Obi-Wan: Ontology-Based RDF Integration of Heterogeneous Data

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

Obi-Wan: Ontology-Based RDF Integration of Heterogeneous Data

Maxime Buron[1] François Goasdoué[2] Ioana Manolescu[1] Marie-Laure Mugnier[3]
[1]Inria and Institut Polytechnique de Paris, France
[2]Univ. Rennes, CNRS, IRISA, France [3]Univ. Montpellier, LIRMM, Inria, France

ABSTRACT

The proliferation of digital data sources in many domains brings a new urgency to the need for tools which allow to flexibly query heterogeneous data (relational, JSON, key-values, graphs etc.) Traditional data integration systems fall into two classes: *data warehousing*, where all data source content is materialized in a single repository, and *mediation*, where data remains in their original stores and all data can be queried through a *mediator*.

We propose to demonstrate Obi-Wan, a novel mediator following the Ontology-Based Data access (OBDA) paradigm. Obi-Wan integrates data sources of many data models under an interface based on RDF graphs and ontologies (classes, properties, and relations between them). The novelty of Obi-Wan is to combine maximum integration power (GLAV mappings, see below) with the highest query answering power supported by an RDF mediator: RDF queries not only over the data but also over the integration ontologies [6]. This makes it more flexible and powerful than comparable systems.

1 INTRODUCTION

Prior mediator approaches can be classified according to two main dimensions (see Table 1 that references some of the most prominent works). A first dimension concerns the **data model and query language** provided by the mediator to its applications.

(*i*) Many mediators mimic a single database, and expose to their users one data model and its query language, e.g., relational and SQL, or XML and XPath/XQuery. More recent polystore systems support side-by-side different (data model, query language) pairs. These database-style mediators appear in the **DB** row in Table 1.

(*ii*) Ontology-based mediators provide a view of the data sources as a set of *classes* and *relationships*, also endowed with a set of *semantic constraints*, or *ontology*. In such systems, users ask conjunctive (relational) queries; answering them involves not only evaluation over the data (as in DB mediators), but also reasoning on the data with the help of ontologies. This mediation approach is also commonly termed *Ontology-Based Data Access* (OBDA) [20], with ontologies expressed in Description Logics (DL, in short). Works following this approach are listed in the row we label **CQ** in Table 1.

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

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

		Mappings		
		GAV	**LAV**	**GLAV**
Model	**DB**	[12, 14, 15]	[2, 12, 18]	[10]
	CQ	[20, 21]	[1, 16, 19]	[9]
	SPARQL-data	[8]	[23]	[11]
	SPARQL	[7, 22]		Obi-Wan [6]

Table 1: Positioning of Obi-Wan in the related literature.

A second dimension is **how source (or local) schemas are connected to the global (integration) schema** using *mappings* [13]. There are three types of mappings, each corresponding to a column in Table 1. Global-As-View, or **GAV** mappings define each element of the global schema, e.g., each global relation, as a view over the local schemas. A query over the global (virtual) schema is easily transformed into a query over the local schemas by *unfolding* each global schema relation, i.e., replacing it with its definition. In contrast, Local-As-View (**LAV**) mappings define elements of the local schemas as views over the global one. Query answering then requires *rewriting the query with the views* describing the local sources [17]. Global-Local-As-View (**GLAV**) data integration generalizes both GAV and LAV. A GLAV mapping pairs a query q_1 over one or several local schemas with a query q_2 over the global schema having the same answer variables. The semantics is: for each answer of q_1, the integration system exposes the data comprised in a corresponding answer of q_2. *GLAV maximizes flexibility* or, equivalently, *integration expressive power*: unlike LAV, a GLAV mapping may expose only part of a given source's data, and may combine data from several sources; unlike GAV, a GLAV mapping may include joins or complex expressions over the global schema.

