Ontology-Based RDF Integration of Heterogeneous Data

Maxime Buron, François Goasdoué, Ioana Manolescu, Marie-Laure Mugnier

Ontology-Based RDF Integration of Heterogeneous Data

Maxime Buron
Inria and LIX
(UMR 7161, CNRS and
Ecole polytechnique), France
maxime.buron@inria.fr

François Goasdoué
Univ Rennes, CNRS, IRISA,
France
fg@irisa.fr

Ioana Manolescu
Inria and LIX
(UMR 7161, CNRS and
Ecole polytechnique), France
ioana.manolescu@inria.fr

Marie-Laure Mugnier
Univ. Montpellier, LIRMM,
Inria Sophia, France
mugnier@lirmm.fr

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 all 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 introduce the novel class of *RDF Integration Systems (RIS)*, going beyond the state of the art in the expressive power, that is, in the ability to expose, integrate and flexibly query data from heterogeneous sources through GLAV (global-local-as-view) mappings. Our second contribution is a set of *query answering strategies*, two combining existing techniques and three others based on an innovative integration of view-based rewriting; our experiments show that the latter bring strong performance advantages.

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. Since the early days of data integration [26], *graphs* have been chosen as an integration data model, for their flexibility; more recently, the advent of Semantic Web technologies has put forward the RDF data model [1] endowed with the standard SPARQL query language. Beyond its flexibility, RDF brings two qualitative advantages. (i) It is natively connected with standardized *ontology* languages such as RDFS and OWL; an ontology allows to capture important application knowledge, thus querying an integration system through the ontology allow interpreting the data through the application's semantic lens; (ii) going beyond logical integration settings based on Description Logics (DL, in short), e.g., [36, 29, 31, 3, 40, 19, 13], SPARQL enables flexible querying of data together with the ontology. For instance, the RDF query "find all the company's personnel together with their types" is not expressible in a DL setting, just like in SQL, one must specify the relations one wants to query. The ability to query the data together with the ontology is valuable for complex RDF graphs whose ontology is not fully known to users. For these reasons, in this work we rely on **RDF as the integration model**, and on **ontologies** to describe application semantics.

In an integration system, the data sources schemas, commonly called *local schemas*, must be related to the integration or *global* schema [23]. The simplest option is to 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, in Local-As-View (LAV) data integration, elements of the local schemas are defined as views over the global one. Query answering in this context requires *rewriting the query with the views* describing the local sources [32]. GLAV (Global-Local-As-View) data integration generalizes both GAV and LAV. In GLAV scenarios, a query q_1 over one or several local schemas is associated to a query over the global schema q_2, having the same answer variables; the pair (q_1, q_2) is commonly called a *mapping*. The semantics is that for each answer of q_1, the integration system exposes the data comprised in an 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. Moreover, GLAV mappings enable a certain kind of *value invention* which (in a precise sense explained in Section 3, Example 9) *increases the amount of information accessible through the integration system*, e.g., to state *the existence* of some data whose *values* are not known in the sources. Thus, in this work, we follow the most ambitious **GLAV integration approach**.

Ontology-Based Data Access (OBDA) has recently emerged as a data integration paradigm based on the use of ontologies and reasoning [40]. It separates a conceptual level, defined by an ontology that describes the application knowledge (thus plays the role of the global schema) and a data level defined by data sources, both levels being connected by mappings. Classically, an OBDA system combines a DL

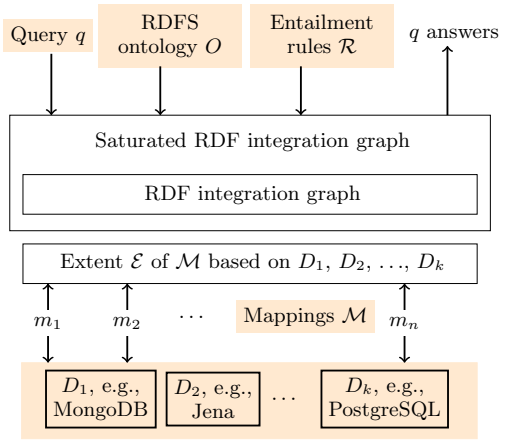

Figure 1: Outline of an RDF Integration System.

ontology, a relational database and GAV mappings (see Section 6 for details). Our work follows the OBDA vision although it implements it in a different setting (RDFS, heterogeneous data sources and GLAV mappings).

Contributions and novelty The contributions we make in this work are as follows.

(1). RIS Architecture We introduce **RDF Integration Systems** (RIS, in short), a novel class of integration systems 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 on such integration RDF graphs. RIS go beyond the state of the art by applying GLAV mappings to a heterogeneous data setting, and by answering, on the integrated RDF graph, queries that are strictly more expressive than allowed in similar systems using DL as the integration model. Thus, RIS provide *more flexibility*, or, equivalently, *the power to express more integration scenarios* than previous methods.

To give a global view of our work, Figure 1 outlines a RIS; a full formalization appears in Section 3. Colored areas identify RIS "ingredients", namely: a set of *heterogeneous data sources* $D_1, D_2, \ldots, D_k$; a set of GLAV *mappings* $\mathcal{M}$; an RDFS *ontology* O characterizing the semantics of the integration graph; and a set of *inference (or entailment) rules* $\mathcal{R}$, which enable reasoning with the ontology. For instance, a rule $r \in \mathcal{R}$ may state that *subclassOf is transitive*, i.e., for any semantic classes c_1, c_2, c_3, if c_1 is a subclass of c_2 and c_2 is a subclass of c_3, then c_1 is a subclass of c_3; the ontology may state that a "tempAccountant" is a subclass of "accountant" while the latter is a subclass of "employee". From $D_1, \ldots, D_k$ and $\mathcal{M}$ derives the *extent* $\mathcal{E}$, i.e., the source data exposed by the source queries q_1 in the mappings; this data is transformed into an *RDF integration graph*. Adding the application ontology O and enriching the result through reasoning with O and rules $\mathcal{R}$ leads to more information becoming available through the RIS, in the *saturated RDF integration graph*. For instance, if the integration graph states that Alice is a "tempAccountant", and assuming O contains the two subclass statements above, its saturation also states that Alice is an "accountant", and an "employee". We define RIS query answers with respect to this largest graph, which benefits from the results of ontological reasoning.

(2). Novel, efficient RIS query answering techniques We describe two RIS query answering methods relying on ingredients known from the previous literature, as well as three novel methods using an innovative transformation of *mappings into materialized views*, then relying on view-based rewriting to identify ways to answer the query. We implemented all these methods in a platform which combines and extends off-the-shelf components; we provide a quantitative and qualitative analysis of their performance, and identify the last one as the most efficient and robust when the integrated data changes.

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 our 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), as well as how they can be queried using the widely-considered SPARQL Basic Graph Pattern queries (Section 2.3). Then, we recall two reasoning mechanisms on queries (Section 2.4), namely *query reformulation* and *query saturation*, as well as the technique of *view-based query rewriting* (Section 2.5). They will serve as basic building blocks for the query answering techniques we shall devise for our RDF integration systems.

2.1 RDF Graph

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* [5, 30]. 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 $(\mathtt{s}, \mathtt{p}, \mathtt{o})$ states that its *subject* $\mathtt{s}$ has the *property* $\mathtt{p}$ with the *object* value $\mathtt{o}$ [1]. 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 strings between quotes to denote literals.

Within an RDF graph, we distinguish **data** triples from **schema** ones. The former describe data (either attach a type, or a class, to a resource, or state the value of a certain data property of a resource), while the latter state RDF Schema (RDFS) constraints 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 1 introduces short notation 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 1. The rest of the IRIs comprises 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 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)*

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

Table 1: RDF triples.

	Rule [2]	Entailment rule	
$\mathcal{R}$	rdfs5	$(\mathtt{p}_1, \prec_{sp}, \mathtt{p}_2), (\mathtt{p}_2, \prec_{sp}, \mathtt{p}_3) \rightarrow (\mathtt{p}_1, \prec_{sp}, \mathtt{p}_3)$	$\mathcal{R}_c$
	rdfs11	$(\mathtt{s}, \prec_{sc}, \mathtt{o}), (\mathtt{o}, \prec_{sc}, \mathtt{o}_1) \rightarrow (\mathtt{s}, \prec_{sc}, \mathtt{o}_1)$	
	ext1	$(\mathtt{p}, \hookleftarrow_{d}, \mathtt{o}), (\mathtt{o}, \prec_{sc}, \mathtt{o}_1) \rightarrow (\mathtt{p}, \hookleftarrow_{d}, \mathtt{o}_1)$	
	ext2	$(\mathtt{p}, \hookrightarrow_{r}, \mathtt{o}), (\mathtt{o}, \prec_{sc}, \mathtt{o}_1) \rightarrow (\mathtt{p}, \hookrightarrow_{r}, \mathtt{o}_1)$	
	ext3	$(\mathtt{p}, \prec_{sp}, \mathtt{p}_1), (\mathtt{p}_1, \hookleftarrow_{d}, \mathtt{o}) \rightarrow (\mathtt{p}, \hookleftarrow_{d}, \mathtt{o})$	
	ext4	$(\mathtt{p}, \prec_{sp}, \mathtt{p}_1), (\mathtt{p}_1, \hookrightarrow_{r}, \mathtt{o}) \rightarrow (\mathtt{p}, \hookrightarrow_{r}, \mathtt{o})$	
	rdfs2	$(\mathtt{p}, \hookleftarrow_{d}, \mathtt{o}), (\mathtt{s}_1, \mathtt{p}, \mathtt{o}_1) \rightarrow (\mathtt{s}_1, \tau, \mathtt{o})$	$\mathcal{R}_a$
	rdfs3	$(\mathtt{p}, \hookrightarrow_{r}, \mathtt{o}), (\mathtt{s}_1, \mathtt{p}, \mathtt{o}_1) \rightarrow (\mathtt{o}_1, \tau, \mathtt{o})$	
	rdfs7	$(\mathtt{p}_1, \prec_{sp}, \mathtt{p}_2), (\mathtt{s}, \mathtt{p}_1, \mathtt{o}) \rightarrow (\mathtt{s}, \mathtt{p}_2, \mathtt{o})$	
	rdfs9	$(\mathtt{s}, \prec_{sc}, \mathtt{o}), (\mathtt{s}_1, \tau, \mathtt{s}) \rightarrow (\mathtt{s}_1, \tau, \mathtt{o})$	

Table 2: Sample RDFS entailment rules.

