Semantic Mediator Querying
Béatrice Bouchou Markhoff, Cheikh Niang

To cite this version:
Béatrice Bouchou Markhoff, Cheikh Niang. Semantic Mediator Querying. IDEAS 2014, Jul 2014, Porto, Portugal. 10.1145/2628194.2628218 . hal-01198496

HAL Id: hal-01198496
https://hal.science/hal-01198496
Submitted on 18 Feb 2022

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.
Semantic Mediator Querying

Béatrice Bouchou
Université François-Rabelais de Tours
LI EA 6300, Tours, France
beatrice.bouchou@univ-tours.fr

Cheikh Niang
Université François-Rabelais de Tours
LI EA 6300, Tours, France
cheikh.niang@univ-tours.fr

ABSTRACT
We present the whole querying process of our ontology-based data integration proposal, that we call Semantic Mediator. The global schema (a TBox) is composed of the source schemas (also Tboxes) and a taxonomy, which links the sources to each other. The querying process is based on the global-schema’s structure and consists of three steps: global query rewriting, source querying and global answer building. We describe the overall distributed system and the query-rewriting algorithm. Then we present an application of such a semantic mediation, the Personae project, which is for enabling historians to share their prosopographic data from the Middle Ages and the Renaissance.

Categories and Subject Descriptors
H.2.4 [Systems]: Query Processing, Distributed Databases

Keywords
Semantic Data Integration, Distributed Query Answering

1. INTRODUCTION
We devised an ontology-based data integration system whose query resolution component is detailed in this paper. Data integration is a broad research topic, that has received for many years the attention of researchers in databases and knowledge representation, and it is now revisited with the growth of the Semantic Web. As defined in [11], there are two main approaches for data integration, the data warehouse (the source databases are loaded in the warehouse) and the mediation. The mediation approach allows information to be retrieved dynamically from original databases at query time. It provides a unified global query-interface and relies on mappings between the global schema and each of the local source schemas. These mappings are used to rewrite the global query into a union of queries that match local schemas. They are directed, either from entities in the global schema to entities in the local sources (“Global As View” (GAV) mappings), or from entities in the local sources to the global schema (“Local As View” (LAV) mappings). LAV mappings require more sophisticated inferences to resolve a query on the global schema than GAV mappings, but they make it easier to add new data sources to the mediation framework.

We call our proposal Semantic Mediator because it is a mediation, in which the schemas are ontologies, more precisely they are the conceptual part of ontologies. Our semantic mediator is an application of the ontology-based data access (OBDA) paradigm, which has been proposed as a data integration solution that offers an efficient access to large quantities of data stored in relational databases, via a conceptual model of the data. Its principles, as illustrated on the left of Figure 1, have been introduced in [5, 3, 15], and then fully implemented as reported in [9, 17]. While leaving the RDBMs the tasks of efficient storage, maintenance and querying of the data, it allows (i) several relational databases to be integrated, and (ii) their querying to be enriched with the ontological knowledge.

In an OBDA system, the global schema $\mathcal{G}$ of a classical mediator data integration system $\mathcal{J} = (\mathcal{G}, \mathcal{S}, \mathcal{M})$ (as defined in [11]) is a TBox of the DL-Lite Description Logics [3]. As analyzed in [4], using a conceptual model rather than a logical one for the global schema, allows the users to easily manage their queries while the logical schema of each source is still used for its strenghts: optimizations of the storage, optimizations of the querying, etc. The declarative approach based on $\mathcal{M}$, set of semantic mappings between the conceptual global schema and the logical database schema implies that, when global and local schemas evolve, only those mappings must be updated. However, the implementation of a classical OBDA system architecture requires (i) the construction of a consensual ontology to represent the domain of integration, which is known as a difficult task [23], and (ii) the design of the mappings between this ontology and the heterogeneous sources’ relational databases.