We propose to demonstrate Obi-Wan, a novel **GLAV mediator system supporting SPARQL queries over the data and the ontology**, described in a recent work [6]. As Table 1 shows, Obi-Wan is *the first capable of integrating multiple data sources of heterogeneous data models through GLAV mappings, for SPARQL querying over the data and the ontology*. A benefit of using GLAV is the ability to support a form of *incomplete information*, naturally present in RDF through the so-called *blank nodes*, in the virtual RDF graph exposed by the mediator (see Section 2).

Our closest competitors only support GAV mappings, even though some support more expressive ontology and/or query languages [7, 22]. Some formal OBDA frameworks based on GLAV mapping, e.g., [9] lack known implementations.

Below, we introduce our query answering setting, our novel query

answering techniques, then the demonstration scenarios.

2 RDF INTEGRATION SYSTEM (RIS)

We consider integrating data from *heterogeneous sources* (each with its own data model and query language) into a *virtual RDF graph*. This graph consists of an RDFS ontology, and of data triples derived from the sources by means of GLAV *mappings*. A mapping specifies (*i*) which source data is made available in the integration system, and (*ii*) how to expose it as RDF triples using classes and properties from the ontology. Users can query the (virtual) RDF graph containing this data by means of conjunctive SPARQL queries; query answers need to reflect not only the data exposed in the graph, but also the reasoning enabled by the ontology.

Star Wars example scenario Consider the (partial) ontology:

$O = \{$(:uses, $\hookleftarrow_d$, :Character), (:uses, $\hookrightarrow_r$, :FictionalObj),
(:LightSaber, $\prec_{sc}$, :FictionalObj), (:StarShip, $\prec_{sc}$, :FictionalObj),
(:StarFighter, $\prec_{sc}$, :StarShip), (:usesWeapon, $\prec_{sp}$, :uses)
(:pilotOf, $\prec_{sp}$, :uses), (:pilotOf, $\hookrightarrow_r$, :StarShip)$\}$

where $\prec_{sc}$, $\prec_{sp}$, $\hookleftarrow_d$ and $\hookrightarrow_r$ stand for the RDFS properties subClassOf, subPropertyOf, domain and range, respectively. This ontology states that characters use fictional objects, some of which are light sabers or starships; starfighters are specific starships. Using weapons or piloting are two specific ways of using fictional objects, in the latter case the object is a starship.

A *mapping* is of the form $m = q_1(\bar{x}) \rightsquigarrow q_2(\bar{x})$ where the *mapping body* q_1 is a query on a data source (in SQL, XQuery, etc.), and the *mapping head* q_2 is a query over the RDF graph; q_1 and q_2 have the same answer variables. The *extension* of m is the set of answer tuples of q_1 on a data source D that m integrates, transformed into tuples of RDF resources. Intuitively, m specifies that the extension of m is exposed to the system as the result of q_2.

Example 2.1 (Mappings). We consider the following two mappings:
m_1 with head $q_2^1(x) \leftarrow (x, $:pilotOf$, y), (y, \tau, $:StarFighter) and
m_2 with head $q_2^2(x, y) \leftarrow (x, $:usesWeapon$, y), (y, \tau, $:LightSaber),
where τ is a shortcut for the property rdf:type. Assume the body of m_1 retrieves a value v translated into the IRI :p. Then, the extension of m_1 is: ext(m_1) = $\{V_{m_1}($:p$)\}$, where V_{m_1} is a view relation name. Similarly, we assume the extension of m_2 is ext(m_2) = $\{V_{m_2}($:p, :a$)\}$.

Given a set of RIS mappings $\mathcal{M}$, the *extent* $\mathcal{E}$ of $\mathcal{M}$ is the union of the mappings' extensions, i.e., $\mathcal{E} = \bigcup_{m \in \mathcal{M}}$ *ext(m)*. The data triples *induced* by $\mathcal{M}$ and $\mathcal{E}$ define an RDF graph $G_{\mathcal{E}}^{\mathcal{M}}$ containing *all the data which is exposed (can be queried) through a RIS*. Because we use GLAV mappings, RIS data triples may include fresh blank nodes, as exemplified below; these correspond to the non-answer variables, i.e., *incomplete information*, allowed in GLAV mapping heads.