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 [30]. More importantly, we forbid ontology triples from altering the common semantics of RDF itself. For instance, we do not allow $(\hookleftarrow_d, \prec_{sp}, \hookrightarrow_r)$, that 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 [10] thus the W3C's Web Ontology Language (OWL), Datalog± [18] and existential rules [38], etc.

Example 1 (Running example, based on [15]). *Consider the following RDF graph:*

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

The ontology of G_{ex}, i.e., the first eight schema triples, states that persons are working for organizations, some of which are public administrations or companies. Further, national companies are particular 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 company* $\_{:}b_c$*, which is a national company, and* :p$_2$ *is hired by the public administration* :a.

2.2 RDF Entailment Rules

An *entailment rule* r has the form body(r) $\rightarrow$ head(r), where body(r) and head(r) are RDF graphs, respectively called *body* and *head* of the rule r. In this work, we consider the **RDFS entailment rules** $\mathcal{R}$ shown in Table 2, 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 $\mathtt{p}_1$ is a subproperty of another property $\mathtt{p}_2$, and further $\mathtt{p}_2$ is a subproperty of $\mathtt{p}_3$ (body of `rdfs5`), it follows that $\mathtt{p}_1$ is a subproperty of $\mathtt{p}_3$ (head of `rdfs5`). Similarly, rule `rdfs7` states that if $\mathtt{p}_1$ is a subproperty of $\mathtt{p}_2$ and resource `s` has the value `o` for $\mathtt{p}_1$, then `s` also has `o` as a value for $\mathtt{p}_2$. The triples $(\mathtt{p}_1, \prec_{sp}, \mathtt{p}_3)$ and $(\mathtt{s}, \mathtt{p}_2, \mathtt{o})$ 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 [15], we view $\mathcal{R}$ as partitioned in two subsets: the rules $\mathcal{R}_c$ lead to implicit schema triples, while rules $\mathcal{R}_a$ lead to implicit data triples.

The **direct entailment** of an RDF graph G with a set of RDF entailment rules $\mathcal{R}$, denoted by $C_{G,\mathcal{R}}$, 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 $(\_{:}b_c, \tau, :\text{NatComp})$, $(:\text{NatComp}, \prec_{sc}, :\text{Comp})$, leads to the implicit triple $(\_{:}b_c, \tau, :\text{Comp})$. 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 [2]. Formally:

Definition 2 (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. *The saturation of G_{ex} w.r.t. the set $\mathcal{R}$ of RDFS entailment rules shown in Table 2 is attained after the following* two *saturation steps:*

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

$(G_{\text{ex}})_2 = (G_{\text{ex}})_1 \cup$
$\{(:\text{p}_1, \tau, :\text{Person}), (:\text{p}_2, \tau, :\text{Person}), (\_{:}b_c, \tau, :\text{Org})\}$

2.3 Basic Graph Pattern Queries

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

Definition 3 (BGP Queries). *A* BGP query *q is of the form $q(\bar{x}) \leftarrow P$, where P is a BGP (also denoted by body(q)), and $\bar{x} \subseteq$ Var(P) are the answer variables of q. The* arity *of q is that of $\bar{x}$, i.e., $|\bar{x}|$.*

Partially instantiated BGPQs generalize BGPQs and have been used in the literature [30, 15]. Starting from a BGPQ q, partial instantiation replaces *some* variables and/or blank nodes with values from $\mathcal{I} \cup \mathcal{L} \cup \mathcal{B}$, as specified by a substitution σ. The partially instantiated query is denoted by q_σ; we denote by $q(\bar{x}_\sigma)$ its head, and by body$(q)_\sigma$

its body. Observe that when $\sigma = \emptyset$, q_σ coincides with q. Further, due to σ, and in contrast with standard BGPQs, some answer variables of q_σ can be bound:

EXAMPLE 3. *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* q_σ *is* $q(:\text{p}_1, y) \leftarrow (:\text{p}_1, :\text{worksFor}, z), (z, \tau, y), (y, \prec_{sc}, :\text{Comp})$.

The semantics of a (partially instantiated) BGPQ on an RDF graph is defined through homomorphisms from the query body to the queried graph:

DEFINITION 4 (BGP TO RDF GRAPH HOMOMORPHISM). *A* homomorphism *from a BGP* q *to an RDF graph* G *is a function* φ *from* $\text{Val}(body(q))$ *to* $\text{Val}(G)$ *such that for any triple* $(s, p, o) \in body(q)$*, the triple* $(\varphi(s), \varphi(p), \varphi(o))$ *is in* G*, with* φ *the identity on IRIs and literals. Such a homomorphism is noted* $G \models^\varphi q$*, or simply* $G \models q$ *to solely denote its existence.*

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 5 (EVALUATION AND ANSWERING). *Let* q_σ *be a partially instantiated BGPQ* q_σ*.*

The answer set *to* q_σ *on an RDF graph* G *w.r.t. a set* $\mathcal{R}$ *of RDF entailment rules is:* $q_\sigma(G, \mathcal{R}) = \{\varphi(\bar{x}_\sigma) \mid G^\mathcal{R} \models^\varphi body(q)_\sigma\}$*. If* $\bar{x} = \emptyset$*,* q_σ *is a* Boolean *query, in which case* q_σ *is false when* $q_\sigma(G, \mathcal{R}) = \emptyset$ *and true when* $q_\sigma(G, \mathcal{R}) = \{\langle\rangle\}$*, i.e., the answer to* q_σ *is an empty tuple.*

The evaluation *of* q_σ *on* G*, denoted* $q_\sigma(G, \emptyset)$ *or* $q_\sigma(G)$ *in short, is obtained only through standard BGP-to-RDF homomorphisms. It can be seen as a particular case of query answering when* $\mathcal{R} = \emptyset$*.*

These notions and notations naturally extend to *unions* of (partially instantiated) BGPQs. For simplicity, below we term **UBGPQ** a union of BGPQs, whether or not its BGPQs are partially instantiated.

EXAMPLE 4. *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* $:\text{p}_1$ *being CEO of* $_:b_c$*,* $:\text{p}_1$ *implicitly works for it, and* $_:b_c$ *is explicitly a company of the particular type* :NatComp*.*

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

2.4 Reasoning on Queries

Above, we have discussed reasoning on an RDF graph using an ontology, through entailment rules. We now introduce two reasoning techniques which apply on *queries*.

2.4.1 BGPQ Reformulation

Reformulation-based query answering is an alternative technique to the widely adopted saturation-based query answering. It consists in *reformulating* a query, so that simple evaluation of the reformulated query on G yields the complete answer set to the original query on G. Intuitively, reformulation injects the ontological knowledge into the query, just as saturation injects it into the graph. We rely here on the very recent technique of [15], which proposes the first reformulation algorithm able to take account of the whole set $\mathcal{R}$ of entailment rules (recall Table 2). This work extends a previous reformulation algorithm from [30], which is restricted to answering queries on data triples, i.e., according to the subset $\mathcal{R}_a$ of $\mathcal{R}$.

