

# An approach for Query Decomposition on Federated SPARQL Query Systems

Danusa R B Cunha<sup>1</sup>, Bernadette Farias Lóscio<sup>1</sup>

Center of Informatics - Federal University of Pernambuco - Brazil  
drbc, bfl@cin.ufpe.br

**Abstract.** Providing integrated access to data distributed over Linked Data Federations has become a major research challenge, mainly due to heterogeneity problems. In such context, this work proposes a solution for query decomposition over Linked Data Federations, i.e., sets of RDF data sources published according to the principles of Linked Data. Our main contribution lies in the definition and implementation of a query decomposition process, considering that the data sources have structurally distinct ontologies, which describe their schemas. In order to evaluate the proposed approach, a prototype was implemented and some experiments were performed.

Categories and Subject Descriptors: H.2 [Database Management]: Miscellaneous; H.2.5 [Heterogeneous Databases]: Data translation; H.3.3 [Information Search and Retrieval]: Query formulation; H.3.4 [Systems and Software]: Distributed systems

Keywords: Data Integration, Linked Data, RDF, Web of Data

## 1. INTRODUCTION

Over the years, different solutions for data integration have been proposed, as conventional Data Integration Systems (DIS) [Doan et al. 2012], Peer Data Management Systems (PDMS) [Halevy et al. 2006; Sung et al. 2005] and Dataspaces Systems [Franklin et al. 2005]. The primary goal of a data integration system consists of offering a uniform interface that provides integrated access to a collection of distributed data sources, which are usually heterogeneous, autonomous and dynamic. In general, a data integration system should enable users to specify what they want, rather than thinking about how they obtain the answers.

Initially, data integration solutions were mainly focused on solving the problem of integrating data distributed in structured data sources (e.g in relational databases). Later, there was a huge interest on using XML as a common data model to integrate data available on the Web [Doan et al. 2012]. Recently, the growing adoption of Semantic Web technologies, such as RDF (Resource Description Framework) and OWL (Ontology Web Language), combined with the Linked Data Principles [Bizer et al. 2009] motivated the development of new data integration solutions.

The principles of data integration in the Semantic Web context are very similar to those of conventional data integration, despite the different terminologies employed to denote them. In both contexts there are two main approaches for data integration: virtual and materialized approaches [Wiederhold 1992; Langegger 2010; Lenzerini 2002; Lóscio 2003; Salgado et al. 2011]. In conventional data integration systems the mediator architecture implements the virtual approach, in which the data remains in the sources and queries submitted to the data integration system are decomposed into queries addressed directly to the sources. In the Semantic Web scenario, the virtual approach is implemented by a federated query architecture [Haase et al. 2010; Hartig and Langegger 2010], where multiple RDF datasets are accessed by a federation layer through a SPARQL endpoint. Loading of the data is, thus, not necessary, and an *ad hoc* federation can be built by simply incorporating an additional SPARQL endpoint. The materialized approach, on the other hand, is based on a central repository, where all data is physically loaded from dumps of the data sources, and queries are evaluated against

this single repository [Haase et al. 2010; Hartig and Langeegger 2010].

In this article, we are interested on virtual data integration in the Semantic Web context. We focus on Linked Data Federations that offer an integrated view of data distributed in multiple RDF data sources. In this context, the federator is the component that plays the role of the mediator in conventional DIS. It receives a SPARQL query, decomposes it into subqueries over the RDF data sources and integrates the corresponding results [Bizer et al. 2009]. Ontologies are used to describe the federator schema and the RDF data sources.

We propose a query decomposition strategy that addresses important challenges faced by virtual data integration solutions. These challenges include the use of different representations to describe the same real world object and partial data retrieval, which happens when data sources have different query answering capabilities and provide partial answers for a given query. In order to deal with the structural heterogeneity, we use a formalism to describe ontology mappings that takes into account the most common ontology mismatches. Based on these mappings is also possible to generate subqueries to retrieve complementary information distributed on different datasets of the federation. Another important aspect of our approach is that it allows the decomposition of SPARQL queries that use FILTER and OPTIONAL operators. As presented in the literature, these operators are not considered in similar approaches for decomposition of SPARQL queries [Correndo et al. 2010; Leme et al. 2009; Makris et al. 2012].