Example 2.2. Let $\mathcal{M} = \{m_1, m_2\}$ for the mappings introduced above; the extent of $\mathcal{M}$ is $\mathcal{E} = \{V_{m_1}($:p$), V_{m_2}($:p, :a$)\}$. The RIS data triples they lead to are:

$G_{\mathcal{E}}^{\mathcal{M}} = \{$(:p, :pilotOf, $\_{:}b_c$), ($\_{:}b_c$, τ, :StarFighter),
(:p, :usesWeapon, :a), (:a, τ, :LightSaber)$\}$

These triples are obtained by instantiating the answer variables in m_1 and m_2 by values appearing in the extent $\mathcal{E}$. The first and second triples contain the blank node $\_{:}b_c$, introduced by the non-answer variable y in the head of m_1.

An **RDF Integration System (RIS)** is a tuple $S = \langle O, \mathcal{R}, \mathcal{M}, \mathcal{E} \rangle$. It allows to access (query) the data triples induced by the mappings $\mathcal{M}$ and their extent $\mathcal{E}$; it also allows to *reason* on this data, with the ontology O and the reasoning power of the *entailment rule set* $\mathcal{R}$ for RDFS ontologies [6]. Importantly, $\mathcal{R}$ is partioned into two subsets: $\mathcal{R}_a$ derives new *data* triples, while $\mathcal{R}_c$ derives new *ontology* triples. A sample $\mathcal{R}_a$ rule is $(\mathsf{p}_1, \prec_{sp}, \mathsf{p}_2), (\mathsf{s}, \mathsf{p}_1, \mathsf{o}) \rightarrow (\mathsf{s}, \mathsf{p}_2, \mathsf{o})$, stating that if a graph asserts that p_1 is a subproperty of p_2, and a resource s_1 has the property p_1 with value o_1, then s_1 has the property p_2 with value o_1. $\mathcal{R}_c$ rules state that $\prec_{sc}$ and $\prec_{sp}$ are transitive; they also allow deducing new triples with property $\hookleftarrow_d$ or $\hookrightarrow_r$, e.g., the triple (:pilotOf, $\hookrightarrow_r$, :FictionalObj) from (:pilotOf, $\hookrightarrow_r$, :StarShip) and (:StarShip, $\prec_{sc}$, :FictionalObj). The (finite) process of enriching a graph with all the triples it entails through $\mathcal{R}$ is called *saturation*.

The **query answering problem** we consider is answering conjunctive RDF queries[1] in a RIS. The *certain answers* of q on S, denoted by cert(q, S), result from the evaluation of q on the saturation of the RDF graph $O \cup G_{\mathcal{E}}^{\mathcal{M}}$, restricted to tuples fully built from source values (i.e., excluding *incomplete* tuples containing blank nodes generated by mappings).

Example 2.3 (Certain answers). Consider the RIS $S = \langle O, \mathcal{R}, \mathcal{M}, \mathcal{E} \rangle$ introduced in the previous examples and the query
$q(x, y) \leftarrow (x, y, z), (z, \tau, t), (y, \prec_{sp}, $:uses$), (t, \prec_{sc}, $:StarShip$)$,
$(x, $:uses$, a), (a, \tau, $:LightSaber$)$
asking "Who uses a light saber, and how is she/he using starships?" Then cert(q, S) = $\{\langle$:p, :pilotOf$\rangle\}$. This answer is obtained by matching q on the triples (:p, :pilotOf, $\_{:}b_c$), ($\_{:}b_c$, τ, :StarFighter), (:pilotOf, $\prec_{sp}$, :uses), (:StarFighter, $\prec_{sc}$, :StarShip), (:p, :uses, :a), (:a, τ, :LightSaber) in the saturation of $O \cup G_{\mathcal{E}}^{\mathcal{M}}$, where (:p, :uses, :a) is derived using the above-mentioned $\mathcal{R}_a$ rule.

