reformulates a BGPQ q w.r.t. an RDFS ontology O and the set of rules $\mathcal{R}_c$ into a UBGPQ, say Q_c, which is guaranteed not to contain ontology triples. Intuitively, this step generates new BGPQs obtained from q by instantiating variables that query the ontology with all their possible bindings; for instance, y in a query triple $(y, \prec_{sc}, \text{:Comp})$ is bound to the IRIs of all explicit and implicit subclasses of :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, $q(G, \mathcal{R}_c) = Q_c(G)$.

(*ii*) The second step reformulates Q_c w.r.t. O and $\mathcal{R}_a$, and outputs a UBGPQ, say $Q_{c,a}$. This step, alone, is sound and complete w.r.t. $\mathcal{R}_a$ for query answering, i.e., for any graph G with ontology O, $Q_c(G, \mathcal{R}_a) = Q_{c,a}(G)$. Furthermore, a key property is that $q(G, \mathcal{R}) = Q_c(G, \mathcal{R}_a)$, i.e., only $\mathcal{R}_a$ 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 sucessive 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 $q(x, y) \leftarrow (x, \text{:worksFor}, z), (z, \tau, y), (y, \prec_{sc}, \text{:Comp})$ from the preceding example and the ontology O in Example 2.2. The first reformulation step instantiates the triple $(y, \prec_{sc}, \text{:Comp})$ 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 :worksFor according to its subproperties in O. It can be checked that $Q_{c,a}(G_{\text{ex}}) = q(G_{\text{ex}}, \mathcal{R}) = q(G_{\text{ex}}^{\mathcal{R}}) = \{\langle \text{:p}_1, \text{:NatComp} \rangle\}$, obtained here from the third BGPQ 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 $V(\bar{x})$:- $\psi(\bar{x})$ 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 *Emp(eID, name, dID), Dept(dID, cID, country), Salary(eID,amount)*, where *eID*, *dID* and *cID* are respectively identifiers for employees, departments and companies.

Consider the views V_1*(eID, name, country) :- Emp(eID, name, dID), Dept(dID, "IBM", country)* providing the names of IBM employees and where they work, and V_2*(eID, amount) :- Emp(eID, name, "R&D"), Salary(eID,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 $\mathcal{V}$; 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 $q(n, a)$:- $Emp(e, n, d)$, $Dept(d, c,$ *"France"*), *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 be equivalent to q (but just contained in q). In our example, $q_r(n, a)$:- $V_1(e, n, \text{"}France\text{"}), V_2(e, a)$ 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 ((U)CQs)*, *conjunctive views* (views V such that the associated query $V(\bar{x})$:- $\psi(\bar{x})$ 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 *Person(eID, name)* and *Contract(eID, dID, 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_1^D over the D schema shown in the figure (note that V_1^D hides the department from system $\mathcal{I}$); V_1^D provides the extension of V_1. Similarly, view V_2 may be defined as a query over some data source (or sources).

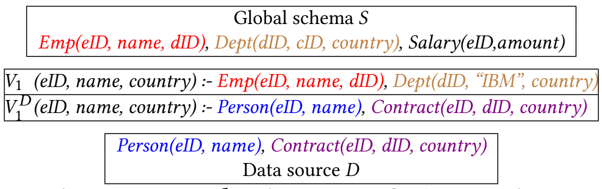

Figure 1: Example: view V_1 as a GLAV mapping.

Query rewriting is unchanged, whether the views are stored or defined by source queries. In the latter case, to obtain answers, a view-based rewriting needs to be *unfolded*, replacing every occurence of a view symbol V with the body of the source query defining that view. Executing the resulting query (potentially over different data sources) computes the answers. This integration setting, which considers views as intermediaries between sources and the integration schema, has been called "global-local-as-view" (GLAV) [26]. An association of a query q_1 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 $q_2 = V_1$ above, is commonly called a *GLAV mapping* (denoted $q_1(\bar{x}) \rightsquigarrow q_2(\bar{x})$).

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 $q_1(\bar{x}) \rightsquigarrow q_2(\bar{x})$, q_2 defines a single element of the global schema (hence $body(q_2)$ is restricted to a single atom if q_2 is a CQ, or a single triple pattern if q_2 is a BGPQ) and its variables are exactly $\bar{x}$. 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,*"John Doe"*,*"France"*⟩ is an answer to V_1^D above. Then, V_1 exposes this tuple in the global schema as: *Emp*(1,*"John Doe"*,x), *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 system* (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 is of the form $m = q_1(\bar{x}) \rightsquigarrow q_2(\bar{x})$ where q_1 and q_2 are two queries with the same answer variables, and q_2 is a BGPQ whose body contains only triples of the forms:

- $(\mathsf{s}, p, \mathsf{o})$ where $p \in \mathscr{I}_{\text{user}}$,
- (s, τ, C) where $C \in \mathscr{I}_{\text{user}}$.

The *body* of m is q_1 and its *head* is q_2. The *extension* of m is the set of tuples $\text{ext}(m) = \{V_m(\delta(v_1), \dots, \delta(v_n)) \mid \langle v_1, \dots, v_n\rangle \in q_1(D)\}$, where $q_1(D)$ is the answer set of q_1 on the data source D that m integrates and δ 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 (BGP) query q_2.

Example 3.2 (Mappings). Consider the two mappings:
m_1 with head $q_2(x) \leftarrow (x, \text{:ceoOf}, y), (y, \tau, \text{:NatComp})$ and
m_2 with head $q_2(x, y) \leftarrow (x, \text{:hiredBy}, y), (y, \tau, \text{:PubAdmin})$.
Suppose that the body of m_1 returns $\langle p^{D_1}\rangle$ as its results, and that the δ function maps the value $p_1^{D_1}$ from the data source D_1 to the IRI :p$_1$. Then, the extension of m_1 is: $\text{ext}(m_1) = \{V_{m_1}(\text{:p}_1)\}$. Further, suppose that the body of m_2 returns $\langle p_2^{D_2}, a^{D_2}\rangle$, and that δ maps the values $p_2^{D_2}, a^{D_2}$ from the data source D_2 to the IRIs :p$_2$, :a. Then, the extension of m_2 is: $\text{ext}(m_2) = \{V_{m_2}(\text{:p}_2, \text{:a})\}$.