The remainder of the paper is organized as it follows. Section 2 presents an example to illustrate the data integration problem in the context of Linked Data Federations. Section 3 provides basic definitions, which will be used along this paper. Section 4 describes the proposed approach. Section 5 presents some aspects concerning the validation of our proposal. Section 6 discusses some related works and Section 7 presents some conclusions.

## 2. MOTIVATING EXAMPLE

In this section, we present an example to illustrate some of the main challenges of federated query processing on Linked Data. Consider a federation  $F$  composed by three different RDF data sources: DBLP<sup>1</sup>, Kisti<sup>2</sup> and DBpedia<sup>3</sup>. DBLP and Kisti data sources provide information about Computer Science publications, while DBpedia is a cross-domain dataset. Each data source is associated with an ontology, called local ontology, that describes the concepts and relationships used to represent the data. An integrated view of the data is offered by the federator through the SWRC (Semantic Web for Research Communities) ontology (see Figure 1 (a)), which is a domain ontology that describes entities of research communities, such as persons, organizations, publications (bibliographic metadata) and their relationships. The local ontologies are depicted in Figure 1 (b) (note that AKT ontology describes the DBLP dataset).

In what follows, consider the following global query  $Q_F$  submitted to the federator: "*Return the title, the year (if available) and the abstract of all publications about Data Integration whose author is Alon Y. Halevy with a brief biography of the author*". To answer this query,  $Q_F$  needs to be decomposed into a set of queries to be executed on the data sources that compose the federation  $F$ .

However, decomposing  $Q_F$  into a set of subqueries is not an easy task. The first challenge is the use of different concepts or properties to describe the same real world object. For example, the concept  $s:Person$  from SWRC ontology corresponds to the concept  $k:Person$  from Kisti ontology, and the property  $s:name$  of  $s:Person$  corresponds to the property  $k:engNameOfPerson$ . Another example is the property  $s:publisher$  that relates an article to its authors. In the Kisti ontology, the

<sup>1</sup>dblp.rkbexplorer.com/sparql/

<sup>2</sup>kisti.rkbexplorer.com/sparql/

<sup>3</sup>wiki.dbpedia.org

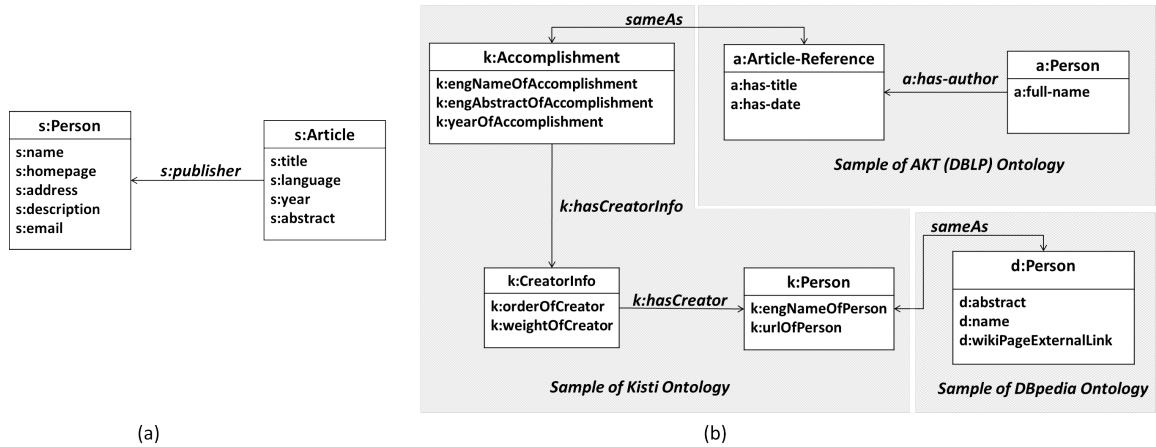


Fig. 1: (a) Sample of SWRC Ontology and (b) Sample of Local Ontologies

association between article and author is not direct: an instance of *k:Accomplishment* has to be associated with an instance of *k:CreatorInfo*, and then it should have an association between this instance of *k:CreatorInfo* and the instance of *k:Person* that represents the author of the article. In this case, there is a correspondence between a property and a set of properties.