3 QUERY ANSWERING STRATEGIES

Since we adopt a mediator-style approach, the RIS data triples $G_{\mathcal{E}}^{\mathcal{M}}$ are not materialised, hence the saturation of $O \cup G_{\mathcal{E}}^{\mathcal{M}}$ cannot be computed to answer queries as defined above. Instead, queries are rewritten in terms of the remote heterogeneous sources, based on the RIS ontology O, reasoning power $\mathcal{R}$ and mappings $\mathcal{M}$. We present three query answering strategies, which differ in how the ontological reasoning is incorporated: we may have *all*, *some* or *no* reasoning performed at query time, as outlined in Figure 1.

In all strategies, RIS mappings of the form $m = q_1(\bar{x}) \rightsquigarrow q_2(\bar{x})$ are seen as relational LAV views of the form $V_m(x) \leftarrow rel(q_2)(\bar{x})$, where $rel(q_2)$ is the translation of q_2 into a conjunctive query (CQ). The two first strategies make use of *query reformulation*, which injects relevant ontological knowledge into the query: given an ontology O and entailment rules $\mathcal{R}$, an RDF query q is reformulated into a union of queries Q, such that for *any* set G of data triples, the evaluation of Q on G yields the same answers as the evaluation of q on the saturation of $G \cup O$ by $\mathcal{R}$. We use the reformulation technique introduced in [5].

All reasoning at query time (REW-CA). The first strategy starts with reformulating the query q, based on the RIS ontology O and entailment rules $\mathcal{R} = \mathcal{R}_c \cup \mathcal{R}_a$, into a query $Q_{c,a}$ (step (1) in Figure 1). Since RIS data triples are not materialized, we rewrite $Q_{c,a}$, seen as a union of CQs (UCQ), using the RIS mappings $\mathcal{M}$ seen

[1] Commonly called Basic Graph Pattern Queries (BGPQs) in the literature.

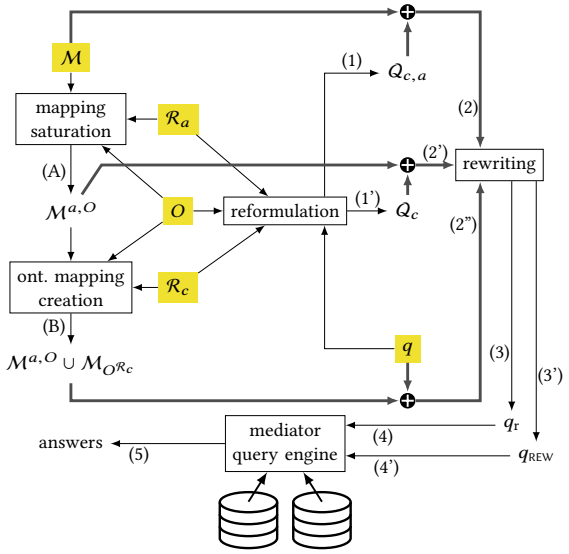

Figure 1: Outline of query answering strategies.

$$
\begin{aligned}
Q_{c,a} = \quad & q(x, \text{:pilotOf}) \leftarrow && (x, \text{:pilotOf}, z), (z, \tau, \text{:StarFighter}),\\
& && (x, \text{:uses}, a), (a, \tau, \text{:LightSaber})\\
& \cup\, q(x, \text{:pilotOf}) \leftarrow && (x, \text{:pilotOf}, z), (z, \tau, \text{:StarFighter}),\\
& && (x, \text{:usesWeapon}, a), (a, \tau, \text{:LightSaber})\\
& \cup\, q(x, \text{:pilotOf}) \leftarrow && (x, \text{:pilotOf}, z), (z, \tau, \text{:StarFighter}),\\
& && (x, \text{:pilotOf}, a), (a, \tau, \text{:LightSaber})\\
& \cup\, q(x, \text{:usesWeapon}) \leftarrow && (x, \text{:usesWeapon}, z), (z, \tau, \text{:StarFighter}),\\
& && (x, \text{:uses}, a), (a, \tau, \text{:LightSaber})\\
& \cup\, q(x, \text{:usesWeapon}) \leftarrow && (x, \text{:usesWeapon}, z), (z, \tau, \text{:StarFighter}),\\
& && (x, \text{:usesWeapon}, a), (a, \tau, \text{:LightSaber})\\
& \cup\, q(x, \text{:usesWeapon}) \leftarrow && (x, \text{:usesWeapon}, z), (z, \tau, \text{:StarFighter}),\\
& && (x, \text{:pilotOf}, a), (a, \tau, \text{:LightSaber})
\end{aligned}
$$