Given a set of RIS mappings $\mathcal{M}$, the *extent* of $\mathcal{M}$ is the union of the mappings' extensions, i.e., $\mathcal{E} = \bigcup_{m \in \mathcal{M}}$ *ext(m)*, and we denote by Val($\mathcal{E}$) the set of values occurring in $\mathcal{E}$. 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 $\mathcal{M}$ of RIS mappings and an extent $\mathcal{E}$ of $\mathcal{M}$, the *RIS data triples* induced by $\mathcal{M}$ and $\mathcal{E}$ form an RDF graph defined as follows:

$$G_{\mathcal{E}}^{\mathcal{M}} = \bigcup_{m = q_1(\bar{x}) \rightsquigarrow q_2(\bar{x}) \in \mathcal{M}} \{\text{bgp2rdf}(\text{body}(q_2)_{[\bar{x} \leftarrow \bar{t}]})) \mid V_m(\bar{t}) \in \mathcal{E}\}$$

where

- $\text{body}(q_2)_{[\bar{x} \leftarrow \bar{t}]}$ is the BGP body(q_2) in which the answer variables $\bar{x}$ are bound to the values in the tuple $V_m(\bar{t})$, part of $\mathcal{E}$;
- 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 $\mathcal{M} = \{m_1, m_2\}$ and its extent $\mathcal{E} = \{V_{m_1}(\text{:p}_1), V_{m_2}(\text{:p}_2, \text{:a})\}$. The RIS data triples they lead to are:

$$G_{\mathcal{E}}^{\mathcal{M}} = \{(\text{:p}_1, \text{:ceoOf}, \_\text{:}b_c), (\_\text{:}b_c, \tau, \text{:NatComp}),\\ (\text{:p}_2, \text{:hiredBy}, \text{:a}), (\text{:a}, \tau, \text{:PubAdmin})\}$$

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 $V_m(\bar{t})$, thus from Val($\mathcal{E}$).

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

3.2 Query answering problem

The problem we consider is answering BGPQs 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 BGPQ q on a RIS $S = \langle O, \mathcal{R}, \mathcal{M}, \mathcal{E}\rangle$ is:

$\text{cert}(q, S) = \{\varphi(\bar{x}) \mid \varphi$ *homomorphism from* body(q) *to* $(O \cup G_{\mathcal{E}}^{\mathcal{M}})^{\mathcal{R}}\}$

where $\varphi(\bar{x})$ comprises only values from Val($\mathcal{E}$).

The certain answer set cert(q, S) is thus the subset of $q(O \cup G_{\mathcal{E}}^{\mathcal{M}}, \mathcal{R})$ 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 $\mathcal{R}$ of entailment rules shown in Table 3, and the set of mappings $\mathcal{M}$ together with the extent $\mathcal{E}$ from Example 3.4.

Let $q(x, y) \leftarrow (x, \text{:worksFor}, y), (y, \tau, \text{:Comp})$ be the query asking "who works for which company", while the query $q'(x) \leftarrow (x, \text{:worksFor}, y), (y, \tau, \text{:Comp})$ 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 $\emptyset$,

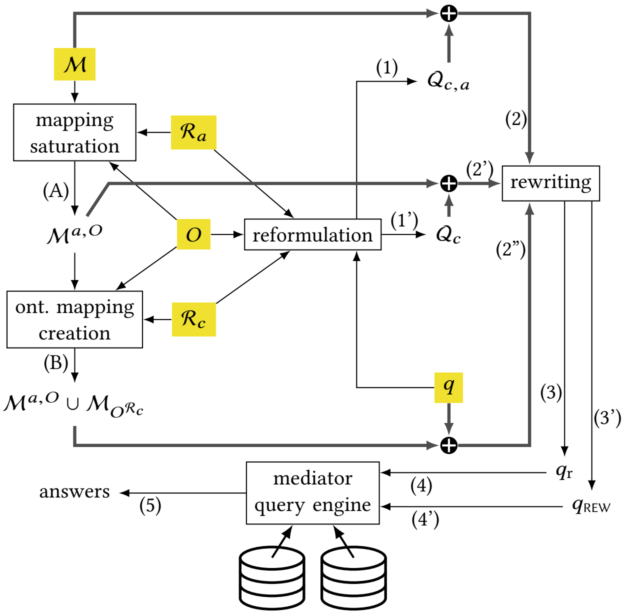

Figure 2: Outline of query answering strategies.

while the certain answer set of q' is $\{\langle:\mathrm{p}_1\rangle\}$. This answer results from the RIS data triples ($:\mathrm{p}_1$, :worksFor, $\_:b_c$), ($\_:b_c$, τ, :Comp), which are entailed from:

- the $G_{\mathcal{E}}^{\mathcal{M}}$ triples ($:\mathrm{p}_1$, :ceoOf, $\_:b_c$), ($\_:b_c$, τ, :NatComp), with the blank node $\_:b_c$ discussed in Example 3.4, and:
- either the O triples (:ceoOf, $\prec_{sp}$, :worksFor), (:ceoOf, $\hookrightarrow_r$, :Comp) together with the $\mathcal{R}$ rules `rdfs3`, `rdfs7`, or the O triples (:ceoOf, $\prec_{sp}$, :worksFor), (:NatComp, $\prec_{sc}$, :Comp) together with the $\mathcal{R}$ rules `rdfs3`, `rdfs9`.

The query q has no answer because it requires a value not available from the source: the company for which $:\mathrm{p}_1$ works; the RIS only knows the existence of such value through the blank node $\_:b_c$ begotten by bgp2rdf in its data triples. In contrast, q' allows finding out that $:\mathrm{p}_1$ works for (as CEO of) some (national) company, even though the mapping m_1 (the only one involving companies) does not expose the company IRI through the RIS.

The problem we study in the next section is:

Problem 1. *Given a RIS S, compute the certain answer set of a BGPQ q on S, i.e.,* cert(q, S).