The query-answering process detailed in this paper is based on a more flexible architecture, illustrated on the right of Figure 1. It consists in keeping all the previously listed advantages while limiting the difficulty of constructing the consensual global ontology. In our architecture, each source is an OBDA system. With existing tools such as Ontop\textsuperscript{1} [17] or Mastro\textsuperscript{2} [9], installing an OBDA system on an existing database consists in (i) building the lightweight ontology that represents local data and (ii) defining the mappings

\textsuperscript{1}http://ontop.inf.unibz.it/
\textsuperscript{2}http://www.dis.uniroma1.it/quonto/
between this ontology and the local RDB. This is not such a
difficult task for a local database administrator who knows
her data well, compared to the task of building a global on-
tology from several relational databases, such as reported
in [19]. As discussed in [23], it is simpler to obtain one on-
tology for each data source and then to relate these, either
in a peer-to-peer network or in a "hybrid" architecture that
introduces a global level, such as our proposed global ontol-
yogy.

Moreover, our proposal includes the management of dis-
tribution. To the best of our knowledge, the existing OBDA
systems rely on tools for federated relational databases in
order to consider that the OBDA system comprises a single
relational data source. By contrast, as presented in Sec-
tion 2, our proposal uses the Sesame\(^3\) architecture in order
to implement a truly distributed mediator system.

The paper is organized as follows: in Section 2 we intro-
duce the general architecture of our mediation system. In
Section 3 we present the global schema’s structure and we
show in detail how this structure allows us to perform the
query-answering task. Then we present our use-case in Sec-
tion 4.

2. MEDIATOR ARCHITECTURE

An overview of our mediator system architecture is shown
in Figure 2. It consists of two main layers. The lower layer
is that of sources: each one incrementally joins the system
in order to collaborate with others. The upper layer is that
of the mediator, which includes the global schema of the
system. One important part of this global schema is a set of
semantic mappings that link the mediator to the sources in-
volved. The mediator is also composed of a query resolution
module which allows to locate relevant sources to be queried
to answer a given user request. The results returned from
the sources are combined and stored in a temporary reposi-
tory in order to allow the user to analyze them.

2.1 Source Layer

The source layer of our mediator architecture is the set
of sources that incrementally join the semantic mediation
system. Each source is autonomous and stores its data,
which can be very large, in relational databases (RDBs).
Indeed, we consider that relational database technology is
the best support for managing local data and it is also the
most widely used. In particular, it offers an efficient access
to data. In accordance with OBDA principles, we assume that
the local RDBs are linked to a local ontology, which provides
their conceptual view. This local ontology can be designed
by experts to represent the data managed by the source, or it
can be automatically generated from the database by using
suitable tools such as RDBtoOnto\(^5\). Whatever the local
ontology building method (automatic or manual), we com-
pute from this local ontology the relevant parts to be shared
by the source with respect to the domain of interest in which
the mediation system is desired. We have proposed in [14,
13] an approach that uses a domain reference ontology\(^4\) as a
background knowledge support. With this reference ontol-
yogy, our proposed approach selects in the source the knowl-
edge fragments that must be shared in the semantic medi-
ation system. The process, based on a combination of well
established lexical-based algorithms for ontology alignment,
is supervised by the local database administrator [14, 13],
and its result consists in the agreement ontology depicted
in the sources of Figure 2. This agreement ontology is com-
posed of a relevant subset, for the application domain, of the
source’s ontology, plus a set of semantic mappings between
concepts from this subset and those of the domain reference
ontology. This agreement ontology represents the concep-
tual access point to the source’s data, which is used during
interactions between the mediator and the source.

In order to offer access to the data via the agreement
ontology, the local database administrator must implement
OBDA mappings [5, 3, 15]. These mappings establish links
between the local ontology items and some SQL queries to
select the actual values in the local relational databases for
dynamically populating the ontology’s instances at query
time. The OBDA mappings can be automatically gener-
ated when the local ontology is automatically built from
the database, otherwise they must be manually specified.
The framework that we use for each local OBDA system is
Ontop, used as a SPARQL endpoint through the Sesame
server\(^2\). We also use Sesame to give the mediator re-

to access to the agreement ontology of each source. To
sum up, a source which would like to join the integration
process must firstly have an ontology to represent its data.
Next, it must use our agreement-ontology-building tool to
generate its agreement ontology. Afterwards, it must use
Ontop in order to implement the OBDA mappings. And fi-
nally, it has to register with the mediator (cf. sources direc-
tory in Figure 2) by giving its SPARQL endpoints: one for
its agreement repository, one for its interrogation repository