Figure 2: Reformulation of q in Example 3.1.

as relational LAV views (step (2)). This yields a relational rewriting q_r over the integrated sources (step (3)), whose evaluation in a mediator engine provides the desired answers (steps (4) and (5)).

Example 3.1 (REW-CA). Consider again the RIS and query q from Example 2.3. The reformulation $Q_{c,a}$ of q is shown in Figure 2. Then $Q_{c,a}$ is turned into a UCQ, using a single ternary relation name t (for triple), i.e., any triple (s, p, o) becomes $t(s, p, o)$. This UCQ is finally rewritten using mappings seen as LAV views: m_1 is seen as $V_{m_1}(x) \leftarrow t(x, \text{:pilotOf}, y), t(y, \tau, \text{:StarFighter})$ and m_2 as $V_{m_2}(x, y) \leftarrow t(x, \text{:usesWeapon}, y), t(y, \tau, \text{:LightSaber})$. It turns out that only the second conjunctive query in $Q_{c,a}$ yields a CQ that can be rewritten. The obtained (maximally-contained) rewriting on the integrated sources is: $q_r(x, \text{:pilotOf}) \leftarrow V_{m_1}(x), V_{m_2}(x, y)$, which yields the answer $\langle$:p, :pilotOf$\rangle$ on $\mathcal{E} = \{V_{m_1}(\text{:p}), V_{m_2}(\text{:p}, \text{:a})\}$.

Some reasoning at query time (REW-C). The second strategy has the best performances and is a main contribution of OBI-WAN. First, it reformulates (step (1')) the query q based on O and $\mathcal{R}_c$ only (not $\mathcal{R} = \mathcal{R}_c \cup \mathcal{R}_a$ as previously). The obtained reformulation Q_c yields the expected answers when evaluated on the RIS data triples *saturated with O and $\mathcal{R}_a$* (see details in [5]). Again, since these RIS triples are not materialized, hence cannot be saturated, Q_c is rewritten using the mappings *saturated with O and $\mathcal{R}_a$*, seen as LAV views. These saturated mappings, denoted $\mathcal{M}^{a,O}$, are obtained (step (A)) from the original ones by adding to their head queries (q_2)

$$
\begin{aligned}
& q(x, \text{:pilotOf}) \leftarrow && V_{m_1}(x), V_{m_{<_{sp}}}(\text{:pilotOf}, \text{:uses}),\\
& && V_{m_{<_{sc}}}(\text{:StarFighter}, \text{:StarShip}), V_{m_2}(x, a)\\
& \cup\, q(x, \text{:pilotOf}) \leftarrow && V_{m_1}(x), V_{m_{<_{sp}}}(\text{:pilotOf}, \text{:uses}),\\
& && V_{m_{<_{sc}}}(\text{:StarShip}, \text{:StarShip}), V_{m_2}(x, a)\\
& \cup\, q(x, \text{:pilotOf}) \leftarrow && V_{m_1}(x), V_{m_{<_{sp}}}(\text{:pilotOf}, \text{:uses}),\\
& && V_{m_{<_{sc}}}(\text{:FictionalObj}, \text{:StarShip}), V_{m_2}(x, a)\\
& \cup\, 15 \textit{ other BGPQs...}
\end{aligned}
$$