4 QUERY ANSWERING IN A RIS

Since we adopt a mediator-style approach, the RIS data triples $G_{\mathcal{E}}^{\mathcal{M}}$ are not materialised, hence the saturation of $O \cup G_{\mathcal{E}}^{\mathcal{M}}$ 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 $\mathcal{R}$ and mappings $\mathcal{M}$. We present three query answering strategies, which differ in how the ontological 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 $\mathcal{R} = \mathcal{R}_c \cup \mathcal{R}_a$. 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 problem is in turn reduced to *relational view-based query answering*, by rewriting $Q_{c,a}$ using the RIS GLAV mappings $\mathcal{M}$ seen as LAV views (step (2)). This produces a relational rewriting q_r 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 $\mathcal{R}_c$ only (not $\mathcal{R} = \mathcal{R}_c \cup \mathcal{R}_a$ as above). The obtained reformulation Q_c thus yields the expected certain answer set when evaluated on the RIS data triples *saturated with* $\mathcal{R}_a$ (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 $\mathcal{R}_a$, 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 $\mathcal{R}_a$*, seen as LAV views. These saturated mappings, denoted $\mathcal{M}^{a,O}$, 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 $\mathcal{R}_a$. Then, the partially reformulated query Q_c is rewritten using $\mathcal{M}^{a,O}$ (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_r 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 $\mathcal{M}_{O^{\mathcal{R}_c}}$ (step (B)), comprising all the explicit and implicit RIS schema triples w.r.t. O and $\mathcal{R}$; since only $\mathcal{R}_c$ rules entail new schema triples (Table 3), $O^{\mathcal{R}}$ is actually equal to $O^{\mathcal{R}_c}$. 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 $\mathcal{M}^{a,O} \cup \mathcal{M}_{O^{\mathcal{R}_c}}$ 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 *bgp2ca* function transforms a BGP into a conjunction of atoms with ternary predicate T (standing for "triple") as follows: $bgp2ca(\{(\mathtt{s}_1, \mathtt{p}_1, \mathtt{o}_1), \ldots, (\mathtt{s}_n, \mathtt{p}_n, \mathtt{o}_n)\}) = T(\mathtt{s}_1, \mathtt{p}_1, \mathtt{o}_1) \wedge \cdots \wedge T(\mathtt{s}_n, \mathtt{p}_n, \mathtt{o}_n)$. The *bgpq2cq* function transforms a BGPQ $q(\bar{x}) \leftarrow body(q)$ into a CQ $q(\bar{x}) \leftarrow bgp2ca(body(q))$. Finally, the function *ubgpq2ucq* function transforms a UBGPQ $\bigcup_{i=1}^{n} q_i(\bar{x}_i)$ into a UCQ by applying the above *bgpq2cq* function to each of its q_i.

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 $\mathcal{R} = \mathcal{R}_c \cup \mathcal{R}_a$ 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 $\mathcal{R}$, 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 $G_{\mathcal{E}}^{\mathcal{M}}$ are not materialized.

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

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

The proof of this and our following claims can be found in [13]. Recall that the RIS data triples are defined from the mappings $\mathcal{M}$ by, for every mapping $m = q_1(\bar{x}) \rightsquigarrow q_2(\bar{x}) \in \mathcal{M}$, (*i*) evaluating the mapping body $q_1(\bar{x})$ on the data source to produce its extension $\mathrm{ext}(m) \in \mathcal{E}$, and then (*ii*) instantiating the mapping head $q_2(\bar{x})$ 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 $m = q_1(\bar{x}) \rightsquigarrow q_2(\bar{x})$ be a mapping. Its corresponding relational LAV view is: $V_m(\bar{x}) \leftarrow bgp2ca(\mathrm{body}(q_2))$.

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

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

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

Based on the above Lemma 4.1, treating mappings and their extent as relational LAV views and their extent, and seeing (U)BGPQs 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 $S = \langle O, \mathcal{R}, \mathcal{M}, \mathcal{E} \rangle$ be a RIS and q be a BGPQ. Let $Q_{c,a}$ be the reformulation of q w.r.t. O and $\mathcal{R}$ using [12]. Then:*

$$\mathrm{cert}(q, S) = \mathrm{cert}(ubgpq2ucq(Q_{c,a}), \mathrm{Views}(\mathcal{M}), \mathcal{E})$$

where $\mathrm{cert}(ubgpq2ucq(Q_{c,a}), \mathrm{Views}(\mathcal{M}), \mathcal{E})$ denotes the certain answer set of $ubgpq2ucq(Q_{c,a})$ over $\mathrm{Views}(\mathcal{M})$ and $\mathcal{E}$.

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 $q(x, y) \leftarrow (x, y, z), (z, \tau, t), (y, \prec_{sp}, \text{:worksFor}), (t, \prec_{sc}, \text{:Comp}), (x, \text{:worksFor}, a), (a, \tau, \text{:PubAdmin})$ 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: $q_r(x, \text{:ceoOf}) \leftarrow V_{m_1}(x), V_{m_2}(x, y)$, obtained from the second CQ in the above union. This becomes clear when the views are replaced by their bodies: $q(x, \text{:ceoOf}) \leftarrow T(x, \text{:ceoOf}, y_1), T(y_1, \tau, \text{:NatComp}), T(x, \text{:hiredBy}, y_2), T(y_2, \tau, \text{:PubAdmin})$. 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., $\mathrm{cert}(q, S) = \emptyset$, because the extent of the mappings, hence of the views, is: $\mathcal{E} = \{V_{m_1}(\text{:p}_1), V_{m_2}(\text{:p}_2, \text{:a})\}$. However, if we add $V_{m_2}(\text{:p}_1, \text{:a})$ to $\mathcal{E}$, then $\mathrm{cert}(q, S) = \{\langle \text{:p}_1, \text{:ceoOf} \rangle\}$.

$$\begin{aligned}
Q_{c,a} = \; & q(x, \text{:ceoOf}) \leftarrow T(x, \text{:ceoOf}, z), T(z, \tau, \text{:NatComp}), \\
& \qquad T(x, \text{:worksFor}, a), T(a, \tau, \text{:PubAdmin}) \\
\cup \; & q(x, \text{:ceoOf}) \leftarrow T(x, \text{:ceoOf}, z), T(z, \tau, \text{:NatComp}), \\
& \qquad T(x, \text{:hiredBy}, a), T(a, \tau, \text{:PubAdmin}) \\
\cup \; & q(x, \text{:ceoOf}) \leftarrow T(x, \text{:ceoOf}, z), T(z, \tau, \text{:NatComp}), \\
& \qquad T(x, \text{:ceoOf}, a), T(a, \tau, \text{:PubAdmin}) \\
\cup \; & q(x, \text{:hiredBy}) \leftarrow T(x, \text{:hiredBy}, z), T(z, \tau, \text{:NatComp}), \\
& \qquad T(x, \text{:worksFor}, a), T(a, \tau, \text{:PubAdmin}) \\
\cup \; & q(x, \text{:hiredBy}) \leftarrow T(x, \text{:hiredBy}, z), T(z, \tau, \text{:NatComp}), \\
& \qquad T(x, \text{:hiredBy}, a), T(a, \tau, \text{:PubAdmin}) \\
\cup \; & q(x, \text{:hiredBy}) \leftarrow T(x, \text{:hiredBy}, z), T(z, \tau, \text{:NatComp}), \\
& \qquad T(x, \text{:ceoOf}, a), T(a, \tau, \text{:PubAdmin})
\end{aligned}$$

Figure 3: Sample reformulation for Example 4.5.