The query reformulation process from [15], which we denote here by $\text{Ref}(q, O)$, is decomposed into two steps according to the partition of $\mathcal{R}$ into $\mathcal{R}_a$ and $\mathcal{R}_c$. The first step, denoted by $\text{Ref}_c(q, O)$, reformulates a BGPQ q w.r.t. the ontology O and the set of rules $\mathcal{R}_c$ into a UBGPQ, say $\mathcal{Q}_c$, which is guaranteed not to contain ontology triples. Intuitively, this step generates new BGPQs by instantiating variables that query the ontology, e.g., y in a query triple (x, τ, y), with all IRIs (in this case, class IRIs) known in the ontology. This step is sound and complete w.r.t. $\mathcal{R}_c$, i.e., for any graph G, $q(G, \mathcal{R}_c) = \mathcal{Q}_c(G) = \mathcal{Q}_c(G \setminus O)$; furthermore, $q(G, \mathcal{R}) = \mathcal{Q}_c(G, \mathcal{R}_a)$. The second step, which we denote by $\text{Ref}_a(\mathcal{Q}_c, O)$, reformulates $\mathcal{Q}_c$ w.r.t. O and $\mathcal{R}_a$, and outputs a UBGPQ, say $\mathcal{Q}_{c,a}$. This step is sound and complete w.r.t. $\mathcal{R}_a$, i.e., for any graph G, $\mathcal{Q}_c(G, \mathcal{R}_a) = \mathcal{Q}_{c,a}(G)$. Together, these two steps make reformulation-based BGPQ answering sound and complete [15]:

THEOREM 1. *For any RDF graph* G *with* O *its ontology, and BGPQ* q*, it holds that* $q(G, \mathcal{R}) = \mathcal{Q}_{c,a}(G) = \mathcal{Q}_{c,a}(G \backslash O)$*, where* $Q_{c,a} = Ref(q, O) = Ref_a(Ref_c(q, O), O)$*.*

EXAMPLE 5. *The above reformulation of [15] for the query* $q(x, y) \leftarrow (x, :\text{worksFor}, z), (z, \tau, y), (y, \prec_{sc}, :\text{Comp})$ *from the preceding example is computed by reformulating first* q *into* $\mathcal{Q}_c = q(x, :\text{NatComp}) \leftarrow (x, :\text{worksFor}, z), (z, \tau, :\text{NatComp})$*, here by instantiating the* q*'s triple* $(y, \prec_{sc}, :\text{Comp})$ *on* O*. Then,* $\mathcal{Q}_c$ *is reformulated into* $\mathcal{Q}_{c,a} =$

$q(x, :\text{NatComp}) \leftarrow (x, :\text{worksFor}, z), (z, \tau, :\text{NatComp})$
$\cup\ q(x, :\text{NatComp}) \leftarrow (x, :\text{hiredby}, z), (z, \tau, :\text{NatComp})$
$\cup\ q(x, :\text{NatComp}) \leftarrow (x, :\text{ceoOf}, z), (z, \tau, :\text{NatComp})$

by specializing :worksFor *according to its subproperties in* O*. It can be checked that* $\mathcal{Q}_{c,a}(G_{\text{ex}}) = q(G_{\text{ex}}, \mathcal{R}) = q(G_{\text{ex}}^\mathcal{R}) = \langle :\text{p}_1, :\text{NatComp}\rangle$*, obtained here from the third BGPQ in* $\mathcal{Q}_{c,a}$*.*

2.4.2 BGPQ Saturation

BGPQ saturation has been introduced recently [25] to complement a query q with all the implicit triples it asks, given the RDFS ontology O and set $\mathcal{R}$ of RDF entailment rules.

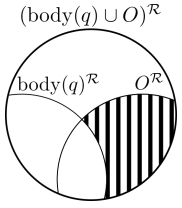

Figure 2: BGPQ saturation.

To this aim, the saturation of RDF graphs has been extended to BGPs by treating variables as IRIs. Then, the saturation of q w.r.t. $\mathcal{R}$ and O, which we note $q^{\mathcal{R},O}$, is obtained as the saturation of the body of q together with O, using $\mathcal{R}$, from which are pruned out the triples entailed by O but not by $\text{body}(q)$, i.e., not relevant to q. The set characterisation of the $q^{\mathcal{R},O}$ triples is shown in Figure 2; the hatched area corresponds to the removed triples. From a practical viewpoint, this characterisation also provides a simple way to compute the saturation of q w.r.t. $\mathcal{R}$ and O:

$$q^{\mathcal{R},O} = (\text{body}(q) \cup O)^\mathcal{R} \setminus (O^\mathcal{R} \setminus \text{body}(q)^\mathcal{R})$$

EXAMPLE 6. *Consider the ontology* O *of* G_{ex} *and the query*

$q(x) \leftarrow (x, :\text{hiredBy}, y), (y, \tau, :\text{NatComp})$ *asking for who has been hired by a national company. Its saturation w.r.t.* $\mathcal{R}, O$ *is:* $q^{\mathcal{R},O}(x, y) \leftarrow body(q), (x, :\text{worksFor}, y), (x, \tau, :\text{Person}), (y, \tau, :\text{Comp}), (y, \tau, :\text{Org})$. *This saturation is computed as shown above, where the pruned triples are those in* $O^{\mathcal{R}}$ *(because* $(O^{\mathcal{R}} \setminus body(q)^{\mathcal{R}}) = O^{\mathcal{R}}$ *here).*

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 [32, 23], in particular for the so-called *local-as-view* data integration. Then we present a generalization of the notion of views as mappings [35] (Section 2.5.2).

2.5.1 View-based 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 S assigns a set of tuples to each relation in S. The data stored in a view V is called its *extension*. Further, to each view V is associated a query $V(\bar{x}) :- \varphi(\bar{x})$ over the global schema S, specifying how its data fits into S. For instance, let S consist of two relations *Emp(eID, name, dID), Dept(dID, cID, country)*, where *eID*, *dID* and *cID* are respectively identifiers for employees, departments and companies. Consider the views V_1*(eID, name, dID) :- Emp(eID, name, dID), Dept(dID, "IBM", country)* providing information about IBM employees, and V_2*(dID, country) :- Dept(dID, cID, country)*, which brings (department, country) pairs. 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, possibly overlapping with V_1; this is called the "Open World Assumption" (OWA).

In an OWA setting, we are interested in *certain answers* [32], 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 $\mathcal{V}$, commonly called *view-based rewriting* (or rewriting, in short); this can be evaluated over the view extensions in order to produce the answers.

Ideally, a rewriting should be *equivalent* to the original query, i.e., have the same answers as the query. Such a rewriting may not always exist, depending on the views and the query. For instance, the query $q(n, d) :-$ *Emp*(e, n, d), *Dept*$(d, c,$ *"France"*$)$ does not have an equivalent rewriting using V_1 and V_2, because only IBM employees are available through these views. 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, $q_r(n, d) :- V_1(e, n, d), V_2(d,$ *"France"*$)$ is a maximally contained rewriting of q; it returns employees of French IBM departments, whereas the query was not restricted to IBM.

A remarkable result holds for *(unions of) conjunctive queries ((U)CQs)*, *conjunctive views* (views V such that the associated query $V(\bar{x}) :- \varphi(\bar{x})$ is a CQ) and *rewritings that are UCQs*: any maximally contained rewriting computes exactly the certain answers. This follows from Theorem 3.2 of [4]; we will build upon this result for answering queries in our RDF integration systems.

2.5.2 Mapping-based 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, salary)* with people and their work contracts, the view V_1 from the above example may be defined on D by: V_1^D *(eID, name, dID) :- Person(eID, name), Contract(eID, dID, salary)* (note that V_1^D hides the salaries from $\mathcal{I}$); 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).

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).

An association of a query over a data source and another query over the global schema, e.g., V_1^D and V_1 above, is commonly called a *mapping*. Depending on the expressiveness allowed for the query over the global schema, a mapping is said either *GLAV* or just "global-as-view" (GAV). In the former case, the query may comprise non-answer variables thus the view provides an *incomplete* sub-instance for the global schema, while in the latter case all the variables must be answer ones so the view provides a *complete* sub-instance. For example, suppose that $\langle 1,$ *"John Doe"*$,2\rangle$ is an answer to V_1^D above, hence a tuple in the extension of V_1. For this tuple, V_1 models in the incomplete sub-instance of S: *Emp*$(1,$ *"John Doe"*$,2)$, *Dept*$(2,$ *"IBM"*$,x)$, where x is a *variable* or a *labelled null* [5] modelling an existing but non-available information.

GAV mappings have been used in the ontology-based data access setting to connect data sources to a DL ontology (see Section 6). Here we adopt the more general GLAV mappings to access data from heterogeneous sources through an RDFS ontology; also, we have developed the first concrete system implementing this type of mappings.

3. PROBLEM STATEMENT

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

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 data triples derived from the sources by means of GLAV-style *mappings*:

DEFINITION 6 (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 arity, and further* q_2 *is a BGPQ whose body contains only triples of the forms:*

- $(\mathtt{s}, p, \mathtt{o})$ *where* $p \in \mathscr{I}_{\text{user}}$,
- $(\mathtt{s}, \tau, C)$ *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* $\text{ext}(m) = \{m(\delta(v_1), \ldots, \delta(v_n)) \mid \langle v_1, \ldots, v_n\rangle \in q_1(D)\}$, *where* $q_1(D)$ *is the answer set of* q_1 *on the data source* D *and* δ *is a function that maps source values to RDF values, i.e., IRIs, blank nodes and literals.*

Intuitively, m specifies that data from D comprised in the

result of query q_1, expressed in the native query language of D, is also present in the RIS integration graph, as result of the (BGP) query q_2. Since the actual data resides in the data sources, m *exposes* some D data in the integration graph. The extension of m is the set of tuples obtained by instantiating the body of q_2 for each result of q_1.

EXAMPLE 7 (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 URI* $:\text{p}_1$. *Then, the extension of* m_1 *is:* $\text{ext}(m_1) = \{m_1(:\text{p}_1)\}$. *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 URIs* $:\text{p}_2, :\text{a}$. *Then, the extension of* m_2 *is:* $\text{ext}(m_2) = \{m_2(:\text{p}_2, :\text{a})\}$.

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 $\text{Val}(\mathcal{E})$ the set of values occurring in $\mathcal{E}$. We can now define the RIS data triples *induced* by some mappings and an extent thereof.