Figure 3: Sample rewriting for Example 3.3.

all the implicit RIS data triples they entail w.r.t. O and $\mathcal{R}_a$. Hence, the data triples induced by the saturated mappings $\mathcal{M}^{a,O}$ and the extent $\mathcal{E}$ are exactly the data triples in the saturation of the graph induced by O, the original mappings $\mathcal{M}$ and $\mathcal{E}$, i.e., $O \cup G^{\mathcal{M}}_{\mathcal{E}}$. Then, the partially reformulated query Q_c is rewritten using $\mathcal{M}^{a,O}$ (step (2')) and the resulting query (step (3)) is evaluated as in the first strategy (steps (4) and (5)). Importantly, mappings are saturated offline and the result has to be updated only when some mapping changes. This technique limits both the reasoning effort at query time *and* the syntactic complexity (size) of the reformulated UCQ to rewrite, hence the time needed to obtain a rewriting q_r over the data sources; this translates into reducing the query answering time by up to two orders of magnitude [6].

Example 3.2 (REW-C). The mappings in $\mathcal{M}^{a,O}$ have the following *heads* (where added implicit triples are in blue):

$$
\begin{aligned}
(m_1)\ & q_2^{\mathcal{R}_a,O}(x) \leftarrow && (x, \text{:pilotOf}, y), (y, \tau, \text{:StarFighter}),\\
& && (x, \text{:uses}, y), (y, \tau, \text{:StarShip}), (y, \tau, \text{:FictionalObj}),\\
& && (x, \tau, \text{:Character})\\
(m_2)\ & q_2^{\mathcal{R}_a,O}(x, y) \leftarrow && (x, \text{:usesWeapon}, y), (y, \tau, \text{:LightSaber}),\\
& && (x, \text{:uses}, y), (y, \tau, \text{:FictionalObj}),\\
& && (x, \tau, \text{:Character})
\end{aligned}
$$

The reformulation Q_c of q is:

$$
\begin{aligned}
& q(x, \text{:pilotOf}) \leftarrow && (x, \text{:pilotOf}, z), (z, \tau, \text{:StarFighter}),\\
& && (x, \text{:uses}, a), (a, \tau, \text{:LightSaber})\\
& \cup\, q(x, \text{:usesWeapon}) \leftarrow && (x, \text{:usesWeapon}, z), (z, \tau, \text{:StarFighter}),\\
& && (x, \text{:uses}, a), (a, \tau, \text{:LightSaber})
\end{aligned}
$$

Rewriting Q_c using the views obtained from $\mathcal{M}^{a,O}$ yields, as previously, $q_r(x, \text{:pilotOf}) \leftarrow V_{m_1}(x), V_{m_2}(x, y)$, obtained from the first union term in Q_c (the second term has no rewriting).

No reasoning at query time (REW). Finally, in the third strategy, the mappings are saturated offline as above (step (A)) in order to model all explicit and implicit RIS data triples. Moreover, these mappings are complemented with another set of mappings, denoted $\mathcal{M}_{O^{\mathcal{R}_c}}$ (step (B)), comprising all the explicit and implicit ontology triples w.r.t. O and $\mathcal{R}$; since only $\mathcal{R}_c$ rules entail new ontology triples, $O^{\mathcal{R}}$ is actually equal to $O^{\mathcal{R}_c}$. This second set of mappings is also computed offline and is updated upon ontology updates. A query q does not have to be reformulated at all. It just needs to be rewritten using the mappings $\mathcal{M}^{a,O} \cup \mathcal{M}_{O^{\mathcal{R}_c}}$ seen as LAV views (step (2")) to obtain, as above, a rewriting q_{REW} over the data sources (step (3')) evaluated through (steps (4') and (5)).