Another challenge is to deal with data sources that may provide partial answers for a given query. Kisti and DBLP data sources, for example, can return publications about Data Integration that belong to Alon Y. Halevy, but they are not capable of returning all of the required properties. In this case, subqueries must be evaluated on these data sources based on the information that each source is able to provide. In a similar way, DBpedia doesn't provide information about papers, but it may offer complementary information about authors obtained from Kisti and DBLP. For instance, the biography of the authors obtained from Kisti may be retrieved from DBpedia (*d:abstract*) using the property *sameAs* between *k:Person* and *d:Person*. This type of partial answering should be considering during query decomposition.

In order to solve the aforementioned problems, we propose a query decomposition strategy based on heterogeneous mappings between the federator ontology and the local ontologies. These mappings capture the correspondence between heterogeneous concepts such as a property and a set of properties, and make it possible to generate subqueries that retrieve partial answers for a given global query. In the next section, we introduce the formalism to describe these mappings.

### 3. ONTOLOGY MAPPINGS

In order to perform the query decomposition process, it becomes necessary to obtain the mappings between the local ontologies that describe the data sources that are being integrated and the domain ontology that describes the integrated view offered by the federator. The mapping generation process can be divided in two main phases: (i) *vocabulary matching*: during this phase is performed the alignment between the domain ontology and the local ontologies in order to identify correspondences between their concepts, and (ii) *generation of mapping rules*, which induces the mapping rules from the step (i) in order to describe how to map concepts from the domain ontology into concepts from local ontologies.

In our approach, we extend the *vocabulary matching* previously defined in [Leme et al. 2009; Sacramento et al. 2010] to capture correspondences between ontologies with structural heterogeneities. Our approach deals with situations where a concept is represented as a property in the domain ontology and as a path of properties in a local ontology. To do this, we use the following path definition [Cunha

and Lóscio 2014]:

**Definition 1.** *Property Path.* Given a set of properties  $p_1, \dots, p_n$  of an ontology  $O$ ,  $\mu = p_1, \dots, p_n$  is a path if: (i) each property  $p_i$ ,  $1 \leq i < n$ , is a property of  $O$  and (ii) the range of  $p_i$  is equal to the domain of  $p_{i+1}$ ,  $\forall p_i$  such that  $1 \leq i < n$ .

In the following, the extended version of the formalism adopted to capture correspondences between a local ontology ( $O_L$ ) and the domain ontology ( $O_D$ ) is described.

Let  $O_L$  and  $O_D$  be two ontologies, and  $V_L$  and  $V_D$  be their respective vocabularies. Let  $C_L$  and  $C_D$  be the sets of classes, and  $P_L$  and  $P_D$  be the sets of datatype or object properties in  $V_L$  and  $V_D$ , respectively. A vocabulary matching of a local ontology  $O_L$  to a domain ontology  $O_D$  is a finite set  $S$  of quadruples  $(v_1, e_1, v_2, e_2)$  such that:

- If  $(v_1, v_2) \in C_L \times C_D$ , then  $e_1$  and  $e_2$  are top class  $\top$ , i.e, there are no restrictions on the respective classes. In this case,  $(v_1, e_1, v_2, e_2)$  represents a class correspondence;
- If  $v_1 \in P_D$  and  $v_2$  is a path  $\mu = p_1, \dots, p_n$ , where each property  $p_i \in P_L$ ,  $1 \leq i < n$ , then  $e_1$  and  $e_2$  are classes in  $C_L$  and  $C_D$ , that must be subclasses of the domains, or the domains themselves or the restrict domains of properties  $v_1$  and  $p_n$ , respectively. If  $v_1$  is a datatype property or an object property then  $p_n$  is a datatype property or an object property, respectively, and the domain of  $v_1$  has a correspondence with the domain of  $p_1$ . A special case of a property path happens when  $n = 1$ . This type of correspondence is called property correspondence.

To illustrate how this formalism is applied, Tables I and II show some samples of a vocabulary matching between two ontologies.

Table I: Sample of the result of a vocabulary matching between SWRC ontology and the AKT ontology.