\(\text{http://www.openrdf.org/}\)

\(\text{http://www.openrdf.org/}\)
expressed in terms of the global ontology, into a set of queries expressed in terms of source ontologies, (ii) compute the sub-queries that must be sent to the sources involved in the original query and send each sub-query to the corresponding source, and (iii) retrieve the partial answers from sources and combine them into a temporary repository. This repository can be seen as a data warehouse of results, that can be graphically visualized, giving the users a comprehensive view about the global answer built from source answers. As we have already published the algorithms devised for building the agreement ontology and to integrate an agreement ontology in the global ontology, in this paper we focus on the query processing, which is presented in greater detail in the following section.

3. GLOBAL SCHEMA STRUCTURE AND QUERY ANSWERING

3.1 Structure Definition

We present in Figure 4 an example with two sources S1 and S2 which illustrates the structure of the global schema of our mediator system. At the top, we have the taxonomy of concepts which allows us to link the sources to each other. Remember that this taxonomy is extracted from a reference domain ontology. The middle part of Figure 4 contains subsumption relations between source concepts and those of the taxonomy: these are the mappings output of the agreement-ontology building performed by each source. Lastly, the lower part presents the source’s agreement-ontologies.

The global schema is a TBox of DL-Lite_A. Indeed, our query-rewriting algorithm relies on the same conceptual framework as the well-established OBDA systems of Calvanese et al., i.e., (i) the DL-Lite_A [3, 15], which belongs to the family of Description Logics (DLs) [1], (ii) conjunctive queries and union of conjunctive queries, and (iii) Consistent and PerfectRef algorithms [3, 18]. Precisely, the global schema is specified in the following definition.

Definition 1. - Global Schema $T_g$
The global schema $T_g$ is a DL-Lite$_A$ TBox that is composed of the following TBoxes:

- **The TBox $T_i$**, which (i) is a taxonomy of concepts of the considered application domain and (ii) serves to link the sources. $T_i$ has two important properties:
  - **Property 1**: $T_i$ is an atomic TBox, i.e., it contains positive inclusions $A_1 \sqsubseteq A_2$, or negative inclusions $A_1 \sqsubseteq \neg A_2$, where $A_1$ and $A_2$ are atomic concepts.
  - **Property 2**: if $B$ is a concept of $T_i$, then $B$ subsumes (directly or transitively) at least one source concept.

- **The set $S'$ of TBoxes $T_i$**, one for each source $S_i$. $T_i$ is the agreement ontology (the part of the source ontology that is involved in the mediation).

- **The set $A$ of TBoxes $A_i$**, one for each source $S_i$. $A_i$ contains the mappings between $T_i$ and the taxonomy, i.e., a set of assertions $B_i \sqsubseteq C$ where $B_i$ is a concept of $T_i$ and $C$ is a concept of $T_i$.

This global schema has some important properties to notice. Firstly, it is possible to have a taxonomy $T_i$ with several disconnected hierarchies if the reference ontology from which it is extracted contains disconnected hierarchies itself. As concepts of $T_i$ do not come from a source, but are taken from the reference ontology of the mediation domain, by construction each concept of this taxonomy is a parent or an ancestor of a source concept. Thus, if a query is about a concept of $T_i$ then there is always at least one source that can give an answer.

Secondly, the assertions of the mappings $A_i$ associate to the taxonomic concepts a union of source concepts, without conjunctions. So, we can say that sources are "loosely coupled" by the taxonomy.

Lastly, the TBoxes $T_i$ contain concepts, roles and attributes that represent data in the sources $S_i$. Notice that $T_i$ may be different from the complete TBox representing all the data in a source $S_i$, as $T_i$ denotes the part that is involved in the mediation process. For instance, in the Personae mediator described in Section 4, the database Bude is an example of a source that does not provide all its content to the mediation process. $T_i$ is the output of the agreement-ontology building process, together with the mappings $A_i$.

All global schema’s items may be queried using a global query. If we consider the more formal notation $J = (G, M, S)$ used in Definition 2, each of its three components is illustrated in the right part of Figure 1, in the following way: $G$ is denoted Global TBox in the figure, $M$ is represented by the mappings and $S$ is the set of source ontology parts that are involved in the mediation.