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 $\mathcal{R} = \mathcal{R}_c \cup \mathcal{R}_a$ 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 UBGPQ 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 $\mathcal{R}$ (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 $\mathcal{R}_c$. Formally:

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

$$\mathrm{cert}(q, S) = \mathrm{cert}(Q_c, \langle O, \mathcal{R}_a, \mathcal{M}, \mathcal{E} \rangle)$$

In other words, the desired answer set could be obtained by evaluating Q_c on the RIS data triples $G_{\mathcal{E}}^{\mathcal{M}}$ saturated by $\mathcal{R}_a$. 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 $\mathcal{R}$ on these "virtual" data triples, we rely on *BGPQ saturation* [25]: given a BGPQ q, O and $\mathcal{R}$, the *saturation* $q^{\mathcal{R},O}$ *is q augmented with* all *the triples q implicitly asks for, given the ontology O and the rules $\mathcal{R}$*. BGPQ saturation is exemplified below:

Example 4.7 (BGPQ saturation). Consider the ontology O of G_{ex} and the query $q(x) \leftarrow (x, \text{:hiredBy}, y), (y, \tau, \text{:NatComp})$ asking who has been hired by a national company. Its saturation w.r.t. $\mathcal{R}_a, O$ is: $q^{\mathcal{R}_a,O}(x) \leftarrow \mathrm{body}(q), (x, \text{:worksFor}, y), (x, \tau, \text{:Person}), (y, \tau, \text{:Comp}), (y, \tau, \text{:Org})$.

We use BGPQ saturation to saturate the RIS mapping heads w.r.t. $\mathcal{R}_a, O$, so that the *saturated* mappings together with $\mathcal{E}$ model the *saturated RIS data triples w.r.t. $\mathcal{R}_a, O$*. To compute $q^{\mathcal{R}_a,O}$ we (1) saturate $\mathrm{body}(q) \cup O$ 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 $\mathcal{M}$ of RIS mappings w.r.t. entailment rules $\mathcal{R}_a$ and ontology O is:

$$\mathcal{M}^{a,O} = \bigcup_{m \in \mathcal{M}} \{q_1(\bar{x}) \rightsquigarrow q_2^{\mathcal{R}_a,O}(\bar{x}) \mid m = q_1(\bar{x}) \rightsquigarrow q_2(\bar{x})\}$$

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 $\mathcal{M}^{a,O}$ are (added implicit triples are in blue):

$$\begin{aligned}
m_1: \quad & q_2^{\mathcal{R}_a,O}(x) \leftarrow && (x, \text{:ceoOf}, y), (y, \tau, \text{:NatComp}) \\
& && (x, \text{:worksFor}, y), (y, \tau, \text{:Comp}) \\
& && (x, \tau \text{:Person}), (y, \tau, \text{:Org}) \\
m_2: \quad & q_2^{\mathcal{R}_a,O}(x, y) \leftarrow && (x, \text{:hiredBy}, y), (y, \tau, \text{:PubAdmin}) \\
& && (x, \text{:worksFor}, y), (y, \tau, \text{:Org}) \\
& && (x, \tau, \text{:Person})
\end{aligned}$$

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

Lemma 4.10. *Let $S = \langle O, \mathcal{R}, \mathcal{M}, \mathcal{E}\rangle$ be a RIS, q be a BGPQ and Q_c its reformulation w.r.t. $O, \mathcal{R}_c$ [12]. Then:*

$$\text{cert}(q, S) = \text{cert}(Q_c, \langle \emptyset, \emptyset, \mathcal{M}^{a,O}, \mathcal{E}\rangle)$$

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 $S = \langle O, \mathcal{R}, \mathcal{M}, \mathcal{E}\rangle$ be a RIS, q be a BGPQ and Q_c its reformulation w.r.t. $O, \mathcal{R}_c$.Then:*

$$\text{cert}(q, S) = \text{cert}(ubgpq2ucq(Q_c), \text{Views}(\mathcal{M}^{a,O}), \mathcal{E})$$

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. $O, \mathcal{R}_c$, seen as a UCQ, is:

$$\begin{aligned}
& q(x, \text{:ceoOf}) \leftarrow && T(x, \text{:ceoOf}, z), T(z, \tau, \text{:NatComp}), \\
& && T(x, \text{:worksFor}, a), T(a, \tau, \text{:PubAdmin}) \\
\cup\ & q(x, \text{:hiredBy}) \leftarrow && T(x, \text{:hiredBy}, z), T(z, \tau, \text{:NatComp}), \\
& && T(x, \text{:worksFor}, a), T(a, \tau, \text{:PubAdmin})
\end{aligned}$$

This reformulation is therefore rewritten using the RIS views as: $q_r(x, \text{:ceoOf}) \leftarrow V_{m_1}(x), V_{m_2}(x, y)$. 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 $\mathcal{M}^{a,O}$ 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:

$$\mathcal{M}_{O^c} = \bigcup_{x \in \{\prec_{sc}, \prec_{sp}, \hookleftarrow_d, \hookrightarrow_r\}} \{m_x \mid m_x = q_1(\mathsf{s}, \mathsf{o}) \rightsquigarrow q_2(\mathsf{s}, \mathsf{o})\}$$

with $q_2(\mathsf{s}, \mathsf{o}) \leftarrow (\mathsf{s}, x, \mathsf{o})$. The *extension* of an ontology mapping m_x is $\text{ext}(m_x) = \{V_{m_x}(\mathsf{s}, \mathsf{o}) \mid (\mathsf{s}, x, \mathsf{o}) \in O^{\mathcal{R}_c}\}$. The *extent* of $\mathcal{M}_{O^c}$ is denoted $\mathcal{E}_{O^c}$.

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

Lemma 4.14. *Let $S = \langle O, \mathcal{R}, \mathcal{M}, \mathcal{E}\rangle$ be a RIS and q be a BGPQ. Then:*

$$\text{cert}(q, S) = \text{cert}(q, \langle O, \mathcal{R}_a, \mathcal{M}_{O^c} \cup \mathcal{M}, \mathcal{E}_{O^c} \cup \mathcal{E}\rangle)$$

This lemma effectively "pushes" $\mathcal{R}_c$ reasoning in the set of mappings (to which we add $\mathcal{M}_{O^c}$) and the extent (to which we add $\mathcal{E}_{O^c}$). Next, we rely (as we did for REW-CA) on mappings saturation with $O, \mathcal{R}_a$ to also push $\mathcal{R}_a$ reasoning in the mappings, leading to:

Lemma 4.15. *Let $S = \langle O, \mathcal{R}, \mathcal{M}, \mathcal{E}\rangle$ be a RIS and q be a BGPQ. Then:*

$$\text{cert}(q, S) = \text{cert}(q, \langle \emptyset, \emptyset, \mathcal{M}_{O^c} \cup \mathcal{M}^{a,O}, \mathcal{E}_{O^c} \cup \mathcal{E}\rangle)$$