#	SWRC		AKT(DBLP)	
	$v_1$	$e_1$	$v_2$	$e_2$
1	s:title	s:Article	a:has-title	a:Article-Reference
2	s:date	s:Article	a:has-date	a:Article-Reference
3	s:Article	$\top$	a:Article-Reference	$\top$

Table II: Sample of the result of a vocabulary matching between SWRC ontology and the Kisti ontology.

#	SWRC		Kisti	
	$v_1$	$e_1$	$v_2$	$e_2$
1	s:title	s:Article	k:engNameOfAccomplishment	k:Accomplishment
2	s:date	s:Article	k:yearOfAccomplishment	k:Accomplishment
3	s:description	s:Article	k:engAbstractOfAccomplishment	k:Accomplishment
4	s:publisher	s:Article	[k:hasCreator.k:hasCreatorInfo]	k:Accomplishment
5	s:Article	$\top$	k:Accomplishment	$\top$

For example, in Table I, line 3 indicates that classes  $s:Article$  and  $a:Article-Reference$  match (a class correspondence) and line 1 indicates that properties  $s:title$  and  $a:has-title$  match (a property correspondence), as their classes  $s:Article$  and  $a:Article-Reference$  are equivalent. That is, the property  $s:title$  of an instance of  $s:Article$  is equivalent to property  $a:has-title$  of a corresponding instance of  $a:Article-Reference$ . It is important to note that, in both examples,  $s:Article$  and  $a:ArticleReference$  are the domains of  $s:title$  and  $a:has-title$ , respectively, representing the context of the properties. Furthermore, we consider not only homogeneous correspondences (between classes or between properties), but also correspondences expressing structural heterogeneity (Path Property Correspondence). For example, line 4 in Table II indicates an example of a matching between a property and a property

path. In this case, in the ontology  $O_D$ , the object property  $s:publisher$  corresponds to the path  $v_2 = k:hasCreatorInfo.k:hasCreator$ .

It is important to note that, in our work, the generation of such correspondences can be a semi-automatic process, requiring the user intervention for creation, validation and refinement of the ontology alignment. After obtaining the correspondences between the local ontologies and the domain ontology, the set of mappings can be automatically generated.

Generally, mappings between the domain ontology and each local ontology are specified through rules of the form  $\psi(w) \leftarrow \phi_1(t_1) \wedge \dots \wedge \phi_n(t_n)$ , where  $\phi_1(t_1) \wedge \dots \wedge \phi_n(t_n)$ , called the body of the mapping rule, is an atom or an atom conjunction, and each  $\phi_i(t_i)$  is a class or a property that occurs in the local ontology  $O_L$ .  $\psi(w)$ , called the head of the mapping, is an atom, where  $\psi(w)$  could be a class or a property that occurs on the domain ontology  $O_D$  and  $w$  are terms sequences. To understand the semantics of these mappings consider, for example, the following mapping rules regarding classes and properties:

- (1)  $s:Person(p) \leftarrow a:Person(p)$
- (2)  $s:Person(p) \leftarrow k:Person(p)$
- (3)  $s:Person(p) \leftarrow d:Person(p)$
- (4)  $s:name(p,n) \leftarrow a:full-name(p,n), a:Person(p)$
- (5)  $s:name(p,n) \leftarrow k:engNameOfPerson(p,n), k:Person(p)$
- (6)  $s:name(p,n) \leftarrow d:name(p,n), d:Person(p)$

Rule (1) expresses the fact that if there is an instance  $p$  from *Person* in the dataset DBLP then there will be an instance  $p$  from *Person* in the integrated view offered by the federator. The same applies for rules (2) and (3) for datasets Kisti and DBPedia, respectively. Rule (4) expresses the fact that if  $n$  is the value of the property *full-name* of an instance  $p$  from *Person* in the DBLP dataset, then  $n$  will be the value of the property name of the instance  $p$  from *Person* in the integrated view offered by federator. The same applies for rules (5) and (6) for datasets Kisti and DBPedia, respectively.