**Definition 2. Mediator System**

Our mediator system is a triplet $J = (G, M, S)$, where:

- **The global schema $G$** is $T_g$, defined in Definition 1.

- **The set of mappings $M$** is $M_G \cup M_L$, where:
  - $M_G$ is represented by $T_i \cup A$ (with $T_i$ and $A = \bigcup A_i$ specified in Definition 1), i.e. the taxonomy that allows us to relate each query atom of $T_i$’s alphabet to one or several atoms of one or several source alphabets defined by $T_i$.
  - $M_L$ is represented by the $T_i$, belonging to $T_g$: each query atom using the alphabet of a source $S_i$ exists in $T_i$ and may eventually appear in other source TBoxes.

- **The set of sources $S$** is equal to $S'$ defined in Definition 1.
It can be noted that our global schema $T_g$ is structured so that it contains all the relevant knowledge fragments of the mediator system, including the mappings and the sources schemas, which is an original feature, compared to classical data-integration frameworks. Our aim is to facilitate the query processing of the mediator through this global schema. Let us still consider the example in Figure 4: with mappings $\mathcal{M}_{GAV}$ we can easily determine that, if a global query atom involves the concept $A$, then the concepts $B$, $H$, $E$ of $T_1$ and $B$, $E$, $J$ of $T_2$ should be queried. In the same way, thanks to mappings $\mathcal{M}_{LAV}$, a global query atom involving $a_{e}$ will lead to query items of $S_1$ and $S_2$, if there is in both sides a concept that has this attribute and shares a common ancestor with the concept on the other side. This is verified by our adaptation of MiniCon, presented in Algorithm 1.

### 3.2 Query Algorithm

Algorithm 1 performs the Query Reformulation and Query Decomposition stages illustrated in Figure 2. The user, or the application developer, is given a view of the global schema where the different parts (taxonomy / sources) are clearly identified. For the global schema in Figure 4, we give in Figure 5 an abstract idea of such a view.

![Figure 5: View of the Global Schema.](image)

We should consider several scenarios for querying the global schema in order to query data sources:

- **(A)** The user queries only concepts of the taxonomy.
- **(B)** The user queries concepts, roles and attributes of a given source, and says that she wants only answers from this source.
- **(C)** The user queries concepts, roles and attributes of a given source, without saying that she wants to limit answers to this source.
- **(D)** The user queries concepts, roles and attributes that do not appear in the same source.
- **(E)** The user queries concepts of the taxonomy, together with roles and attributes (that are in some sources).

Considering the example in Figure 5, concerning Scenario (A), if the user queries the concept $A$, with the query $q(x) \leftarrow A(x)$, then she should get answers given by concepts $B$, $H$, $E$ of $T_1$ and concepts $B$, $E$, $J$ of $T_2$. In the same way, if she queries the concept $E$ then she must obtain answers from the concept $E$ of $T_1$ and the concept $E$ of $T_2$. With Scenario (B), the original query can be transmitted to the source without changes. However in Scenario (C), the user should receive answers corresponding to the queried concepts, roles and attributes, but she may also receive answers coming from other sources. For instance, if she queries $H$, $R_1$ and $E$, since $E$ is both in $T_1$ and $T_2$ and also $T_1$, then she will receive answers from $T_2$. In Scenario (D), for instance if the user queries $H$, $G$ and $a_{k1}$ she will receive answers from $S_1$ (via $T_1$) concerning $H$ and $a_{k1}$, and also from $S_2$ (via $T_2$) concerning $G$. Finally, with Scenario (E), for instance if the user queries $D$ together with $a_{e}$ and $a_{j2}$, then she will receive answers from $S_1$ (via $T_1$) concerning instances of concept $E$ with its attribute $a_{e}$ and answers from $S_2$ (via $T_2$) concerning its instances of concept $E$ with its attribute $a_{j2}$ and its instances of concept $J$ with its attribute $a_{j2}$. LAV mappings are important here.