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

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

$$\text{cert}(q, S) = \text{cert}(bgpq2cq(q), \text{Views}(\mathcal{M}_{O^c} \cup \mathcal{M}^{a,O}), \mathcal{E}_{O^c} \cup \mathcal{E})$$

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

$$\begin{aligned}
q(x, y) \leftarrow & T(x, y, z), T(z, \tau, t), T(y, \prec_{sp}, \text{:worksFor}), \\
& T(t, \prec_{sc}, \text{:Comp}), T(x, \text{:worksFor}, a), \\
& T(a, \tau, \text{:PubAdmin})
\end{aligned}$$

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 $\mathcal{E}$ also contains $V_{m_2}(\text{:p}_1, \text{:a})$, as we did in

$$\begin{aligned}
& q(x, \text{:ceoOf}) \leftarrow && V_{m_1}(x), V_{m_{\prec_{sp}}}(\text{:ceoOf}, \text{:worksFor}), \\
& && V_{m_{\prec_{sc}}}(\text{:NatComp}, \text{:Comp}), V_{m_2}(x, a) \\
\cup\ & q(x, \text{:ceoOf}) \leftarrow && V_{m_1}(x), V_{m_{\prec_{sp}}}(\text{:ceoOf}, \text{:worksFor}), \\
& && V_{m_{\prec_{sc}}}(\text{:Comp}, \text{:Comp}), V_{m_2}(x, a) \\
\cup\ & q(x, \text{:ceoOf}) \leftarrow && V_{m_1}(x), V_{m_{\prec_{sp}}}(\text{:ceoOf}, \text{:worksFor}), \\
& && V_{m_{\prec_{sc}}}(\text{:Org}, \text{:Comp}), V_{m_2}(x, a) \\
\cup\ & q(x, \text{:worksFor}) \leftarrow && V_{m_1}(x), V_{m_{\prec_{sp}}}(\text{:worksFor}, \text{:worksFor}), \\
& && V_{m_{\prec_{sc}}}(\text{:NatComp}, \text{:Comp}), V_{m_2}(x, a) \\
\cup\ & q(x, \text{:worksFor}) \leftarrow && V_{m_1}(x), V_{m_{\prec_{sp}}}(\text{:worksFor}, \text{:worksFor}), \\
& && V_{m_{\prec_{sc}}}(\text{:Comp}, \text{:Comp}), V_{m_2}(x, a) \\
\cup\ & q(x, \text{:worksFor}) \leftarrow && V_{m_1}(x), V_{m_{\prec_{sp}}}(\text{:worksFor}, \text{:worksFor}), \\
& && V_{m_{\prec_{sc}}}(\text{:Org}, \text{:Comp}), V_{m_2}(x, a) \\
\cup\ & q(x, \text{:hiredBy}) \leftarrow && V_{m_2}(x, z), V_{m_{\prec_{sp}}}(\text{:hiredBy}, \text{:worksFor}), \\
& && V_{m_{\prec_{sc}}}(\text{:PubAdmin}, \text{:Comp}), V_{m_2}(x, a) \\
\cup\ & q(x, \text{:hiredBy}) \leftarrow && V_{m_2}(x, z), V_{m_{\prec_{sp}}}(\text{:hiredBy}, \text{:worksFor}), \\
& && V_{m_{\prec_{sc}}}(\text{:Org}, \text{:Comp}), V_{m_2}(x, a) \\
\cup\ & q(x, \text{:worksFor}) \leftarrow && V_{m_2}(x, z), V_{m_{\prec_{sp}}}(\text{:worksFor}, \text{:worksFor}), \\
& && V_{m_{\prec_{sc}}}(\text{:PubAdmin}, \text{:Comp}), V_{m_2}(x, a) \\
\cup\ & q(x, \text{:worksFor}) \leftarrow && V_{m_2}(x, z), V_{m_{\prec_{sp}}}(\text{:worksFor}, \text{:worksFor}), \\
& && V_{m_{\prec_{sc}}}(\text{:Org}, \text{:Comp}), V_{m_2}(x, a) \\
\cup \textstyle\bigcup_{\mathsf{r} \in \{\prec_{sc}, \prec_{sp}, \hookleftarrow_d, \hookrightarrow_r\}} & q(x, \mathsf{r}) \leftarrow && V_{m_\mathsf{r}}(x, z), V_{m_2}(v, z), \\
& && V_{m_{\prec_{sp}}}(\mathsf{r}, \text{:worksFor}), \\
& && V_{m_{\prec_{sc}}}(\text{:PubAdmin}, \text{:Comp}), \\
& && V_{m_2}(x, a) \\
\cup\ & q(x, \mathsf{r}) \leftarrow && V_{m_\mathsf{r}}(x, z), V_{m_2}(v, z), \\
& && V_{m_{\prec_{sp}}}(\mathsf{r}, \text{:worksFor}), \\
& && V_{m_{\prec_{sc}}}(\text{:Org}, \text{:Comp}), \\
& && V_{m_2}(x, a)
\end{aligned}$$

Figure 4: Sample rewriting for Example 4.17.

Example 4.5, we obtain again cert$(q, S) = \{\langle :\text{p}_1, :\text{ceoOf}\rangle\}$, which results from the evaluation of the first CQ in the UCQ rewriting; the other CQs yield empty results because some required $\prec_{sc}$ or $\prec_{sp}$ 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, REW considers the additional set $\mathcal{M}_{O^c}$ of ontology mappings. Hence, for queries over the ontology, i.e., featuring in a property position $\prec_{sc}$, $\prec_{sp}$, $\hookleftarrow_d$, $\hookrightarrow_r$, or a variable, a REW rewriting is larger than a REW-CA or REW-C rewriting and, to be answered, requires the additional ontology source. In contrast, REW-CA and REW-C yield logically equivalent rewritings; we minimize them both to avoid possible redundancies, thus they become identical (up to variable renaming). Hence, REW-CA and 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 $\mathcal{R}$. 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 OntoSQL[2], 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 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 $V_m(\bar{x}) \leftarrow$ *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 $V_m(\bar{x}) \rightarrow$ *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_1 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 generator[3] to build databases consisting of 10 relations named producer, product, offer, review 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 $\mathcal{M}_1, \mathcal{M}_2$ of 307, respectively, 3863 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: $S_1 = \langle O_1, \mathcal{R}, \mathcal{M}_1, \mathcal{E}_1\rangle$ and $S_2 = \langle O_2, \mathcal{R}, \mathcal{M}_2, \mathcal{E}_2\rangle$, where $\mathcal{E}_i$ for i in $\{1, 2\}$ are the extents resulting from DS_i and $\mathcal{M}_i$.