#### 4. QUERY DECOMPOSITION APPROACH

The query decomposition approach proposed in this paper aims to solve the following problem: Given a domain ontology  $O_D$  that represents the federator schema; a set of RDF datasets  $D = D_1, \dots, D_n$ ; a set of local ontologies  $O_{L_D} = O_{D_1}, \dots, O_{D_n}$ , where each  $O_{L_D}$  describes the vocabulary of the dataset  $D_i$  and a set of mappings  $M$  between  $O_D$  and each local ontology  $O_{L_D}$ , the query decomposition problem consists of identifying how to decompose a global query  $Q_F$ , defined in terms of  $O_D$ , in one or more queries  $\{Q_{D_1}, \dots, Q_{D_n}\}$  to be submitted to datasets from  $D$ , whereas the local ontologies that describe the schemas of datasets from  $D$  may have distinct structures.

In the remainder of this section, consider the following query  $Q_F$  posed to the federator: "*Return the title, the year (if available) and the abstract of all publications about Data Integration whose author is Alon Y. Halevy with a brief biography of the author*". The corresponding SPARQL query is presented in Figure 2. As we may observe, the main components of a SPARQL query are: (i) the *Basic Graph Pattern* (BGP), which is composed by a sequence of *triple patterns* (TP). A triple pattern is an expression of the form  $(s, p, o) \in (I \cup B) \times I \times (I \cup B \cup L)$  where  $I$ ,  $B$ , and  $L$  are IRIs, Blank Nodes, and literals respectively; (ii) the solution modifiers, which allow to modify the output of the pattern and (iii) the output, which specifies the result form of the query. It is important to note that our approach deals with SELECT SPARQL queries whose final form of the result is a table. Moreover, the subject ( $s$ ) and the object ( $o$ ) of a triple pattern must be a variable or a literal. The operators AND, OPT and FILTER are also allowed.

The proposed approach consists of the following tasks:

```

PREFIX s:<http://swrc.ontoware.org/ontology#>

SELECT ?title ?year ?abstract
WHERE{
    ?paper rdf:type s:Article .
    ?paper s:title ?title .
    ?paper s:abstract ?abstract .
    FILTER regex{?title, "Data Integration", "i"}
    OPTIONAL{?paper s:year ?year .}
}

```

Fig. 2: SPARQL query example.

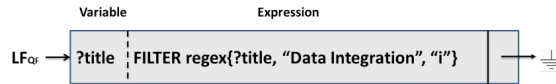
**1) Query Graph Generation:** during this task, a query graph  $G_Q = (V, E)$  is generated from  $Q_F$ , where  $V$  is the set of vertices composed by subjects ( $s$ ) and objects ( $o$ ) of each  $TP \in BGP$  of  $Q_F$ , and  $E$  is the set of edges representing the predicates ( $p$ ) of each  $TP \in BGP$  of  $Q_F$ .

Given a SPARQL query  $Q$ , the first step of this task is to find the FILTER and the OPTIONAL operators that compose the  $BGP$  of  $Q$ . A list  $L_{F_Q}$  is created to store each FILTER operator from  $Q$ .

In this work, an operator FILTER is represented by two attributes: an expression and a set of variables. To illustrate how we deal with the FILTER operator, consider the filter expression of  $Q_F$  (Figure 2): FILTER *regex{?title, "Data Integration", "i"}* where:

**Expression:** *regex{?title, "Data Integration", "i"}*  
**Variable:** ?title

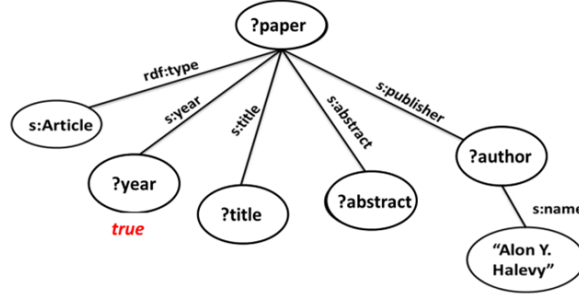
Both expression and variable are stored as a node on the list  $L_{F_{Q_F}}$  as illustrated on Figure 3.

Fig. 3: List of Filters  $L_{F_{Q_F}}$ 

If a triple pattern is optional (*OPTIONAL TP*), then  $TP$  is modified, following this format:  $TP(s, p, < o, op >)$ , where  $op$  is a Boolean variable with true value.