DEFINITION 7 (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:*

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

where

- $body(q_2)_{[\bar{x} \leftarrow \bar{t}]}$ *is the BGP* $body(q_2)$ *in which the answer variables* $\bar{x}$ *are bound to the values in the tuple* $m(\bar{t})$, *part of* $\mathcal{E}$;
- $\text{bgp2rdf}(\cdot)$ *is a function that transforms a BGP into an RDF graph, by replacing each variable by a fresh blank node.*

The RIS data triples are exactly *all* the data available to queries through a RIS. Mappings allow to (i) select only the data relevant to a given application, and (ii) organize it according to the shape desired in the integration RDF graph. These triples may, but do not have to, be materialized, as we discussed in Section 4. Observe that by definition, RIS data triples may include some *invented values*, that is, fresh blank nodes as exemplified below.

EXAMPLE 8. *Reusing the mappings from Example 7, let* $\mathcal{M} = \{m_1, m_2\}$ *and its extent* $\mathcal{E} = \{m_1(:\text{p}_1), m_2(:\text{p}_2, :\text{a})\}$. *The RIS data triples they lead to are:*

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

Note that $G_{\mathcal{E}}^{\mathcal{M}}$ *features some* incomplete information*: 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. *Only* non-answer variables *in a mapping head lead to blank nodes being introduced in this way in RIS data triples: by Def. 7,* answer *variables in* q_2 *are replaced with values from* $m(\bar{t})$ *(thus, from* $\text{Val}(\mathcal{E})$*).*

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:

DEFINITION 8 (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:*

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

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

$\text{cert}(q, S)$ is the subset of $q(O \cup G_{\mathcal{E}}^{\mathcal{M}}, \mathcal{R})$ restricted to *complete* answers, i.e., tuples fully built from source values; tuples with blank nodes introduced by the bgp2rdf function in Def. 7 are excluded from $\text{cert}(q, S)$. Note, however, that the incompleteness (blank nodes) introduced in Def. 7 can be exploited to answer queries, as shown below.

EXAMPLE 9. *Consider the RIS* S *made of the ontology* O *of* G_{ex} *in Example 1, the set* $\mathcal{R}$ *of entailment rules shown in Table 2, and the set of mappings* $\mathcal{M}$ *together with the extent* $\mathcal{E}$ *from Example 8.*

Let $q(x, y) \leftarrow (x, :\text{worksFor}, y), (y, \tau, :\text{Comp})$ *be the query asking "who works for <u>which</u> company", while the query* $q'(x) \leftarrow (x, :\text{worksFor}, y), (y, \tau, :\text{Comp})$ *asks "who works for <u>some</u> company". The only difference between them is that* y *is an answer variable in* q *and not in* q'. *The certain answer of* q *is* $\emptyset$, *while the certain answer of* q' *is* $\langle :\text{p}_1 \rangle$. *This answer results from the RIS data triples* $(:\text{p}_1, :\text{worksFor}, \_{:}b_c)$, $(\_{:}b_c, \tau, :\text{Comp})$, *which are entailed from:*

- *the* $G_{\mathcal{E}}^{\mathcal{M}}$ *triples* $(:\text{p}_1, :\text{ceoOf}, \_{:}b_c), (\_{:}b_c, \tau, :\text{NatComp})$, *with the blank node* $\_{:}b_c$ *discussed in Example 8, and:*
- *either the* O *triples* $(:\text{ceoOf}, \prec_{sp}, :\text{worksFor})$, $(:\text{ceoOf}, \hookrightarrow_r, :\text{Comp})$ *together with the* $\mathcal{R}$ *rules* `rdfs3`, `rdfs7`, *or the* O *triples* $(:\text{ceoOf}, \prec_{sp}, :\text{worksFor})$, $(:\text{NatComp}, \prec_{sc}, :\text{Comp})$ *together with the* $\mathcal{R}$ *rules* `rdfs3`, `rdfs9`.

Because "invented" blank nodes are not allowed in $\text{cert}(q, S)$, q *has no answer. In contrast, its close variant* q' *allows finding out that* $:\text{p}_1$ *is CEO of <u>some</u> national company, even though the mapping* m_1 *(the only one involving companies) does not expose the company URI through the RIS.* The presence of bgp2rdf-inserted blank nodes may increase the amount of information one can get from a RIS*, even if such nodes are not allowed in certain answers. In turn, such blank nodes are enabled by non-answer variables in the heads of the expressive GLAV mappings we consider.*

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.,* $\text{cert}(q, S)$.

4. QUERY ANSWERING IN A RIS

This section is devoted to our core technical contributions: *RIS query answering approaches.* Def. 8 outlines the tasks involved: (i) computing the RIS data triples $G_{\mathcal{E}}^{\mathcal{M}}$; (ii) reasoning on this graph, with the ontology O, under the RDFS entailment rules $\mathcal{R}$; (iii) computing answers to the user query q on the graph resulting from the reasoning.

Below, we present *five strategies for the RIS query answering problem*; they differ in *when and how* these steps are performed. Specifically, Section 4.1 presents two methods which start by actually computing the RIS data triples; this is reminiscent of an extract-transform-load (or warehouse) scenario in classical data integration literature [39], followed by an RDF query answering stage (as recalled in Section 2.2). The strategies described in Section 4.2 are our most innovative ones; they use the mappings as *views*, in order to reduce query answering in a RIS to relational view-based query rewriting (recalled in Section 2.5.1).

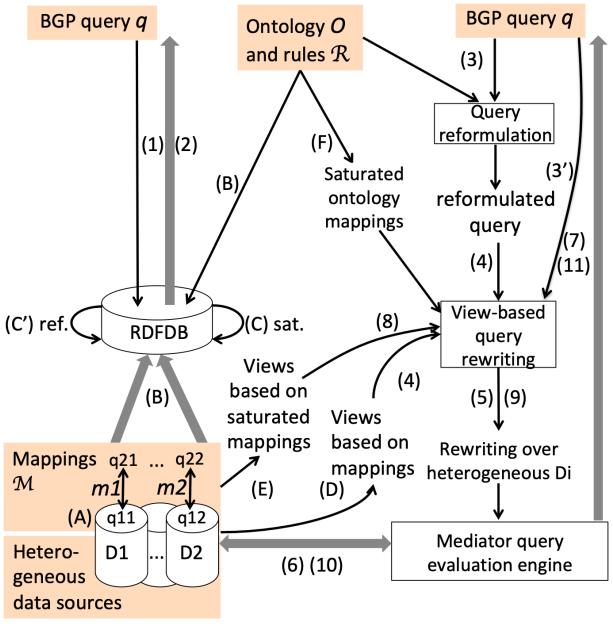

Figure 3: Outline of query answering strategies.

4.1 RIS Query Answering through RIS Data Materialization: MAT and MAT-CA

We first present two strategies which *materialize* the RIS data triples, then reduce RIS query answering to *RDF query answering*: either based on graph saturation (MAT), or on query reformulation (MAT-CA, whose acronym reflects reformulation with both the $\mathcal{R}_c$ and $\mathcal{R}_a$ rules from Table 2).

Figure 3 traces the various steps involved in our strategies; the thick gray arrows follow data (partial or final query answers), whereas the others trace transformations applied on the query, mappings, and/or ontology, all of which are typically orders of magnitude smaller than the data. The main algorithmic building blocks, e.g., "View-based rewriting", are shown in boxes; the figure also shows some partial computation results being exchanged during query answering, e.g., "Saturated mappings" etc. During materialization, the q_1 queries part of each mapping are executed (step (A)), then their results are converted as per Def. 7 into RIS data triples, which are stored into an RDF data management system (step (B)).

Then, when a BGPQ q is asked (step (1)) against the RDF graph G made of the RIS data triples and the ontology O, it can be answered (step (2)): either by the saturation-based technique (step (C)) on the graph G saturated with the set $\mathcal{R}$ of RDFS entailment rules, or by the reformulation-based technique (step (C')), reformulating q using O and $\mathcal{R}$ into a UBGPQ $\mathcal{Q}_{c,a}$ (recall Section 2.4.1), then evaluating $\mathcal{Q}_{c,a}$ on the RIS data triples only.

The next theorem states the correctness of our two above-mentioned RIS query answering methods:

THEOREM 2 (MAT AND MAT-CA CORRECTNESS). *Let q be a BGPQ asked on the RIS $S = \langle O, \mathcal{R}, \mathcal{M}, \mathcal{E}\rangle$, whose materialized RDF graph is $G = O \cup G^{\mathcal{M}}_{\mathcal{E}}$. Then:*

- $\text{cert}(q, S) = \{\varphi(\bar{x}) \mid G^{\mathcal{R}} \models^{\varphi} q(\bar{x}), \varphi(\bar{x}) \subseteq \text{Val}(\mathcal{E})^{|\bar{x}|}\}$;
- $\text{cert}(q, S) = \bigcup_{i=1}^{n} \{\varphi(\sigma_i(\bar{x})) \mid G \models^{\varphi} q^i_{\sigma_i}(\bar{x}), \varphi(\sigma_i(\bar{x})) \subseteq \text{Val}(\mathcal{E})^{|\bar{x}|}\}$ *with* $\bigcup_{i=1}^{n} q^i_{\sigma_i}(\bar{x})$ *the reformulation* $\mathcal{Q}_{c,a}$ *of* q *w.r.t.* $O, \mathcal{R}$ *(recall Section 2.4.1).*

The proofs of the above items follow from the definition of certain answers in a RIS (Def. 8), and either from that of RDF query answers (Def. 5) for the saturation-based approach (first item) or from Theorem 1 for the reformulation-based approach (second item).