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 $DS_{j,1}, DS_{j,2}$; the relational sources $DS_{r,1}, DS_{r_2}$ store the remaining (relational) data. Conceptually, for i in $\{1, 2\}$, the extension based on $DS_{r,i}$ and extension based on $DS_{j,i}$ form a partition of $\mathcal{E}_i$. We devise a set of **JSON-to-RDF mappings** to expose $DS_{j,1}$ and $DS_{j,2}$ into RDF, and denote $\mathcal{M}_3$ the set of mappings exposing $DS_{r,1}$ and $DS_{j,1}$, together, as an RDF graph; similarly, the mappings $\mathcal{M}_4$ expose $DS_{r,2}$ and $DS_{j,2}$ as RDF. Our last two RIS are thus: $S_3 = \langle O_1, \mathcal{R}, \mathcal{M}_3, \mathcal{E}_3\rangle$ and $S_4 = \langle O_2, \mathcal{R}, \mathcal{M}_4, \mathcal{E}_4\rangle$, where $\mathcal{E}_3$ is the extent of $\mathcal{M}_3$ based on $DS_{r,1}$ and $DS_{j,1}$, while $\mathcal{E}_4$ is the extent of $\mathcal{M}_4$ based on $DS_{r,2}$ and $DS_{j,2}$. *The RIS data and ontology triples of S_1 and S_3 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_4.

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 $330 \cdot 10^3$ results in S_1 and S_3 and 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), a capability which most

[2] https://ontosql.inria.fr

[3] https://downloads.sourceforge.net/project/bsbmtools/bsbmtools/bsbmtools-0.2

RIS		Q01	Q01a	Q01b	Q02	Q02a	Q02b	Q02c	Q03	Q04	Q07	Q07a	Q09	Q10	Q13
all	N_{TRI}	5	5	5	6	6	6	6	5	2	3	3	1	3	4
S_1, S_3	$\lvert Q_{c,a} \rvert$	7	21	175	21	49	147	1225	525	1	5	19	7	670	28
S_1, S_3	N_{ANS}	1272	4376	22738	16	56	174	1342	19	91	2	3	5617	9	13190
S_2, S_4	$\lvert Q_{c,a} \rvert$	21	175	1407	63	147	525	1225	4375	1	5	19	7	9350	84
S_2, S_4	N_{ANS}	15514	111793	863729	124	598	1058	1570	5	4487	2	3	299902	10	167760

RIS		Q13a	Q13b	Q14	Q16	Q19	Q19a	Q20	Q20a	Q20b	Q20c	Q21	Q22	Q22a	Q23
all	N_{TRI}	4	4	3	4	9	9	11	11	11	11	3	4	4	7
S_1, S_3	$\lvert Q_{c,a} \rvert$	84	700	1	25	63	147	21	63	525	1225	670	2	40	192
S_1, S_3	N_{ANS}	43157	330142	56200	8114	2015	3515	0	236	2312	7564	1085	28	434	25803
S_2, S_4	$\lvert Q_{c,a} \rvert$	5628	5628	1	201	525	1225	63	525	1225	4221	9350	40	520	192
S_2, S_4	N_{ANS}	4416946	10049829	2998948	249004	39826	60834	904	7818	10486	51988	37176	1528	18588	1329887

Table 4: Characteristics of the queries used in our experiments.

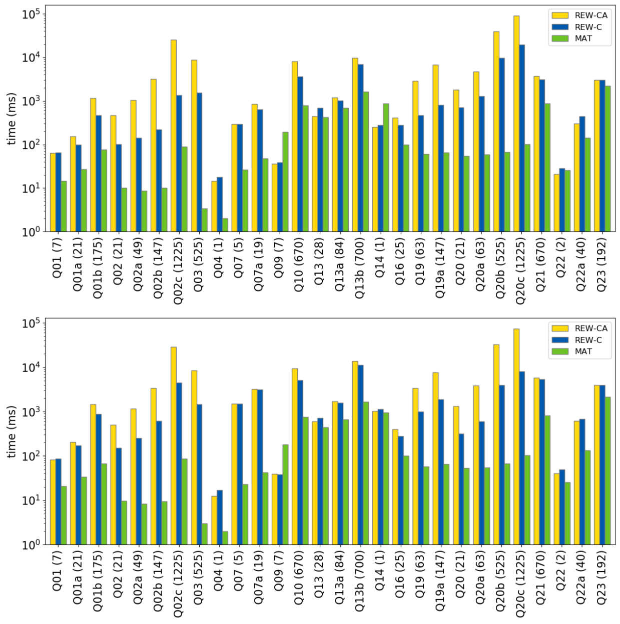

Figure 5: Query answering times on the smaller RIS S_1 (top, relational sources) and S_3 (bottom, heterogeneous sources).

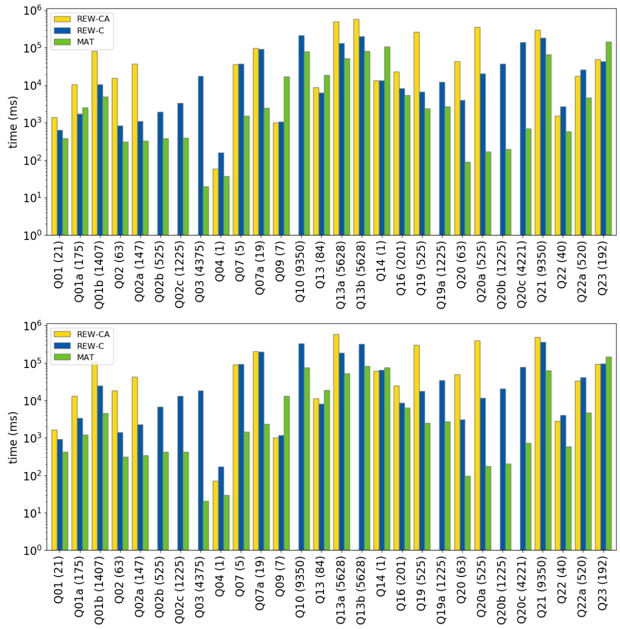

Figure 6: Query answering times on the larger RIS S_2 (top, relational sources) and S_4 (bottom, heterogeneous sources).

competitor systems lack (see Section 6). Table 4 reports three query properties impacting query answering performance: the number of induced triples (N_{TRI}), the number of BGPQ 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_{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 *query families* denoted Q_X, Q_{Xa} 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. *On queries (also) over the ontology*, as explained in Section 4.3, we noted that the size of the rewriting produced by REW is larger (by a multiplicative factor of 29 to 74 in S_1 and S_3, and of 33 to 969 in S_2 and S_4) than the rewritings of the two other strategies, which led to an explosion of the time spent minimizing the rewriting, and made REW overall unfeasible; the details of these tests can be found online[4]. *On queries that do not carry over the ontology*, REW produces the same rewritings as the other methods. Thus, we do not report further REW performance below.