To illustrate the query graph generation task, consider the triple pattern  $(?paper, rdf:type, s:Article)$  from  $Q_F$ . Two new nodes and a new edge are created to represent this triple pattern. New nodes receive values  $?paper$  and  $s:Article$ , while the new edge receives the value  $rdf:type$ . Suppose the next triple pattern  $(?paper s:year ?year)$ . Note that only a new node and a new edge are created with values  $?year$  and  $s:year$  respectively, because there is already a node with the value  $?paper$ . As this triple pattern is optional, then, in the query graph, the node with value  $?year$  is annotated with true. This procedure is done for each triple pattern  $TP$  from  $Q_F$ . At the end of the **Query Graph Generation** task, we obtain the query graph depicted in Figure 4.

**2) Graph Decomposition (subqueries):** This task receives as input a query graph  $G_Q = \langle V, E \rangle$ , a set of datasets  $D$  and a set of mappings  $M$ . During the decomposition process,  $G_Q$  is decomposed into a set of graphs  $G_{D_1}, \dots, G_{D_n}$  according to the mapping rules of  $M$ . In general, for each dataset  $D_i$ ,  $1 \leq i \leq n$ , of  $D$ , a new graph is created based on the mappings between the concepts represented by the vertices of  $G_Q$  and the concepts available on the dataset  $D_i$ . The graph decomposition task is performed for each dataset  $D_i$  and at the end of the task the graph  $G_{D_i}$  is created. In order to produce the graph  $G_{D_i}$  the following steps are executed:

Fig. 4: Query Graph for query  $Q_F$ 

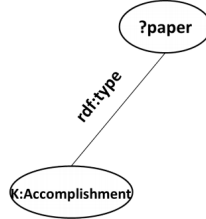
- Node Rewriting*: this step performs the rewriting of a node  $v$ , returning a new rewritten node  $v'$ . Two cases must be considered:
  - (1) if  $v$  is a variable or a literal, then the new node  $v'$  is assigned with the same value of the variable or the literal;
  - (2) if  $v$  is a class  $c$  and there is a mapping rule in  $M$  such that  $c(w) \leftarrow c_i(w)$  then a new node  $n'$  is created with value  $c_i$ .
- Edge Rewriting*: this step performs the rewriting of an edge  $uv$ , returning a new rewritten edge  $e'$ . Three cases must be considered:
  - (1) if  $uv$  is a variable, then the new edge  $e'$  is assigned with the same value of the variable;
  - (2) if  $uv$  is a property  $p$  and there is a mapping rule in  $M$  such that  $p(x,y) \leftarrow p_i(x,y)$  then a new edge is created with value  $p_i$  to connect the rewritten nodes of  $u$  and  $v$ .
  - (3) if  $uv$  is a property  $p$  and there is a mapping rule in  $M$  such that  $p(x,y) \leftarrow p_1(x,z) \leftarrow p_i(w,y)$ , then new  $i-1$  auxiliary nodes are created with values  $?t_j$ ,  $1 \leq j \leq i-1$ , and new edges are created representing each one of the properties  $p_i$  to connect the rewritten nodes of  $u$  and  $v$  through the auxiliary nodes  $?t_j$ .

To illustrate the graph decomposition process, consider query  $Q_F$  (Figure 2), the query graph  $G_{Q_F}$  (Figure 4) and the dataset Kisti. In the set of mapping, consider, for example, the following mapping rules:

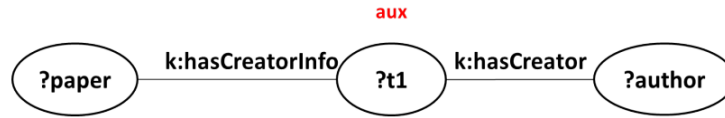
- (1)  $s:Article(Ar) \leftarrow a:Article-Reference(Ar)$
- (2)  $s:Article(Ar) \leftarrow k:Accomplishment(Ar)$
- (3)  $s:publisher(pub, Ar) \leftarrow a:has-author(pub, Ar), a:Article-Reference(Ar)$
- (4)  $s:publisher(pub, Ar) \leftarrow [k:hasCreatorInfo(pub,z).k:hasCreator(z, p)], k:Accomplishment(Ar)$