EXAMPLE 10. *Consider the RIS of Example 9, whose ontology O and $G^{\mathcal{M}}_{\mathcal{E}}$ data triples form exactly the G_{ex} RDF graph in Example 1. Examples 4 and 5 therefore also illustrate how queries are handled within this RIS, through MAT or MAT-CA query answering.*

These two methods push the extent materialization work before queries are received, thus speeding up query answering. However, the materialized RIS data triples must be maintained when the data sources are updated. In turn, changes to the RIS data triples changes require maintaining $G^{\mathcal{R}}$ for the saturation-based strategy. Hence, approaches based on materializing RIS data triples are generally not adapted to contexts where data changes frequently.

4.2 RIS Query Answering through View-based Query Rewriting

Next, we discuss RIS query answering strategies based on view-based query rewriting. The first (Section 4.2.1) combines query reformulation and relational view-based query rewriting; such a combination has been used to integrate data in a centralized [29, 31, 40] or peer-to-peer [3] description logic setting, but, to the best of our knowledge, not in the RDF one. In contrast, the last two (Section 4.2.2) rely on intricate combinations of query reformulation, query saturation (Section 2.4.2) and relational view-based query rewriting. As we will show, BGPQ saturation allows reducing significantly the query-time reasoning effort.

For our discussion, 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 *ubgpq2cq* 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.

4.2.1 Rewriting Reformulated Queries using Mappings as Views: REW-CA

In this approach, the incoming BGPQ q is first reformulated (recall Section 2.4.1) into a UBGPQ $\mathcal{Q}_{c,a}$ with the ontology O and rules $\mathcal{R}$; reformulation with $\mathcal{R}_c$ and $\mathcal{R}_a$ accounts for the second part of the acronym. However, here, $\mathcal{Q}_{c,a}$ cannot be evaluated on the RIS data triples, because they are not materialized. Instead, we view each mapping as a relational LAV view over the integration graph, and rewrite the query based on these views. Specifically, such views are defined on the global relational schema $\{T\}$, where T is the triple relation introduced earlier in this section:

DEFINITION 9 (MAPPINGS AS RELATIONAL VIEWS). *Let $m = q_1(\bar{x}) \rightsquigarrow q_2(\bar{x})$ be a mapping. Its corresponding relational view is: $m(\bar{x}) \leftarrow bgp2ca(body(q_2))$.*

EXAMPLE 11. *The relational views corresponding to the mappings m_1, m_2 from Example 7 are:*

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

The set of all views corresponding to a set of mappings $\mathcal{M}$

$\mathcal{Q}_{c,a} =$

$q(x, :\text{ceoOf}) \leftarrow \quad T(x, :\text{ceoOf}, z), T(z, \tau, :\text{NatComp}),$
$T(x, :\text{worksFor}, a), T(a, \tau, :\text{PubAdmin})$

$\cup\ q(x, :\text{ceoOf}) \leftarrow \quad T(x, :\text{ceoOf}, z), T(z, \tau, :\text{NatComp}),$
$T(x, :\text{hiredBy}, a), T(a, \tau, :\text{PubAdmin})$

$\cup\ q(x, :\text{ceoOf}) \leftarrow \quad T(x, :\text{ceoOf}, z), T(z, \tau, :\text{NatComp}),$
$T(x, :\text{ceoOf}, a), T(a, \tau, :\text{PubAdmin})$

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

$\cup\ q(x, :\text{hiredBy}) \leftarrow T(x, :\text{hiredBy}, z), T(z, \tau, :\text{NatComp}),$
$T(x, :\text{hiredBy}, a), T(a, \tau, :\text{PubAdmin})$

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

Figure 4: Sample reformulation for Example 12.

is denoted by Views($\mathcal{M}$). Crucially, the extent $\mathcal{E}$ of the mapping set $\mathcal{M}$ is also an extent for the corresponding set of views Views($\mathcal{M}$). In particular, we establish that the certain answers to a BGPQ q on a RIS (Def. 8) are exactly the certain answers of its UBGPQ reformulation $\mathcal{Q}_{c,a}$ seen as a UCQ w.r.t. the relational CQ views obtained from the RIS mappings.

THEOREM 3 (REW-CA CORRECTNESS). *Let $S = \langle O, \mathcal{R}, \mathcal{M}, \mathcal{E}\rangle$ be a RIS and q be a BGPQ. Let $\mathcal{Q}_{c,a}$ be the reformulation of q w.r.t. O and $\mathcal{R}$. Then:*

$$\text{cert}(q, S) = \text{cert}(ubgpq2ucq(\mathcal{Q}_{c,a}), \text{Views}(\mathcal{M}), \mathcal{E})$$

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

This result relies on the soundness and completeness of reformulation (Theorem 1), which implies that cert(q, S) is equal to cert($\mathcal{Q}_{c,a}\langle\emptyset, \emptyset, \mathcal{M}, \mathcal{E}\rangle$); by construction of $ubgpq2ucq(\mathcal{Q}_{c,a})$ and Views($\mathcal{M}$), cert($\mathcal{Q}_{c,a}\langle\emptyset, \emptyset, \mathcal{M}, \mathcal{E}\rangle$) is in turn equal to cert($ubgpq2ucq(\mathcal{Q}_{c,a})$, Views($\mathcal{M}$), $\mathcal{E}$) .

The above theorem leads to the following query answering strategy (Figure 3): reformulate q into a UBGPQ $\mathcal{Q}_{c,a}$ (step (3)), translate it into a UCQ, and provide this UCQ as input to a relational view-based query rewriting engine (step (4)), together with the relational views. The view-based rewriting thus obtained (step (5)) refers to the view symbols corresponding to the mappings, e.g., m_1, m_2 etc.Next, we unfold each such view symbol by replacing it with the body of the query q_1 from the respective mapping (recall that q_1 is expressed in the native language of the data source). This yields an expression built with (relational) unions, joins, selections and projections, on top of a set of source queries. We turn this to a mediator engine, which: interacts with the sources (6) to request the evaluation of the queries q_1 involved in the rewriting, possibly pushing inside them selections, projections, joins etc., applies any remaining operations, such as joins and unions, within the mediator's runtime component, and returns the results thus computed, which are the answers to q (7).

EXAMPLE 12. *Consider again the RIS in Example 9 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 relationship he/she has with some company". Its UBGPQ reformulation, obtained with the technique in [15] (recall Section 2.4.1), seen as a UCQ is shown in Figure 4. Its maximally-contained rewriting based on the views obtained from the RIS mappings, for instance with the Minicon algorithm [41], is:* $q(x, :\text{ceoOf}) \leftarrow m_1(x), m_2(x, y)$. *It is obtained from the second CQ in the above union; this becomes clear when the view symbols are replaced by their bodies:* $q(x, :\text{NatComp}) \leftarrow T(x, :\text{ceoOf}, y_1), T(y_1, \tau, :\text{NatComp}), T(x, :\text{hiredBy}, y_2), T(y_2, \tau, :\text{PubAdmin})$. *Note that the five other CQs cannot be rewritten based on the available views. With the current RIS, this rewriting yields an empty certain answer set to the asked query, i.e.,* cert(q, S) = $\emptyset$, *because the extent of the mappings, hence of the views, is:* $\mathcal{E} = \{m_1(:\text{p}_1), m_2(:\text{p}_2, :\text{a})\}$. *However, if we add* $m_2(:\text{p}_1, :\text{a})$ *to* $\mathcal{E}$, *then* cert(q, S) = $\{\langle :\text{p}_1, :\text{ceoOf}\rangle\}$.

4.2.2 Rewriting Partially-Reformulated Queries using Saturated Mappings as Views: REW and REW-C

Instead of pushing reasoning into the query by reformulating it, we can push it *into the mappings* from which we derive the materialized views. To this aim, we *saturate* the heads of the RIS mappings so that they directly integrate the heterogeneous data sources w.r.t. *both* the explicit and implicit knowledge the RIS has, i.e., by taking into account the ontology O and entailment rules $\mathcal{R}$. Specifically:

DEFINITION 10 (SATURATED MAPPINGS). *Given an RDFS ontology O, the saturation of a set $\mathcal{M}$ of RIS mappings w.r.t. O is:*

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

Recall from Section 2.4.2 that $q_2^{\mathcal{R}_a,O}$ denotes the saturation of q_2 with $\mathcal{R}_a$ and O; it complements the query with all it implicitly asks for w.r.t. $\mathcal{R}_a$ and O.

EXAMPLE 13. *Consider the RIS of Example 9, its saturated mapping heads are (added implicit triples are in gray):*

$m_1 : \quad q_2^{\mathcal{R}_a,O}(x) \leftarrow (x, :\text{ceoOf}, y), (y, \tau, :\text{NatComp})$
$(x, :\text{worksFor}, y), (y, \tau, :\text{Comp})$
$(x, \tau{:}\text{Person}), (y, \tau, :\text{Org})$

$m_2 : \quad q_2^{\mathcal{R}_a,O}(x, y) \leftarrow (x, :\text{hiredBy}, y), (y, \tau, :\text{PubAdmin})$
$(x, :\text{worksFor}, y), (y, \tau, :\text{Org})$
$(x, \tau, :\text{Person})$

To take the saturated ontology into account, we propose two different techniques. The first, called REW, consists of *making the (saturation of) the ontology accessible through specific mappings*, one for each kind of schema triple:

DEFINITION 11 (ONTOLOGY MAPPINGS).
The set of mappings assigned to an 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 a mapping* m_x *is* $\text{ext}(m_x) = \{(\mathsf{s}, \mathsf{o}) \mid (\mathsf{s}, x, \mathsf{o}) \in O^{\mathcal{R}_c}\}$. *The* extent *of* $\mathcal{M}_{O^c}$ *is denoted* $\mathcal{E}_{O^c}$.

