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

To cite this version:
Maxime Buron, François Goasdoué, Ioana Manolescu, Marie-Laure Mugnier. Obi-Wan: Ontology-Based RDF Integration of Heterogeneous Data. VLDB 2020 - 46th International Conference on Very Large Data Bases, Aug 2020, Tokyo, Japan. hal-02921434

HAL Id: hal-02921434
https://inria.hal.science/hal-02921434
Submitted on 25 Aug 2020

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

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

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 $q_2$ over one or several local schemas with a query $q_g$ over the global schema having the same answer variables. The semantics is: for each answer of $q_2$, the integration system exposes the data comprised in a corresponding answer of $q_g$. 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).

<table>
<thead>
<tr>
<th>Model</th>
<th>GAV</th>
<th>LAV</th>
<th>GLAV</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>[12, 14, 15]</td>
<td>[2, 12, 18]</td>
<td>[10]</td>
</tr>
<tr>
<td>CQ</td>
<td>[20, 21]</td>
<td>[1, 16, 19]</td>
<td>[9]</td>
</tr>
<tr>
<td>SPARQL-data</td>
<td>[8]</td>
<td>[23]</td>
<td>[11]</td>
</tr>
</tbody>
</table>

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

\[ O = \{(\text{uses}, \leftarrow_d, \text{Character}), (\text{uses}, \leftarrow_r, \text{FictionalObj})\} \]

(R;LightSaber, \leftarrow_{sc};FictionalObj), (StarShip, \leftarrow_{sc};FictionalObj),

(StarFighter, \leftarrow_{sc};StarShip), (\text{usesWeapon}, \leftarrow_{sp};\text{uses})

(\text{pilotOf}, \leftarrow_{sp};\text{uses}), (\text{pilotOf}, \leftarrow_{r};\text{StarShip}) \]

where \( \leftarrow_{sc} \) and \( \leftarrow_{sp} \) are transitive. This ontology states that characters use fictional objects, some of which are light sabers or starships; starships 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 \( m = q_1(\bar{x}) \approx q_2(\bar{x}) \) 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:

\[ m_1 \text{ with head } q_1^2(x) \leftarrow (x, \text{pilotOf}, y), (y, r, \text{StarFighter}) \quad \text{and} \quad m_2 \text{ with head } q_2^2(x, y) \leftarrow (x, \text{usesWeapon}, y), (y, r, \text{LightSaber}) \]

where \( r \) is a shortcut for the property rdf:type. Assume the body of \( m_1 \) retrieves a value \( v \) translated into the IRI \( \text{pilot} \). Then, the extension of \( m_1 \) is:

\[ \text{ext}(m_1) = \{ (\text{pilot}(p)) \mid (\text{pilot}(p), \text{Character}, \text{pilot}(p), \text{Character}) \in D \} \]

Similarly, we assume the extension of \( m_2 \) is:

\[ \text{ext}(m_2) = \{ (\text{usesWeapon}(p, a)) \mid \text{LightSaber}(p), \text{Character}(p), \text{LightSaber}(p), \text{Character}(p) \in D \} \]

Given a set of RIS mappings \( M \), the extent \( E \) of \( M \) is the union of the mappings’ extensions, i.e., \( E = \bigcup_{m \in M} \text{ext}(m) \). The data triples induced by \( M \) and \( E \) define an RDF graph \( G_E^M \) 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 \( M = \{ m_1, m_2 \} \) for the mappings introduced above; the extent of \( M \) is:

\[ E = \{ \text{pilot}(p), \text{Character}(p), \text{pilot}(p), \text{Character}(p) \} \]

The RDF data triples they lead to are:

\[ G_E^M = \{ (\text{pilot}(p), \text{Character}(p), \text{pilot}(p), \text{Character}(p)) \} \]

These triples are obtained by instantiating the answer variables \( m_1 \) and \( m_2 \) by values appearing in the extent \( E \). The first and second triples contain the blank node \( \_bc \), introduced by the non-answer variable \( y \) in the head of \( m_1 \).

An RDF Integration System (RIS) is a tuple \( S = (O, R, M, E) \). It allows to access (query) the data triples induced by the mappings \( M \) and their extent \( E \); it also allows to reason on this data, with the ontology \( O \) and the reasoning power of the entailment rule set \( R \) for RDFS ontologies [6]. Importantly, \( R \) is partitioned into two subsets: \( R_0 \) derives new data triples, while \( R_c \) derives new ontology triples.