Consider the edge  $uv$  of  $G_{Q_F}$  that connects nodes  $?paper$  and  $s:Article$ , whose value is  $rdf:type$ . The first step of the graph decomposition is to rewrite  $u$ . Considering that  $u$  is a variable ( $?paper$ ), then a new node will be created in the query graph  $Q_{Kisti}$  and the value  $?paper$  will be assigned to the new node. The next step is to rewrite the node  $v$ . In this case, the value of  $v$  is the name of a class ( $s:Article$ ). Then, according to mapping rule (2), a new node will be created in the query graph  $Q_{Kisti}$  with value  $k:Accomplishment$ . Finally, the edge  $uv$  is rewritten. In this case, as the value of  $uv$  is a predicate, then a new edge will be created in the query graph  $Q_{Kisti}$  with the same value of the edge  $uv$ , i.e.,  $rdf:type$ . Figure 5 shows the query graph  $Q_{Kisti}$  obtained after the rewriting of the edge that connects nodes  $?paper$  and  $s:Article$ .

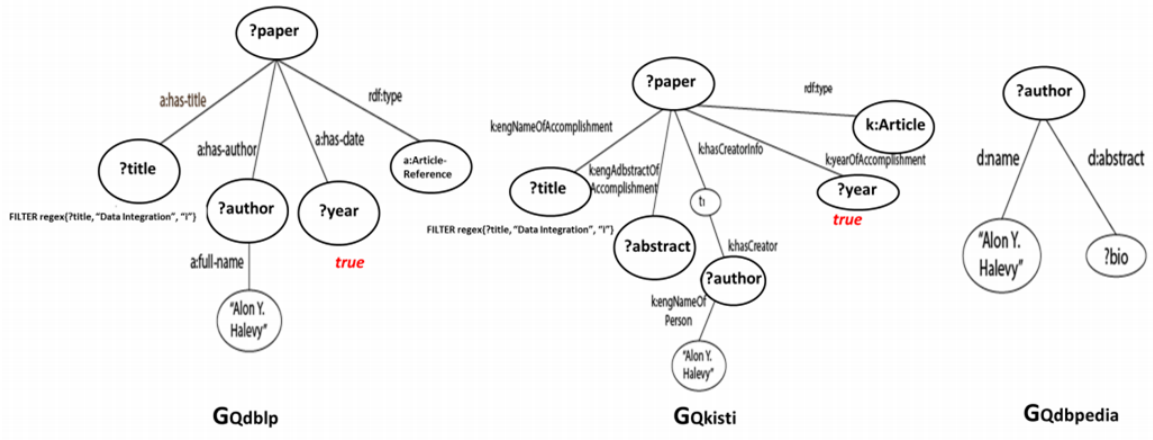
To illustrate the graph decomposition process in the presence of a structural heterogeneity between the domain ontology and one of the local ontologies, consider the edge  $uv$  from  $G_Q$  that connects nodes

Fig. 5: Partial Query Graph  $G_{Q_{kisti}}$ .

$?paper$  and  $?author$  whose value is  $s:publisher$ . Considering that nodes  $?paper$  and  $?author$  were already rewritten, i.e., corresponding nodes were created for the query graph  $G_{Q_{kisti}}$ , then the next step is to rewrite the edge  $uv$  ( $s:publisher$ ). According to the set of mappings  $M$  (Rule 4), there is a mapping between  $s:Publisher$  and the following property path:  $([k:hasCreator(pub,Ar).k:hasCreatorInfo(z)], k:Accomplishment(Ar))$ . In this case, each property that composes this path ( $k:hasCreator$  and  $k:hasCreatorInfo$ ) must be considered during the rewriting process. In order to connect nodes  $?paper$  and  $?author$  through properties  $k:hasCreator$  and  $k:hasCreatorInfo$  it becomes necessary to create an auxiliary node in  $G_{Q_{kisti}}$ . This auxiliary node is assigned with value  $?t_1$  as depicted in Figure 6. Then the edge that connects  $?paper$  and  $?t_1$  is assigned with value  $k:hasCreatorInfo$  and the edge that connects  $?t_1$  and  $?author$  is assigned with value  $k:hasCreator$ .

Fig. 6: Partial Query Graph  $G_{Q_{kisti}}$  after the rewriting of the edge  $s:publisher$ .