Ontology mappings do not fulfill RIS mapping restrictions, they use RDFS properties in their head. Users can not define such mappings as RIS ingredient, because general mappings may lead to incomplete query answering approaches. For this reason, ontology mappings are used in a specific RIS, and they can be considered as RIS mappings for rest of the process. These mappings can be computed offline, and need to be updated when ontology is updated.

Based on the saturated mappings and the ontology mappings, as well as their extensions, query answers can be computed disregarding the entailment rules $\mathcal{R}$ and the ontology

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

Figure 5: Sample rewriting for Example 14.

O at query processing time:

LEMMA 1. *Let $S = \langle O, \mathcal{R}, \mathcal{M}, \mathcal{E} \rangle$ be a RIS and q be a BGPQ. Then:*

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

The main underpinning for this is that reasoning using $\mathcal{R}$ can be split according to $\mathcal{R}_a$ and $\mathcal{R}_c$ [15], which are applied independently on respectively data triples (using the unsaturated ontology) and ontology. It follows that the induced graphs of S and $\langle \emptyset, \emptyset, \mathcal{M}_{O^c} \cup \mathcal{M}^{a,O}, \mathcal{E}_{O^c} \cup \mathcal{E} \rangle$ are the same, hence the certain query answers in these two RIS as well. Recalling from the start of Section 4 the steps involved in RIS query anwering, the above lemma states how to encapsulate reasoning with the ontology O and the rules $\mathcal{R}$, into the ontology mappings and their extension. In particular, turning this (larger) set of mappings $\mathcal{M}_{O^c} \cup \mathcal{M}^{a,O}$ into relational views as in Section 4.2.1, we establish that the certain answers of a BGPQ q on a RIS are exactly the certain answers of q seen as a CQ w.r.t. these views:

THEOREM 4 (REW CORRECTNESS). *Let $S = \langle O, \mathcal{R}, \mathcal{M}, \mathcal{E} \rangle$ be a RIS and q be a BGPQ. Then:*

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

EXAMPLE 14 (REW). *Consider again the RIS in Example 9 and the query q of Example 12 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 based on the views obtained from the RIS saturated *and* ontology *mappings appears in Figure 5.*

If we assume that $\mathcal{E}$ also contains $m_2(:\text{p}_1, :\text{a})$, as we did in Example 12, we obtain again $\text{cert}(q, S) = \{\langle :\text{p}_1, :\text{ceoOf} \rangle\}$*, which results from the evaluation of the first CQ in the above 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.*

Below, we introduce a last approach for certain query anwering on RIS denoted REW-C. Instead of making the saturated ontology explicit in the mappings, we *partially reformulate q using $\mathcal{R}_c$* (thus the -C suffix in its acronym). This corresponds to the first step of the reformulation technique presented in Section 2.4.1. The reformulation w.r.t. $\mathcal{R}_c$ leads to the equality of $\text{cert}(q, \langle O, \mathcal{R}, \mathcal{M}, \mathcal{E} \rangle)$ and $\text{cert}(\mathcal{Q}_c, \langle O, \mathcal{R}_a, \mathcal{M}, \mathcal{E} \rangle)$. We recall that the $\mathcal{Q}_c$ reformulation obtained after this first step does not contain any schema triple anymore: it evaluation only requires data triples. Therefore the saturated mappings are sufficient to answer it. These observations lead to variants of the previous lemma and theorem:

LEMMA 2. *Let $S = \langle O, \mathcal{R}, \mathcal{M}, \mathcal{E} \rangle$ be a RIS, q be a BGPQ and $\mathcal{Q}_c$ its reformulation w.r.t. $O, \mathcal{R}_c$ [15]. Then:*

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

THEOREM 5 (REW-C CORRECTNESS). *Let $S = \langle O, \mathcal{R}, \mathcal{M}, \mathcal{E} \rangle$ be a RIS, q be a BGPQ and $\mathcal{Q}_c$ its reformulation w.r.t. $O, \mathcal{R}_c$. Then:*

$$\text{cert}(q, S) = \text{cert}(\mathit{bgpq2cq}(\mathcal{Q}_c), \text{Views}(\mathcal{M}^{a,O}), \mathcal{E})$$

EXAMPLE 15 (REW-C). *Consider again the RIS in Example 9 and the query q of Example 12. Its $\mathcal{Q}_c$ reformulation 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: $q(x, :\text{ceoOf}) \leftarrow m_1(x), m_2(x, a)$. It is obtained from the first CQ in the above; the second one has no rewriting based on the available RIS views. We obtain a rewriting isomorphic to the one obtained in Example 12, hence the same answers as by the above approaches.

In Fig. 3, REW and REW-C are pictured as follows. Offline (independent of the arrival of a query), relational CQ views are computed from the saturated source-to-RIS mappings (step (E)). In REW, these views are complemented with a special set of views derived from O and the ontological mappings (step (F)). Then, every incoming BGPQ q (step (3')) is transformed into a CQ and rewritten into a UCQ using the views (step (8)). In contrast, REW-C does not consider ontological mappings; reasoning with O takes place at query time, by reformulating q into a BGPQ $\mathcal{Q}_c$ (variant of (3) considering only $\mathcal{R}_c$), which is then transformed into a UCQ (variant of (4)) and rewritten still as a UCQ (variant of (8)) using the set of views obtained through step (E). Once a rewriting is obtained, its optimization and evaluation is delegated to the mediator (steps (9) and (10)), leading to the obtention of query answers (step (11)).

How do our strategies differ? Importantly, since all strategies based on view-based rewriting are correct, *they lead to equivalent UCQ rewritings* as soon as the same set of

mappings is considered. This means that REW-CA and REW-C yield equivalent rewritings. Strategy REW considers the additional set $\mathcal{M}_{O^c}$ of ontology mappings. Hence, the REW rewriting is larger than the one obtained by REW-CA and REW-C for queries over the ontology (i.e., when q contains triples with $\prec_{sc}$, $\prec_{sp}$, $\hookleftarrow_d$, $\hookrightarrow_r$ or a variable in property position); otherwise, $\mathcal{M}_{O^c}$ is not used for query rewriting, and the three strategies yield equivalent results. Rewritings may contain redundancies, which we remove by minimization. It is well-known that equivalent UCQs are isomorphic (i.e., equal up to variable renaming) once minimized. Hence, REW-CA and REW-C *differ in how the view-based rewriting is computed*, or, equivalently, on *the distribution of the reasoning effort on the data and mappings, across various query answering stages*, but they do not differ in how the final rewriting is evaluated (steps (9), (10), (11)).

5. EXPERIMENTAL EVALUATION

We describe our experiments with RIS query answering.

5.1 Experimental settings

Software Our platform is developed in Java 1.8, as follows. Our **RDFDB** (recall Figure 3) is OntoSQL[1], a Java platform providing efficient RDF storage, saturation, and query evaluation on top of an RDBMS [16, 30]; in particular, we used 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 also provides the RDF query reformulation algorithm [15] we rely on.

We rely on the Graal engine [11] for **view-based query rewriting**. Graal is a Java toolkit dedicated to query answering algorithms in knowledge bases with existential rules (also known as tuple-generating dependencies or TGDs). Since the relational view $m(\bar{x}) \leftarrow bgp2ca(\text{body}(q_2))$ corresponding to a mapping m (recall Def. 9) can be seen as a specific existential rule of the form $m(\bar{x}) \rightarrow bgp2ca(\text{body}(q_2))$, 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 [12], a Java-based mediator (or polystore) system. We chose it notably due to its support for joins in the mediator; a recent study [6] shows this often gives it a performance edge over polystores such as [24] which can evaluate a join only by shipping its inputs inside a same data source. The other modules and algorithms described in this paper (notably, our novel query answering methods) were coded in Java 1.8.

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

RIS used We now describe the data, ontology, and mappings used in our experiments.

Our first interest was to study scalability of RIS query answering, in particular in the *relational* setting studied in many prior works. We used the BSBM benchmark **relational data** generator[2] to build databases consisting of **10 relations** named producer, product, offer, review etc. Using two different 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.

Our **RDFS ontologies** O_1 respectively O_2 contain, first, subclass hierarchies of 151, respectively, 2011 product types, which come with DS_1, respectively, DS_2. To these, we add 26 classes and 36 properties, used in 40 subclass, 32 subproperty, 42 domain and 16 range statements.

All-relational RIS We devised two sets $\mathcal{M}_1, \mathcal{M}_2$ of 307, respectively, 3863 mappings, which expose the relational data from $DS_{r,1}$, respectively, $DS_{r,2}$ 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 mappings, partially exposing the results of join queries over the BSBM data; interestingly, these mappings expose incomplete knowledge, in the style of Example 8.

The mapping sets lead to the RIS data triple sets (or 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 RIS Second, we test our approaches in a *heterogeneous data source* setting. 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 $DS_{j,1}, DS_{j,2}$; we call $DS_{r,1}, DS_{r_2}$ the relational sources storing 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 data source heterogeneity.* The same holds for S_2 and S_4.