A sample \( R_0 \) rule is \((p_1, \leftarrow_{sp}, p_2), (s, p_1, o) \rightarrow (s, p_2, o)\), stating that if a graph asserts that \( p_1 \) is a subproperty of \( p_2 \), and a resource \( s \) has the property \( p_1 \) with value \( o \), then \( s \) has the property \( p_2 \) with value \( o \). \( R_c \) rules state that \( \leftarrow_{sc} \) and \( \leftarrow_{sp} \) are transitive; they also allow deducing new triples with property \( \leftarrow_{d} \) or \( \leftarrow_{r} \), e.g., the triple \((\text{pilotOf}, \leftarrow_r, \text{FictionalObj})\) from \((\text{pilotOf}, \leftarrow_r, \text{StarShip})\) and \((\text{StarShip}, \leftarrow_{sc}, \text{FictionalObj})\). The (finite) process of enriching a graph with all the triples it entails through \( R \) 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 \( O \cup G_E^M \), 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 \( S = (O, R, M, E) \) introduced in the previous examples and the query

\[ q(x, y) \leftarrow (x, y), (z, r, t), (y, \leftarrow_{sp}, \text{uses}), (t, \leftarrow_{sc}, \text{StarShip}), (\text{x}, \text{uses}, a), (a, r, \text{LightSaber}) \]

asking “Who uses a light saber, and how is she/he using starships?” Then cert(q, S) = \{ (p, \text{pilot}(p), \text{Character}(p), \text{pilot}(p), \text{Character}(p)) \}. This answer is obtained by matching \( q \) on the triples \((\text{pilotOf}, \leftarrow_{bc}, \text{Character}(p), \text{pilotOf}, \leftarrow_{bc}, \text{Character}(p))\), \((\text{usesWeapon}, \leftarrow_{bc}, \text{Character}(p), \text{usesWeapon}, \leftarrow_{bc}, \text{Character}(p))\), \((\text{a}, \text{r}, \text{LightSaber})\) in the saturation of \( O \cup G_E^M \), where \((\text{p}, \text{uses}, a)\) is derived using the above-mentioned \( R_0 \) rule.

3 QUERY ANSWERING STRATEGIES

Since we adopt a mediator-style approach, the RIS data triples \( G_E^M \) are not materialised, hence the saturation of \( O \cup G_E^M \) cannot be computed to answer queries as defined above. Instead, queries are rewritten in terms of the remote heterogeneous sources, based on the RIS ontology \( O \), reasoning power \( R \) and mappings \( M \). We present three query answering strategies, which differ in how 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 \( m = q_1(\bar{x}) \approx q_2(\bar{x}) \) are seen as relational LAV views of the form \( V_{qm}(x) \leftarrow \text{rel}(q_2(\bar{x})) \), where \( \text{rel}(q_2) \) 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 \( R \), 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 \( O \cup G \) by \( R \). 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 \( R = R_0 \cup R_c \), into a query \( Q_{r,c} \) (step (1) in Figure 1). Since RIS data triples are not materialized, we rewrite \( Q_{r,c} \) as seen a union of CQs (UCQ), using the RIS mappings \( M \) seen

---

1Commonly called Basic Graph Pattern Queries (BGPQs) in the literature.
which yields the answer \( q^A \) for \( q \left((x, \text{pilotOf}, z), (z, r, \text{StarFighter}), (x, \text{uses}, a), (a, r, \text{LightSaber}) \right) \). However, since \( q \) is rewritten using the mappings \( M_{a, O} \), the context of the rewritten query \( Q_{c,a} \) is turned into a UCQ, using a single ternary relation \( M_{a, O} \cup_{\mathcal{R}_a} \mathcal{M} \) as relational LAV views (step (2)). This yields a relational rewriting \( q^{r_{rew}} \) for \( q \) as relational views \( \mathcal{O}^{r_{rew}} \left( (x, \text{pilotOf}, z), (z, r, \text{StarFighter}), (x, \text{uses}, a), (a, r, \text{LightSaber}) \right) \).

Example 3.2 (Rewr-C). The mappings in \( M_{a, O} \) have the following heads (where added implicit triples are in blue):

\[
\begin{align*}
&m_1 \quad q_{2_{a, O}}^{R_{a}}(x, y) \leftarrow (x, \text{pilotOf}, y), (y, r, \text{StarFighter}), (x, \text{uses}, y), (y, r, \text{StarFighter}), (x, \text{uses}, y), (x, \tau, \text{Character})
\end{align*}
\]

The reformulation \( Q_{c, a} \) of \( q_1 \) as relational views \( \mathcal{O}^{r_{rew}} \) (step (3)) is obtained from the first union term in \( Q_{c} \) (the second term has no rewriting).

No reasoning at query time (Rewr-C). 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 \( M_{O_{rew}} \). The query rewriting, \( Q_{c} \), is obtained by rewriting \( q_{1} \) as relational views \( \mathcal{O}_{rew} \) (step (3)).
mentioned

Scenarios

Our first scenario comprises a set of queries on the above Demo attendees will also be able to edit the queries, mappings etc. query execution plan which computes the final results (see Figure 4).

through a sequence of dedicated visualizations, until the Tatooine each (RIS, query) pair, the query reformulation/rewriting stages and relational and JSON sources, together with their ontologies. For Our demonstration will introduce a set of RISs, comprising RDF, toolkit for query answering in knowledge bases.

the mediator. For query rewriting, Tatooine 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 BSIM, 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/

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 $\mathcal{M}_{\mathcal{SE}}$ of ontology mappings. Hence, for queries over the ontology, i.e., featuring in a property position $\approx_{SC}$, $\approx_{SP}$, $\approx_{d}$, $\approx_{r}$, 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

Figure 4: Query plan on data sources in Star Wars scenario. other CQs yield empty results because some required $\approx_{SC}$ or $\approx_{SP}$ constraints do not hold in the ontology.

REFERENCES