Example 3.3 (REW). Figure 3 shows part of the (maximally-contained) rewriting of q. This rewriting is much larger than those from the two previous techniques, which is due to the additional ontology mappings. As previously, cert$(q, S) = \{\langle$:p, :pilotOf$\rangle\}$, which results here from the evaluation of the first CQ in the rewriting; the

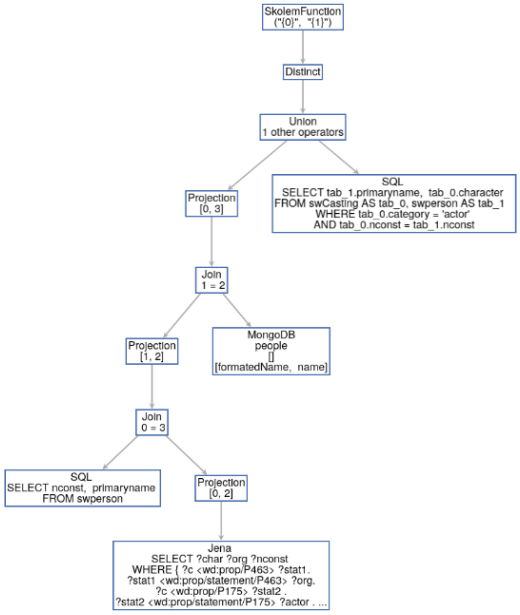

Figure 4: Query plan on data sources in Star Wars scenario.

other CQs yield empty results because some required $\prec_{sc}$ or $\prec_{sp}$ contraints do not hold in the ontology.

How do our strategies compare? They all produce the same answers, however they do not all compute the same view-based rewritings. Indeed, REW considers the additional set $\mathcal{M}_{O^{\mathcal{R}_c}}$ of ontology mappings. Hence, for queries over the ontology, i.e., featuring in a property position $\prec_{sc}$, $\prec_{sp}$, $\hookleftarrow_d$, $\hookrightarrow_r$, or a variable, a REW rewriting is larger than a REW-CA or REW-C rewriting and, to be answered, requires the additional ontology source. In contrast, REW-CA and REW-C yield logically equivalent rewritings; we minimize them both to avoid possible redundancies, thus they even become identical (up to variable renaming). Hence, REW-CA and REW-C do not differ in how these rewritings are evaluated. Instead, they differ in how the rewritings are *computed*, or, equivalently, on *the distribution of the reasoning effort on the data and mappings, across various query answering stages*. As our experiments show, given the computational complexity of view-based query rewriting [17], this difference has a significant impact on their performance.

4 DEPLOYMENT AND SCENARIOS

Obi-Wan is developed in Java 1.8 on top of Tatooine [4], a mediator system handling JSON, relational, key-value and RDF data (based on MongoDB, Postgres, Redis, and Jena TDB, respectively); Tatooine also provides physical query operators (selections, joins etc.) within the mediator. For query rewriting, Obi-Wan relies on Graal [3], a toolkit for query answering in knowledge bases.

Our demonstration will introduce a set of RISs, comprising RDF, relational and JSON sources, together with their ontologies. For each (RIS, query) pair, the query reformulation/rewriting stages and mappings transformations are visualized in step-by-step fashion through a sequence of dedicated visualizations, until the Tatooine query execution plan which computes the final results (see Figure 4). Demo attendees will also be able to edit the queries, mappings etc.

Scenarios Our first scenario comprises a set of queries on the above mentioned *Star Wars* RIS. It integrates three data sources: IMDB relational movie data; RDF triples about Star Wars characters from WikiData (movie characters are no longer present in IMDB, thus WikiData is crucial here); and JSON data from Star Wars API.

Secondly, we will show Obi-Wan on a larger RIS (108M induced triples) used in the experiments of [6], based on an extended version of *BSBM*, with relational and JSON data sources. Queries on this RIS are challenging as some lead to very large and complex rewritings; they illustrate the performance difference between our different query answering methods.