The previously described scenarios are concretely achieved through the query resolution steps that we present in Algorithm 1. Input is a conjunctive query $q(x) \leftarrow \text{conj}(x, y)$ on the global schema $T_g$. Each query $q$ on the global schema $T_g$ as well as all queries from the reformulations of $q$ w.r.t $T_g$ are of the form $q(x) \leftarrow c_t, c_s$, where: $c_t = \text{conj}(x_t, y_t)$ is a conjunction of atoms querying the taxonomy $T_1$, and $c_s = \text{conj}(x_s, y_s)$ is a conjunction of atoms querying the sources. In other words, $c_t$ is composed of “taxonomic atoms” (they are always unary), and $c_s$ is composed of “source atoms” (they can be either unary or binary).

**Input:** the global schema $T_g$, a conjunctive query $q$ on $T_g$  
**Output:** the set $Q_S$ of the queries that must be sent to involved sources

```
begin
    $Q_S := \{\}$
    if ($\text{checkConsistency}(q, T_g)$) then
        $(c_t, c_s) := \text{divide}(T_1, S, q)$
        $Q_s := \{\}$
        if ($c_t \neq \emptyset$) then
            $Q := \text{PerfectRef}(q(x) \leftarrow c_t, T_g \cup \{A\})$
            $Q_s := \text{select}(S', Q)$
            foreach ($q_s \in Q_s$) do
                $Q_s := Q_s \cup \{q(x) \leftarrow c_s\}$
            end
        else
            $Q_s := Q_s \cup \{q(x) \leftarrow c_s\}$
        end
        foreach ($q_s \in Q_s$) do
            $B := \text{buckets}(q_s, S')$
            if ($2B_q \subseteq B$ or $B_q = \emptyset$) then
                $R := \text{combine}(B)$
                foreach ($r_i \in R$) do
                    $q_s := \text{generate}(r_i)$
                    $Q_s := Q_s \cup q_s$
                end
            end
        end
    end
    return $Q_S$
end
```

Algorithm 1: Query resolution algorithm

### 3.3 Analysis

Algorithm 1 runs as follows: for a given conjunctive query $q$ expressed in terms of the global schema $T_g$, the first step consists in applying the Consistent algorithm [3] on $q$ considered as a canonical instance. The purpose is to verify the consistency of $q$ with respect to constraints expressed in $T_g$. In other words, this step makes it possible to avoid evaluating queries that could only lead to an empty result. For instance in our example given in Figure 5, since $B \subseteq \neg D$, it is not useful to evaluate the query $q(x) \leftarrow B(x), D(x)$. The Consistent algorithm has a polynomial time complex-
ity on the size of the TBox $\mathcal{T}_g$ and on the size of the ABox, represented by $q$ [3].

The second step starts by dividing the body of $q$ into the two conjunctions $c_2$ and $c_4$. We then apply the PerfectRef algorithm [3] only on taxonomic atoms of $q$, and using $\mathcal{T}_i \cup \mathcal{A}$ as TBox. The result of PerfectRef is a union of conjunctive queries (UCQ), among which we keep only those conjunctive queries that are entirely expressed on source atoms (function select). Then we complete the selected rewritings with the conjunction $c_1$ of the initial query $q$. Still considering our example in Figure 5, this step allows us to evaluate queries of Scenario (A) and, partly, those of Scenarios (C), (D) and (E). The PerfectRef algorithm compiles the TBox knowledge into reformulations of the given query. In our mediator, this knowledge consists of the subsumption relations contained in $\mathcal{T}_i \cup \mathcal{A}$. Thanks to this knowledge, the atoms involving taxonomic concepts are rewritten into source atoms, i.e., concepts that are at the leaves of the taxonomy $\mathcal{T}_i \cup \mathcal{A}$. These leaves are inevitably source atoms, from the specification of $\mathcal{T}_i$ given in Definition 1. The PerfectRef algorithm has a polynomial time complexity on the size of $\mathcal{T}_i \cup \mathcal{A}$ and it is exponential on the size of $q$ [3]