Queries We picked 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); 7 among them query the data *and* the ontology (recall Example 3), a capability which competitor systems lack (see Section 6). Table 3 reports four query properties impacting query answering performance: the number of triples, the number of reformulations on the ontology (a good indicator for the difficulty of answering such large, often redundant union queries, recall Example 12), and its number of answers on the two RIS groups (S_1, S_3 and S_2, S_4). We have 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

[1] `https://ontosql.inria.fr`

[2] `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	N_{REF}	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	N_{REF}	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	N_{REF}	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	N_{REF}	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 3: Characteristics of the queries used in our experiments.

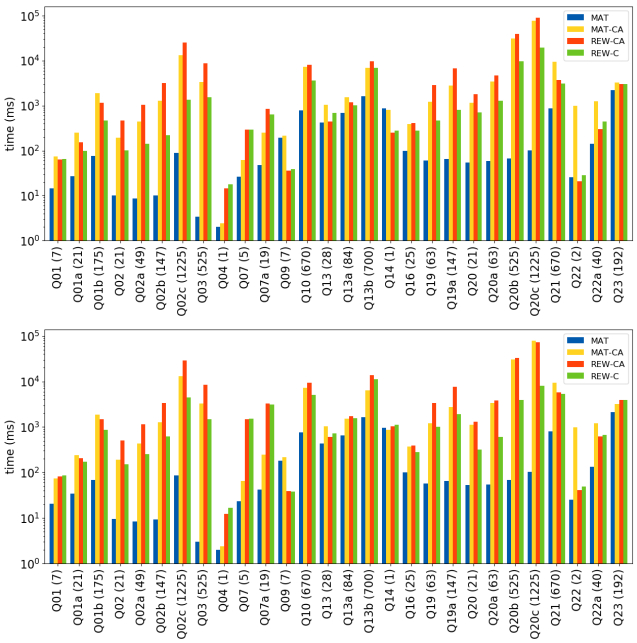

Figure 6: Query answering times on the smaller RIS S_1 (top, all-relational) and S_3 (bottom, heterogeneous).

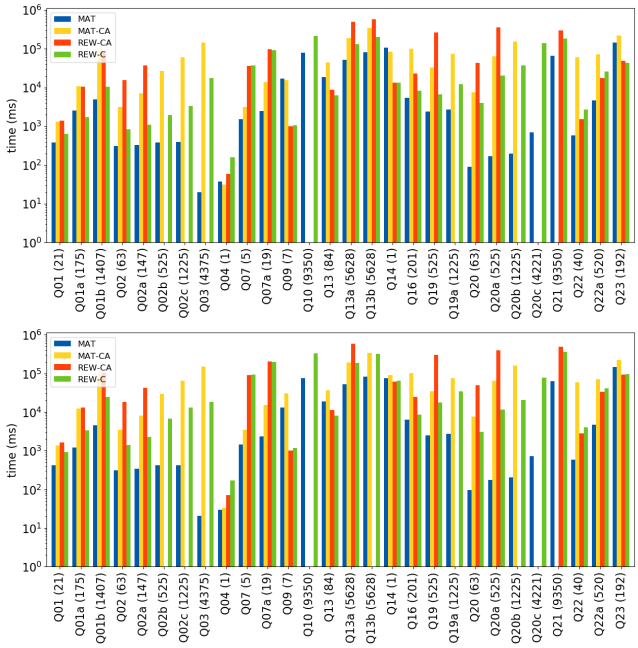

Figure 7: Query answering times on the larger RIS S_2 (top, all-relational) and S_4 (bottom, heterogeneous).

are available online[3].

5.3 Query answering performance

We did not include REW in our study, for the following reason. For *queries which do not involve the ontology*, REW coincides with REW-C (the ontology mappings can be ommitted). In contrast, for queries which *do* involve the ontology, REW yields much larger reformulations (e.g., Figure 5, compared with the one within Example 15), thus view-based rewriting within REW is much less efficient.

Figure 6 depicts the query answering times, on the smaller RIS, of MAT and MAT-CA described in Section 4.1, REW-CA from Section 4.2.1, respectively, REW-C from Section 4.2.2. The number of reformulations N_{REF} appears in parentheses after each query number, in the labels along the x axis. Given that S_1, S_3 have the same RIS data triples, the MAT and MAT-CA strategies coincide among these two RIS. Figure 7 shows the corresponding times for the largest RIS S_2 and S_4; the same observations apply. Both Figure 6 and 7 use a logarithmic axis for the time.

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

Strategy performance analysis We see that MAT *is the fastest in all cases*, which is unsurprising, since a large part of the work (materializing the RIS data triples and saturating them) has been performed *before (outside of)* query answering (and thus is not reported in the graph). For S_1, S_3, materialization took $1.2 \cdot 10^5$ ms and saturating it $1.49 \cdot 10^5$ ms more, whereas for S_2, S_4, these times are 14h46 ($5.31 \cdot 10^7$ ms), respectively, 1h28 ($5.28 \cdot 10^6$ ms).

Not only these are orders of magnitude more than all query answering times; recall also that materializing $G^{\mathcal{M}}_{\mathcal{E}}$ requires maintaining it when the underlying data changes, and its saturation $(G^{\mathcal{M}}_{\mathcal{E}} \cup O)^{\mathcal{R}}$ needs a second level of maintenance. Thus, MAT is not practical when data sources change.

MAT-CA, which does not saturate the RIS data triples, only

[3] https://gitlab.inria.fr/mburon/org/blob/master/projects/het2onto-benchmark/bsbm/

incurs the first out of the two above levels of maintenance. However, it is always slower than MAT, with the difference reaching *4 orders of magnitude* e.g., for Q_{03} on S_4; MAT-CA failed to complete within a time limit of 10 minutes, e.g., for Q_{10}, Q_{20c} and Q_{21} (Figure 7). MAT-CA is slow on queries with very large reformulations: 4375, 9350, 4221 and 9350, respectively, for the four mentioned above. In contrast, the difference between MAT-CA and the fast MAT is less than an order of magnitude for queries with few reformulations, such as Q_{04}, Q_{07}, Q_{09} and Q_{23}. The comparison of MAT and MAT-CA confirms prior observations in RDF query answering works, e.g., [30, 17]. The difference in our RIS setting, where data is not natively RDF, is that both incur a high cost for materializing and possibly maintaining the RIS graph $G_{\mathcal{E}}^{\mathcal{M}}$.

Among the strategies based on view-based rewriting, the performance of REW-CA is close to that of MAT-CA for some queries, e.g., Q_{01}, Q_{01a}, Q_{01b}, and gets up to one order of magnitude faster than MAT-CA e.g., for Q_{14}, Q_{16}, Q_{23} in Figure 7. However, REW-CA is more often slower than MAT-CA, up to one order of magnitude for Q_{02}, Q_{02a}, Q_{19}, Q_{20} in the same Figure. REW-CA also fails to complete for many queries (missing yellow bars in Figure 7), in close correlation with the increased number of reformulations. This is particularly visible e.g., comparing the times for Q_{02}, Q_{02a}, Q_{02b} and Q_{02c} there: REW-CA is increasingly slow, up to going beyond the time limit for the last two. This is because REW-CA involves rewriting a query that is syntactically very large; given the complexity of view-based query rewriting, the time becomes prohibitively high.

Finally, 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 speedup is noticeable even on the smaller RIS S_1, S_3 (Figure 6) for Q_{02a}. As a consequence, REW-C completes successfully in all scenarios we study. This is because by only *partially* reformulating queries to be rewritten, REW-C keeps query rewriting costs under control.

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 6 and 7 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 and MAT-CA). 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 REW-CA and REW may bring.

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 all-relational RIS. Part of this is due to the overhead of marshalling data across system boundaries; we owe the rest to imperfect optimization within Tatooine. Overall, the comparison demonstrates that 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, after the initial materialization and saturation cost has been paid. In contrast, if these are expected to change, REW-C smartly combines partial reformulation and view-based query rewritings; it is capable of dynamically and robustly computing 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, REW-C is the best strategy for dynamic RIS.

6. RELATED WORK AND CONCLUSION

As explained in the introduction, our work pursues a data integration goal, i.e., providing access to a set of data sources under a single unified schema [43, 39].

Ontologies have been used to integrate relational or heterogeneous data sources in mediators [43] in LAV style based on description logics [36, 3] or their combination with Datalog [29, 31], with applications in many areas, e.g., banking [**?**], oil rig management [**?**] etc. However, none of these approaches integrate data as RDF graphs with RDFS ontologies, supporting queries over the data and ontology.

Semantics have been used at the integration level since e.g., [21] for SGML and soon after for RDF [8, 9]; data is considered represented and stored in a flexible object-oriented model, thus no mappings are used. The source and the integrated schemas are aligned semi-automatically, trying to determine the best correspondences based on the available ontologies, and asking users to solve unclear situations. In contrast, we consider heterogeneous sources, and, following the OBDA approach, rely on (GLAV) mappings.

XML trees have also been used as the integration format in systems integrating heterogeneous data sources in LAV style in [37, 22, 7]. *Virtual* views were specified in a pivot relational model (enriched with integrity constraints) to describe how the content of each data source contributes to the global schema. View-based rewriting against this relational model lead to queries over the virtual views, which were then translated back into queries to be evaluated on each individual source. Ontological knowledge was not exploited here, nor in classical relational view-based integration [27, 32], whereas we include them in our framework to enrich the set of results that a user may get out of the system, by making available the application knowledge they encapsulate.