Details on our RISs (data, mappings, query plans...) are available at: https://obi-wan.saclay.inria.fr/

REFERENCES

[1] N. Abdallah, F. Goasdoué, and M. Rousset. 2009. DL-LITER in the Light of Propositional Logic for Decentralized Data Management. In *IJCAI*.
[2] R. Alotaibi, D. Bursztyn, A. Deutsch, I. Manolescu, and S. Zampetakis. 2019. Towards Scalable Hybrid Stores: Constraint-Based Rewriting to the Rescue. In *SIGMOD*.
[3] J.-F. Baget, M. Leclère, M. Mugnier, S. Rocher, and C. Sipieter. 2015. Graal: A Toolkit for Query Answering with Existential Rules. In *RuleML*.
[4] R. Bonaque, T. D. Cao, B. Cautis, F. Goasdoué, J. Letelier, I. Manolescu, O. Mendoza, S. Ribeiro, X. Tannier, and M. Thomazo. 2016. Mixed-instance querying: a lightweight integration architecture for data journalism. In *VLDB*.
[5] M. Buron, F. Goasdoué, I. Manolescu, and M. Mugnier. 2019. Reformulation-Based Query Answering for RDF Graphs with RDFS Ontologies. In *ESWC*.
[6] M. Buron, F. Goasdoué, I. Manolescu, and M. Mugnier. 2020. Ontology-Based RDF Integration of Heterogeneous Data. In *EDBT*.
[7] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao. 2017. Ontop: Answering SPARQL queries over relational databases. *Semantic Web* 8, 3 (2017).
[8] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, and D. F. Savo. 2011. The MASTRO system for ontology-based data access. *Semantic Web* 2, 1 (2011).
[9] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, and M. Ruzzi. 2009. Using OWL in Data Integration. In *Semantic Web Information Management – A Model-Based Perspective*. Springer.
[10] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. 2012. Query Processing under GLAV Mappings for Relational and Graph Databases. *PVLDB* 6, 2 (2012).
[11] G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati. 2018. Using Ontologies for Semantic Data Integration. In *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years*. Vol. 31. Springer, Cham.
[12] A. Deutsch and V. Tannen. 2003. MARS: A System for Publishing XML from Mixed and Redundant Storage.. In *VLDB*.
[13] A. Doan, A. Halevy, and Z. G. Ives. 2012. *Principles of Data Integration*. Morgan Kaufmann, Waltham, MA.
[14] J. Duggan, A. J. Elmore, M. Stonebraker, M. Balazinska, B. Howe, J. Kepner, S. Madden, D. Maier, T. Mattson, and S. B. Zdonik. 2015. The BigDAWG Polystore System. *SIGMOD* 44, 2 (2015).
[15] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Widom. 1997. The TSIMMIS Approach to Mediation: Data Models and Languages. *J. Intell. Inf. Syst.* 8, 2 (1997).
[16] F. Goasdoué, V. Lattès, and M. Rousset. 2000. The Use of CARIN Language and Algorithms for Information Integration: The PICSEL System. *IJCIS* (2000).
[17] A. Y. Halevy. 2001. Answering Queries Using Views: A Survey. *The VLDB Journal* 10, 4 (Dec. 2001).
[18] I. Manolescu, D. Florescu, and D. Kossmann. 2001. Answering XML Queries on Heterogeneous Data Sources. In *VLDB*.
[19] S. Nadal, K. Rabbani, O. Romero, and S. Tadesse. 2019. ODIN: A Dataspace Management System. (2019).
[20] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. 2008. Linking Data to Ontologies. *J. Data Semantics* 10 (2008).
[21] M. Rodriguez-Muro, R. Kontchakov, and M. Zakharyaschev. 2013. Ontology-Based Data Access: Ontop of Databases. In *ISWC*.
[22] J. F. Sequeda, M. Arenas, and D. P. Miranker. 2014. OBDA: Query Rewriting or Materialization? In Practice, Both!. In *ISWC*.
[23] G. Smits, O. Pivert, H. Jaudoin, and F. Paulus. 2014. AGGREGO SEARCH: Interactive Keyword Query Construction. In *EDBT*.