Let $Q_1'$ be the UCQ resulting from Step 2, the next step is to compute the queries to be sent to each source. This is performed by the foreach part in Algorithm 1, a MiniCon algorithm tailored to our semantic context. It addresses queries of Scenarios (C), (D) and (E). Step 2 outputs reformulations that involve only the source parts of the global schema. Then, we can use the LAV mappings (cf. Definition 2) in order to determine all the involved sources. For instance, the user can query a person’s name. There may be several concepts in the global schema corresponding to a Person and having an attribute name. This knowledge is contained in the global schema and we have (i) to compute what are the sources that have this concept with this attribute and (ii) to build the queries that must be sent to these sources. To achieve this step, we have devised an adaptation of the MiniCon algorithm [16], which is one of the well known query rewriting algorithms for LAV mappings, that has been proven to be efficient and scalable in practice. In MiniCon, the information about a source $S$ is given by a set of conjunctive views $v_c(x) \leftarrow \text{conj}(x,y)$, where $\text{conj}(x,y)$ is a conjunction of atoms expressed with $S$’s alphabet. We adapted it to our context, where the information about a source $S_i$ is given by the TBox $\mathcal{T}_i$ included in the global schema $\mathcal{T}_g$.

Our adaptation of MiniCon takes as input a query consisting exclusively of source atoms, that can exist in several sources. We adapted to our semantic context each of the three classical stages: (i) the building of buckets, (ii) the combination of these buckets, and (iii) the generating of valid subqueries to be sent to each involved source. To illustrate the process, we still consider our example given in Figure 5 and, for instance, the query:

$q(x, y, z) \leftarrow A(x), a_1(x, y), a_2(x, z).$

Before Step 3, $Q_1'$ contains the following set of queries:

$q_1(x, y, z) \leftarrow S_1.B(x), a_1(x, y), a_2(x, z),$
$q_1(x, y, z) \leftarrow S_2.B(x), a_1(x, y), a_2(x, z),$
$q_2(x, y, z) \leftarrow S_1.E(x), a_1(x, y), a_2(x, z),$
$q_2(x, y, z) \leftarrow S_2.E(x), a_1(x, y), a_2(x, z),$
$q_3(x, y, z) \leftarrow S_2.J(x), a_1(x, y), a_2(x, z).$

For each of these queries we have to compute the set of buckets. Formally, let $q(x) \leftarrow g_1(z_1), \ldots, g_n(z_n)$ be the input query, for each atom $g_i$ of $q$, the bucket $b_i$ contains the sources owning answers for $g_i$. With the same aim as for MiniCon’s optimization with respect to the bucket algorithm of [12], we introduce here some tests whose purpose is to put in the bucket of an atom $g$ only those sources whose instances of $g$ could actually be joined with instances of at least one of the other atoms of the same query. Consider for example the reformulated query:

$q_1(x, y, z) \leftarrow S_1.B(x), a_1(x, y), a_2(x, z)$

No bucket will be created because $B$ has neither attribute $a_1$, nor attribute $a_2$. In fact, among the 5 queries in $Q_s$, only $q_2$ and $q_2'$ have at least one source in each of their buckets.

In the next stages, the buckets are combined in order to get valid subqueries to be sent to the involved sources. Here again, we adapted the validity property to our context, where each source is represented by a TBox. We then verify all constraints contained in the involved $\mathcal{T}_i$ while building its corresponding subquery. In our example, the subqueries computed for $q_2$ and, respectively, $q_2'$ are:

$q_2.S_1(x, y) \leftarrow E(x), a_2(x, y)$ (sent to $S_1$),
$q_2.S_2(x, y, z) \leftarrow a_1(x, y), a_2(x, z)$ (sent to $S_2$),
$q_2'.S_1(x, y) \leftarrow a_1(x, y)$ (sent to $S_1$),
$q_2'.S_2(x, y, z) \leftarrow E(x), a_2(x, y), a_2(x, z)$ (sent to $S_2$).

Concerning complexity, creating the buckets for each query $q_g$ in $Q_s$ is in (worst case) $O(n \times t \times k \times t)$, where $n$ is the number of queries in $Q_s$, $l$ is the average number of atoms in queries $q_g$, $k$ is the number of sources and $t$ is the average size of the TBoxes $\mathcal{T}_i$. Combining the buckets is in $O(n' \times b')$, where $n'$ is the number of queries $q_g$ which have at least one source in each of their buckets ($n' \leq n$), $l$ is the average number of atoms in these queries and $b$ is the average size of the buckets. Our query resolution algorithm is based on scalable algorithms, but it is clear that optimizing Step 2 in order to get fewer rewritings with shorter length is the way to optimize the whole resolution process.