Our work follows the *Ontology-Based Data Access (OBDA)* paradigm introduced in [40] and further developed in e.g. [28, 34, 33, 20]. This paradigm was conceived to facilitate access to relational data by mappings to an ontology expressed in a dialect of the DL-Lite description logic family. Classical systems consider GAV mappings and the logic DL-Lite$_{\mathcal{R}}$ underpinning the OWL 2 QL profile of the W3C ontological language OWL 2. Mature systems include Mastro[4] and Ontop[5]. Extending OBDA to non-relational data sources is a recent topic. In particular, the extension to JSON data sources was proposed in [14, 13].

Compared to this line of work, our novelty is (*i*) to extend the relational setting to heterogeneous data sources, (*ii*) to rely on RDF as the integration model, in particular allowing applications to query both the integrated data and the ontology, (*iii*) to deal with GLAV mappings instead of the more restricted GAV mappings, and (*iv*) to propose novel query answering techniques dedicated to RDF. To the best of our

[4] `http://obdasystems.com/mastro/`

[5] `https://ontop.inf.unibz.it/`

knowledge, our system is the first OBDA system providing access to heterogeneous data, thanks to a mediating component which performs query evaluation on multiple sources. With respect to (*ii*), an RDFS ontology can be seen as a restricted DL-Lite ontology, but on the other hand we allow to query both ontology and data triples, including the properties that relate two resources or the type of a resource, while only *instances of a given property/given type* are available in a DL setting. With respect to (*iii*), we recall that GLAV mappings are most flexible, and they increase the expressive power of the integration system by allowing to model existing information though unavailable from the sources, i.e., *incomplete* information (recall Example 9).
Finally, with respect to (*iv*), we propose novel query answering techniques, which rely on (G)LAV view-based query rewriting. REW-CA relies on a preliminary query reformulation step: while this approach is typical of OBDA systems, in our RDF setting we use our recent reformulation algorithm [15]. REW-C relies on mapping saturation, which totally avoids both data materialization and query reformulation. It turns out that this technique can be seen as a generalization to GLAV mappings of the $\mathcal{T}$-mapping technique introduced in [42] to optimize query rewriting in a classical OBDA context. In this latter work, the original set of GAV mappings is completed with new ones, which also encapsulate information 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'. Clearly, as soon as mappings are GLAV and not GAV, adding new mappings is not sufficient: instead, we saturate the mapping heads. Our mapping saturation technique could easily 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.

7. REFERENCES

[1] RDF 1.1 Concepts and Abstract Syntax.

[2] RDF 1.1 Semantics.

[3] N. Abdallah, F. Goasdoué, and M. Rousset. DL-LITER in the light of propositional logic for decentralized data management. In *IJCAI*, 2009.

[4] S. Abiteboul and O. M. Duschka. Complexity of answering queries using materialized views. ACM Press, 1998.

[5] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[6] R. Alotaibi, D. Bursztyn, A. Deutsch, I. Manolescu, and S. Zampetakis. Towards Scalable Hybrid Stores: Constraint-Based Rewriting to the Rescue. In *SIGMOD 2019 - ACM SIGMOD International Conference on Management of Data*, Amsterdam, Netherlands, June 2019.

[7] B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Querying XML Sources Using an Ontology-Based Mediator. In *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE*, pages 429–448. Springer Berlin Heidelberg, 2002.

[8] B. Amann and I. Fundulaki. Integrating ontologies and thesauri to build RDF schemas. In *ECDL*, 1999.

[9] B. Amann, I. Fundulaki, and M. Scholl. Integrating ontologies and thesauri for RDF schema creation and metadata querying. *Int. J. on Digital Libraries*, 3(3), 2000.

[10] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[11] J.-F. Baget, M. Leclère, M. Mugnier, S. Rocher, and C. Sipieter. Graal: A toolkit for query answering with existential rules. In *Rule Technologies: Foundations, Tools, and Applications - 9th International Symposium, RuleML 2015, Berlin, Germany, August 2-5, 2015, Proceedings*, pages 328–344, 2015.

[12] R. Bonaque, T. D. Cao, B. Cautis, F. Goasdoué, J. Letelier, I. Manolescu, O. Mendoza, S. Ribeiro, X. Tannier, and M. Thomazo. Mixed-instance querying: a lightweight integration architecture for data journalism. In *VLDB*, New Delhi, India, Sept. 2016.

[13] E. Botoeva, D. Calvanese, B. Cogrel, J. Corman, and G. Xiao. A generalized framework for ontology-based data access. In *AI*IA 2018 - Advances in Artificial Intelligence - XVIIth International Conference of the Italian Association for Artificial Intelligence, Trento, Italy, November 20-23, 2018, Proceedings*, pages 166–180, 2018.

[14] E. Botoeva, D. Calvanese, B. Cogrel, M. Rezk, and G. Xiao. OBDA over non-relational databases. In *Proceedings of the 10th Alberto Mendelzon International Workshop on Foundations of Data Management, Panama City, Panama, May 8-10, 2016*, 2016.

[15] M. Buron, F. Goasdoué, I. Manolescu, and M. Mugnier. Reformulation-based query answering for RDF graphs with RDFS ontologies. In *The Semantic Web - 16th International Conference, ESWC 2019, Portorož, Slovenia, June 2-6, 2019, Proceedings*, pages 19–35, 2019.

[16] D. Bursztyn, F. Goasdoué, and I. Manolescu. Optimizing reformulation-based query answering in RDF. In *EDBT*, 2015.

[17] D. Bursztyn, F. Goasdoué, and I. Manolescu. Teaching an RDBMS about ontological constraints. *PVLDB*, 9(12), 2016.

[18] A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In *Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA*, 2009.

[19] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao. Ontop: Answering SPARQL queries over relational databases. *Semantic Web*, 8(3), 2017.

[20] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, and M. Ruzzi. Using OWL in Data Integration. In *Semantic Web Information*

Management, pages 397–424. Springer, 2010.

[21] V. Christophides, M. Doerr, and I. Fundulaki. A semantic network approach to semi-structured documents repositories. In *ECDL*, 1997.

[22] A. Deutsch and V. Tannen. MARS: A System for Publishing XML from Mixed and Redundant Storage. In *VLDB*, 2003.

[23] A. Doan, A. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, Waltham, MA, 2012.

[24] J. Duggan, A. J. Elmore, M. Stonebraker, M. Balazinska, B. Howe, J. Kepner, S. Madden, D. Maier, T. Mattson, and S. B. Zdonik. The BigDAWG polystore system. *SIGMOD*, 44(2):11–16, 2015.

[25] S. El Hassad, F. Goasdoué, and H. Jaudoin. Learning commonalities in SPARQL. In *ISWC*, 2017.

[26] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: Data models and languages. *J. Intell. Inf. Syst.*, 8(2), 1997.

[27] M. R. Genesereth, A. M. Keller, and O. M. Duschka. Infomaster: An information integration system. In *SIGMOD*, 1997.

[28] M. Giese, A. Soylu, G. Vega-Gorgojo, A. Waaler, P. Haase, E. Jiménez-Ruiz, D. Lanti, M. Rezk, G. Xiao, Ö. L. Özçep, and R. Rosati. Optique: Zooming in on big data. *IEEE Computer*, 48(3), 2015.

[29] F. Goasdoué, V. Lattès, and M. Rousset. The use of CARIN language and algorithms for information integration: The PICSEL system. *Int. J. Cooperative Inf. Syst.*, 9(4), 2000.

[30] F. Goasdoué, I. Manolescu, and A. Roatis. Efficient query answering against dynamic RDF databases. In *EDBT*, 2013.

[31] F. Goasdoué and M. Rousset. Answering queries using views: A KRDB perspective for the semantic web. *ACM Trans. Internet Techn.*, 4(3), 2004.

[32] A. Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4), Dec. 2001.

[33] D. Lanti, G. Xiao, and D. Calvanese. Cost-driven ontology-based data access. In *ISWC*, 2017.

[34] D. Lembo, J. Mora, R. Rosati, D. F. Savo, and E. Thorstensen. Mapping analysis in ontology-based data access: Algorithms and complexity. In *ISWC*, 2015.

[35] M. Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, pages 233–246, 2002.

[36] A. Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *J. Intell. Inf. Syst.*, 5(2), 1995.

[37] I. Manolescu, D. Florescu, and D. Kossmann. Answering XML queries on heterogeneous data sources. In *VLDB*, 2001.

[38] M. Mugnier. Ontological query answering with existential rules. In *Web Reasoning and Rule Systems - 5th International Conference, RR 2011, Galway, Ireland, August 29-30, 2011. Proceedings*, 2011.

[39] M. T. Özsu and P. Valduriez. *Distributed and Parallel Database Systems (3rd. ed.)*. Springer, 2011.

[40] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. Data Semantics*, 10, 2008.

[41] R. Pottinger and A. Y. Halevy. Minicon: A scalable algorithm for answering queries using views. *VLDB J.*, 10, 2001.

[42] M. Rodriguez-Muro, R. Kontchakov, and M. Zakharyaschev. Ontology-based data access: Ontop of databases. In *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, pages 558–573, 2013.

[43] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3), 1992.