Query answering time comparison Figure 5 depicts the query answering times, on the smaller RIS, of REW-CA, REW-C and MAT. *The size of (number of BGPQs in) the reformulation of each query w.r.t. $\mathcal{R}$, $|Q_{c,a}|$ appears in parentheses after the query name, in the labels along the x axis.* Given that S_1, S_3 have the same RIS data triples, the MAT strategy coincides among these two RIS. Figure 6 shows the corresponding times for the largest RIS S_2 and S_4; 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 MAT *is the fastest in most cases*, since it has no reasoning work to do at query answering time. However, it required, for S_1, S_3, $1.2 \cdot 10^5$ ms to build the materialization and $1.49 \cdot 10^5$ ms more to saturate it, whereas for S_2, S_4, these times are 14h46 ($5.31 \cdot 10^7$ ms), respectively, 1h28 ($5.28 \cdot 10^6$

[4]Experiment web site: https://gitlab.inria.fr/mburon/org/blob/master/projects/het2onto-benchmark/bsbm/

ms). Not only these are *orders of magnitude more than all query answering times*; recall also that materializing $G_{\mathcal{E}}^{\mathcal{M}}$ requires maintaining it when the underlying data changes, and its saturation $(G_{\mathcal{E}}^{\mathcal{M}} \cup O)^{\mathcal{R}}$ needs a second level of maintenance. Thus, MAT is not practical when data sources change. We were surprised to see REW-C and REW-CA somehow faster than MAT for queries Q_{09} and Q_{14}. 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. REW-C and 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 REW-C is faster or takes as long as 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 REW-C, the reformulations w.r.t. $\mathcal{R}_c$ 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 REW-CA the reformulation sizes are much larger. REW-C is most often faster than REW-CA, by up to two orders of magnitude e.g., for Q_{02a}, Q_{19} and Q_{20a} on S_2, the latter two on S_4 etc. One order of magnitude speed-up is noticeable even on the smaller RIS S_1, S_3 (Figure 5) for Q_{02a}. As a consequence, REW-C completes successfully in all scenarios we study, whereas 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 REW-C does, avoids the heavy performance penalties that from which REW-CA and REW 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-Lite$_{\mathcal{R}}$ underpinning the OWL 2 QL profile of the W3C ontological language OWL 2). Mature DL-based systems include Mastro[5] [17] and Ontop[6] [16, 43]. Another notable OBDA system, namely Ultrawrap^{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 $m_1 = q_1(\bar{x}) \rightsquigarrow q_2(\bar{x})$ with head $q_2(x) \leftarrow (x, \text{:ceoOf}, y), (y, \tau, \text{:NatComp})$ 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 $m_{1_1} = q_1(\bar{x}) \rightsquigarrow q_{2_1}(\bar{x})$ and $m_{1_2} = q_1(\bar{x}) \rightsquigarrow q_{2_2}(\bar{x})$, with respective head $q_{2_1}(\bar{x}) \leftarrow (x, \text{:ceoOf}, f(x))$ and $q_{2_2}(\bar{x}) \leftarrow (f(x), \tau, \text{:NatComp})$. 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 sceSkolemnario 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.

[5] http://obdasystems.com/mastro/

[6] https://ontop.inf.unibz.it/

It can be seen as a generalization to GLAV mappings of the $\mathcal{T}$-mapping technique introduced in [43] (and further developed in [44]) to optimize query rewriting in a classical OBDA context. The $\mathcal{T}$-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 $m = q_1(x) \rightsquigarrow q_2(x) \leftarrow C(x)$ with C a class, and a DL constraint specifying that C is a subclass of D, a new mapping $m' = q_1(x) \rightsquigarrow q_2'(x) \leftarrow D(x)$ is created by composing m and the DL constraint. On this example, we would saturate the head of m into $q_2(x) \leftarrow C(x) \wedge D(x)$, 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 $m_1 = q_1(\bar{x}) \rightsquigarrow q_2(\bar{x})$ with head $q_2(x) \leftarrow (x, \text{:ceoOf}, y), (y, \tau, \text{:NatComp})$; given the entailment rule `rdfs9` and the ontological triple $(\text{:NatComp}, \prec_{sc}, \text{:Comp})$, the saturation adds the triple $(y, \tau, \text{:Comp})$ to the body of q_2; creating instead a new mapping of the form $m_1' = q_1(\bar{x}) \rightsquigarrow q_2'(\bar{x})$ with head $q_2'(x) \leftarrow (y, \tau, \text{:Comp})$ would be unsatisfactory as y in m_1' should correspond to the same object as y in m_1.
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