After a run of Algorithm 1, the computed subqueries are sent to the sources, by using the Sesame architecture presented in Section 2. Then, each of these queries is evaluated on its source’s relational database, through the OBDA system (i.e. Ontop) and the answer is sent back to the mediator. Each answer feeds the global result relation, whose attributes are the distinguished variables of the initial global query $q$, plus an extra attribute that represents the source that has given the answer. Notice that there are NULL values in tuples of this global answer set, for the variables not instantiated in the corresponding source. These principles and algorithms have been implemented for the application presented in Section 4.

4. APPLICATION TO PERSONAE

The Personae project, led by the History Institute CESR, has been an interesting application for our semantic mediation system because in this project, the partners wanted to share their data but each of them had excellent reasons for keeping their own data schema. Indeed in these fields

---

Notice that other reformulation algorithms exist, such as Presto [18], that optimize and greatly reduce the complexity in practical cases, by avoiding many reformulations that are contained in each others.

---

different sources. They will work with RDF and OWL formats. Moreover, several resources are currently built by our historian colleagues such as a thesaurus of first names (with equivalences in Latin and other European languages of the Renaissance), dictionaries of functions, clerical or military positions, etc. in order to enrich the information retrieval potential offered by the portal.

At the lower level, each local database has an OBDA system installed, i.e., the local administrator has built the TBox $\mathcal{T}_i$ and the OBDA mappings between $\mathcal{T}_i$ and the local relational database. In this way, any authorized application can query the source in SPARQL via its ontology. The source can publish this ontology in order to allow applications to access its data, alternatively it can give the ontology and grant access to some identified partners. Irregardless of this access to applications and/or the mediator, the source administrator can continue to publish her data as she wants, via a classical web site. Indeed, both Bude and Chantres databases have been funded to provide their data in a specific web site, via query forms that are used by several teams of researchers, not necessarily involved in Personae. Moreover a source may cover a wider field than that covered by the integration system, it is the case for the database Bude with respect to Personae. By applying the principles presented in this article, only the conceptual portion of Bude that is relevant for Personae is selected and stored in the agreement ontology built for joining Personae.

This is out of the scope of this paper, but we precise that we had to design and develop new extensions to our agreement-ontology building algorithm, because there exists no reference ontology for the prosopography field. It would be of interest to design such a reference ontology in collaboration with our historian colleagues (as an extension of the well established CIDOC CRM). To apply right now our proposals to the project, we used taxonomies extracted from YAGO, which has the advantage of offering vocabularies...
Section 3, some excerpts of middle parts of Personae’s global schema, or, in the words of the target with the data values returned by SQL evaluation. Thus, mapping axioms allow the system to create RDF triples, by replacing the placeholders in the SQL query. Accordingly, our OBDA mappings, otherwise they must be manually defined. According to Ontop, each mapping axiom is a pair (source, target). The source is a SQL query over the database and the target is a graph pattern in RDF Turtle\textsuperscript{12}. The target’s triples have some subjects or objects that contain place holders which reference column names mentioned in the SQL query. Thus, mapping axioms allow the system to create RDF triples, by replacing the place holders in the target with the data values returned by SQL evaluation.

We present in Figure 8 some excerpts of the top and middle parts of Personae’s global schema, or, in the words of Section 3, some excerpts of $T_s$, $A_1$ and $A_2$.

The global query is expressed either in SPARQL, or via a form. It is then translated in a conjunctive query, input of our query resolution algorithm described in section 3. We give in Table 1, first row, an example of a SPARQL query on the Personae’s global schema illustrated in Figure 8, that asks for the persons whose name starts by “ac” (“iu” is the flag value for function \texttt{regex}).

### Table 1: Example of a global query and the corresponding generated local queries.

The corresponding conjunctive query for the global query

\texttt{PREFIX r:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>}
\texttt{PREFIX : <http://www.li.univ.tours.fr/personae/>}
in Table 1 is: \( q(x, n) \leftarrow \text{Personne}(x), \text{nom}(x, n) \).