[1] Nada Abdallah, François Goasdoué, and Marie-Christine Rousset. 2009. DL-LITER in the Light of Propositional Logic for Decentralized Data Management. In *IJCAI*.
[2] Serge Abiteboul and Oliver M. Duschka. 1998. Complexity of Answering Queries Using Materialized Views. ACM Press.
[3] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Addison-Wesley.
[4] Rana Alotaibi, Damian Bursztyn, Alin Deutsch, Ioana Manolescu, and Stamatis Zampetakis. 2019. Towards Scalable Hybrid Stores: Constraint-Based Rewriting to the Rescue. In *SIGMOD*. https://hal.inria.fr/hal-02070827
[5] Bernd Amann, Catriel Beeri, Irini Fundulaki, and Michel Scholl. 2002. Querying XML Sources Using an Ontology-Based Mediator. In *CoopIS*. Springer Berlin Heidelberg.
[6] Bernd Amann and Irini Fundulaki. 1999. Integrating Ontologies and Thesauri to Build RDF Schemas. In *ECDL*.
[7] Bernd Amann, Irini Fundulaki, and Michel Scholl. 2000. Integrating ontologies and thesauri for RDF schema creation and metadata querying. *Int. J. on Digital Libraries* 3, 3 (2000).
[8] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider (Eds.). 2003. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press.
[9] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, Swan Rocher, and Clément Sipieter. 2015. Graal: A Toolkit for Query Answering with Existential Rules. In *RuleML*.
[10] Raphaël Bonaque, Tien Duc Cao, Bogdan Cautis, François Goasdoué, Javier Letelier, Ioana Manolescu, Oscar Mendoza, Swen Ribeiro, Xavier Tannier, and Michaël Thomazo. 2016. Mixed-instance querying: a lightweight integration architecture for data journalism. In *VLDB*. https://hal.inria.fr/hal-01321201
[11] Elena Botoeva, Diego Calvanese, Benjamin Cogrel, Julien Corman, and Guohui Xiao. 2018. A Generalized Framework for Ontology-Based Data Access. In *AI*IA*.
[12] Maxime Buron, François Goasdoué, Ioana Manolescu, and Marie-Laure Mugnier. 2019. Reformulation-Based Query Answering for RDF Graphs with RDFS Ontologies. In *ESWC*.
[13] Maxime Buron, François Goasdoué, Ioana Manolescu, and Marie-Laure Mugnier. 2018. Rewriting-Based Query Answering for Semantic Data Integration Systems. In *BDA (informal publication only)*. https://hal.archives-ouvertes.fr/hal-01927282
[14] Damian Bursztyn, François Goasdoué, and Ioana Manolescu. 2015. Optimizing Reformulation-based Query Answering in RDF. In *EDBT*.
[15] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. 2009. A general Datalog-based framework for tractable query answering over ontologies. In *PODS*.
[16] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. 2017. Ontop: Answering SPARQL queries over relational databases. *Semantic Web* 8, 3 (2017).
[17] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. 2011. The MASTRO system for ontology-based data access. *Semantic Web* 2, 1 (2011).
[18] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, and Marco Ruzzi. 2009. Using OWL in Data Integration. In *Semantic Web Information Management – A Model-Based Perspective.* Springer.
[19] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. 2012. Query Processing under GLAV Mappings for Relational and Graph Databases. *PVLDB* 6, 2 (2012).
[20] Vassilis Christophides, Martin Doerr, and Irini Fundulaki. 1997. A Semantic Network Approach to Semi-Structured Documents Repositories. In *ECDL*.
[21] Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. 2018. Using Ontologies for Semantic Data Integration. In *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years.* Vol. 31. Springer, Cham.
[22] A. Deutsch and V. Tannen. 2003. MARS: A System for Publishing XML from Mixed and Redundant Storage.. In *VLDB*.
[23] AnHai Doan, Alon Halevy, and Zachary G. Ives. 2012. *Principles of Data Integration.* Morgan Kaufmann, Waltham, MA.
[24] J. Duggan, A. J. Elmore, M. Stonebraker, M. Balazinska, B. Howe, J. Kepner, S. Madden, D. Maier, T. Mattson, and S. B. Zdonik. 2015. The BigDAWG Polystore System. *SIGMOD* 44, 2 (2015).
[25] Sara El Hassad, François Goasdoué, and Hélène Jaudoin. 2017. Learning Commonalities in SPARQL. In *ISWC*.
[26] Marc Friedman, Alon Y. Levy, and Todd D. Millstein. 1999. Navigational Plans for Data Integration. In *I3 workshop@IJCAI*.
[27] Hector Garcia-Molina, Yannis Papakonstantinou, Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey D. Ullman, Vasilis Vassalos, and Jennifer Widom. 1997. The TSIMMIS Approach to Mediation: Data Models and Languages. *J. Intell. Inf. Syst.* 8, 2 (1997).
[28] François Goasdoué, Véronique Lattès, and Marie-Christine Rousset. 2000. The Use of CARIN Language and Algorithms for Information Integration: The PICSEL System. *Int. J. Cooperative Inf. Syst.* 9, 4 (2000).
[29] François Goasdoué, Ioana Manolescu, and Alexandra Roatis. 2013. Efficient query answering against dynamic RDF databases. In *EDBT*.
[30] François Goasdoué and Marie-Christine Rousset. 2004. Answering queries using views: A KRDB perspective for the semantic Web. *ACM TOIT* 4, 3 (2004).
[31] Alon Y. Halevy. 2001. Answering Queries Using Views: A Survey. *The VLDB Journal* 10, 4 (Dec. 2001).
[32] Dag Hovland, Roman Kontchakov, Martin G. Skjæveland, Arild Waaler, and Michael Zakharyaschev. 2017. Ontology-Based Data Access to Slegge. In *ISWC*.
[33] Roman Kontchakov, Martin Rezk, Mariano Rodríguez-Muro, Guohui Xiao, and Michael Zakharyaschev. 2014. Answering SPARQL Queries over Databases under OWL 2 QL Entailment Regime. In *ISWC*.
[34] Davide Lanti, Guohui Xiao, and Diego Calvanese. 2017. Cost-Driven Ontology-Based Data Access. In *ISWC*.
[35] Maurizio Lenzerini. 2002. Data Integration: A Theoretical Perspective. In *PODS*.
[36] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. 1996. Querying Heterogeneous Information Sources Using Source Descriptions. In *VLDB*.
[37] Alon Y. Levy, Divesh Srivastava, and Thomas Kirk. 1995. Data Model and Query Evaluation in Global Information Systems. *J. Intell. Inf. Syst.* 5, 2 (1995).
[38] Ioana Manolescu, Daniela Florescu, and Donald Kossmann. 2001. Answering XML Queries on Heterogeneous Data Sources. In *VLDB*.
[39] Marie-Laure Mugnier. 2011. Ontological Query Answering with Existential Rules. In *RR*.
[40] Floriana Di Pinto, Domenico Lembo, Maurizio Lenzerini, Riccardo Mancini, Antonella Poggi, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. 2013. Optimizing query rewriting in ontology-based data access. In *EDBT*.
[41] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. 2008. Linking Data to Ontologies. *J. Data Semantics* 10 (2008).
[42] Rachel Pottinger and Alon Y. Halevy. 2001. MiniCon: A scalable algorithm for answering queries using views. *VLDB J.* 10 (2001).
[43] Mariano Rodriguez-Muro, Roman Kontchakov, and Michael Zakharyaschev. 2013. Ontology-Based Data Access: Ontop of Databases. In *ISWC*.
[44] Juan F. Sequeda, Marcelo Arenas, and Daniel P. Miranker. 2014. OBDA: Query Rewriting or Materialization? In Practice, Both!. In *ISWC*.
[45] Grégory Smits, Olivier Pivert, Hélène Jaudoin, and François Paulus. 2014. AGGREGO SEARCH: Interactive Keyword Query Construction. In *EDBT*.
[46] W3C. 2013. SPARQL 1.1 Query Language. https://www.w3.org/TR/sparql11-query/
[47] W3C. 2014. RDF 1.1 Concepts and Abstract Syntax. https://www.w3.org/TR/rdf11-concepts/
[48] W3C. 2014. RDF 1.1 Semantics. https://www.w3.org/TR/rdf11-mt/#rdfs-entailment
[49] Gio Wiederhold. 1992. Mediators in the Architecture of Future Information Systems. *IEEE Computer* 25, 3 (1992).