The conjunctive query is generated without the FILTER constraint, which is however added to the SPARQL sub-query that is sent to the selected sources after the query rewriting process. Algorithm 1 takes as input the query \( q \) and the global schema and outputs the following UCQs:

Chantres: \( Q(x, n) \leftarrow \text{Personne}(x), \text{nom}(x, n) \cup Q(x, n) \leftarrow \text{Personne}_{autrenom}(x), \text{nom}(x, n) \).

Bude: \( Q(x, n) \leftarrow \text{Personnages}(x), \text{nom}(x, n) \).

Each UCQ is then translated to a SPARQL query for each involved source. We use the CONSTRUCT form for storing each answer as a set of RDF graphs, to simplify the merge of all answers. The generated queries for sources Chantres and Bude are given in Table 1, rows 2 and 3.

Then the generated queries are sent to the involved source. More precisely, as described in Section 2, the mediator gets the answers from a source by invoking its interrogation repository service through the SPARQL endpoint declared by this source in the sources directory of the mediator. The query is locally evaluated by the source on its relational database through its OBDA mappings. Each answer from a source is a set of RDF graphs which is saved by the mediator in its temporary repository results. Each of these RDF graphs keeps the source it comes from, thanks to the \text{hasSource} property. Figure 9 shows an excerpt of the content of the temporary repository results after the interrogation process, for our example of global query, given in Table 1.

With this repository of results, we can present to the user the answers from both Chantres and Bude sources, we can also allow the user to see only those from a chosen source, or perform an ordering, etc., i.e., running more specific queries on the result repository, without querying the sources again.

5. CONCLUSION

We propose a mediator to integrate OBDA systems, which takes the form of a web portal that provides access to several resources. It may also be queried by applications, through a SPARQL endpoint. Each resource preserves its autonomous running, maintaining its proper web site or any other type of access, independently from the mediator. To join the mediation system, a resource must install an OBDA access (i) to inform about its conceptual model (i.e., a lightweight ontology that represents content it wants to share) and (ii) to output the data queried using this model, through a SPARQL endpoint. As explained in the introduction, our proposal keeps the advantages of OBDA systems while limiting the efforts to build the global schema and while relying on a truly distributed query-answering process.

The incrementally built global schema is composed of the parts of the source’s ontology that are relevant for the mediator. These parts are interlinked via a taxonomy. This taxonomy models the mediation domain, it may be manually built, but in our system it is incrementally built from a reference ontology and from each new source. Our system may be considered as a simplified case of those described for distributed description logics (DDL) [21, 8], because all links between sources are through the taxonomy and not peer-to-peer, with a focus on query processing, as in the framework for ontology integration described in [6]. In this article, we presented the complete query answering functionality, in particular the query-rewriting algorithm.

We have also described an application of our proposal, within the context of the Personae project. This project is to enable historians to share their data on Middle Ages and Renaissance prosopography. Our mediator architecture is a way for them to achieve this goal while keeping their data...
under control and without changing their internal formats.

This experiment has led us to suppose that our mediator architecture could be used to integrate resources of the semantic web, in particular linked open data. Provided that a conceptual definition of the resource is available, it could be used by the mediation system. This is our most interesting future study to formalize this new direction, because the current trends in linked data integration miss some means of assisting data consumers, who have the heavy charge to discover links when they do not exist [10]. Links do not exist in general, because data owners do not know how to link their data to other graphs. Clearly, linked data integration is not just a matter of graphs that can be easily extended, because edges in those graphs do not share a common semantics: some of them associate instances to their class, some of them are instances of concepts properties, some of them are defining concepts and properties, and so on. Our proposal, aiming to integrate the conceptual description parts, may be a way of alleviating the charge of data consumers without increasing the charge of data providers in the LOD context [20].

Another important future work raised by the Personae project is to study how entity-resolution may be supported by the mediator system. i.e. if some similarity measures could be inferred to assist historians to detect, for instance, people described in several different sources. Apart from Personae, we are working on experiments on run-time performance of the query evaluation implementation against different data sources, which may suggest some optimizations of the whole query resolution process.

6. REFERENCES