At the end of the graph decomposition task, a new query graph is created for each one of the local datasets that is able to answer the original query  $Q$ . Considering the query  $Q_F$  from our example, three new query graphs will be created for each one of the datasets from  $D$  (DBLP, DBPedia and Kisti) as depicted in Figure 7.

Fig. 7: Query Graphs for each dataset  $D_i$

**3) Subqueries generation:** the last step of the query decomposition process consists of creating SPARQL queries to be evaluated in the local datasets. For each graph  $G_{Q_i}$ , a new SPARQL query  $Q_i$  is generated considering node annotations for OPTIONAL operators as well as the corresponding filter list.

In our example,  $Q_F$  was transformed into the graph  $G_{Q_F}$  and this graph was decomposed into three graphs  $\{G_{Q_1}, G_{Q_2}, G_{Q_3}\}$ . For each graph, a SPARQL query is generated in a reverse process of the graph generation step. Figure 8 shows the queries generated for each query graph obtained at the end of this task.

<b>Q<sub>1</sub>: Query for dataset DBLP</b>	<b>Q<sub>2</sub>: Query for dataset Kisti</b>	<b>Q<sub>3</sub>: Query for dataset DBpedia</b>
<pre>SELECT ?title ?year WHERE {   ?paper rdf:type a:Article-Reference ;   a:has-title ?title ;   a:has-author ?author .   ?author a:full-name "Alon Y. Halevy".   OPTIONAL { ?paper a:has-date ?year . }   FILTER regex { ?title, "Data Integration", "i" } }</pre>	<pre>SELECT ?title ?year?abstract WHERE {   ?paper rdf:type a:Article ;   k:engNameOfAccomplishment ?title;   k:engAbstractOfAccomplishment ?abstract;   k:hasCreatorInfo ?t1 .   ?t1 k:hasCreator ?author .   ?author k:engNameOfPerson "Alon Y. Halevy".   OPTIONAL { ?paper k:yearOfAccomplishment ?year . }   FILTER regex { ?title, "Data Integration", "i" } }</pre>	<pre>SELECT ?bio WHERE {   ?author d:name "Alon Y. Halevy";   d:abstract ?bio . }</pre>

Fig. 8: Subqueries for each dataset  $D_i$

It is important to note that we focus just on the generation of the subqueries to be executed on the local data sources. The integration of the results obtained from each data sources is out of the scope of this article. This is a very complex task that requires the resolution of relevant problems as entity matching [Euzenat and Shvaiko 2007] and data fusion [Bleiholder and Naumann 2009; Dong et al. 2013]. In the context of federated SPARQL query processing, the data integration required efficient query processing strategies. Thus, join processing strategies as well as other sophisticated optimization approaches are needed to find an appropriate solution. Some approaches deal with these challenges as [Quilitz and Leser 2008; Magalhães et al. 2013; Schwarte et al. 2011]

It is also important to clarify that the focus of this work is not on aspects related with implementation and optimization of SPARQL queries.

## 5. IMPLEMENTATION AND EVALUATION

To evaluate our approach, a prototype, named oLinDa, was developed and some experiments were performed. oLinDa was implemented in Java and Jena API<sup>4</sup> was used to manipulate both ontologies and SPARQL queries. The evaluation was performed on a Dell Vostro 3450 machine with a processor Intel Core i5-2430M and 4GB of memory, and Linux Ubuntu 12.04 as operational system. Jena TDB<sup>5</sup>, which is an RDF repository, was used to store the datasets. Queries were manipulated through ARQ API<sup>6</sup>, whose implementation is available on GitHub<sup>7</sup> under Mozilla Public License.

A set of twenty (20) queries were considered in the experiments, among which sixteen (16) were extracted from the SP2Bench<sup>8</sup> SPARQL benchmark. These queries were chosen considering their main SPARQL constructs and their complexity. To perform the experiments, a repository, called

<sup>4</sup><http://jena.apache.org/>

<sup>5</sup><http://jena.apache.org/documentation/tdb/>

<sup>6</sup><https://jena.apache.org/documentation/query/appapi.html>

<sup>7</sup><https://github.com/danuserbc/olindarewriting>

<sup>8</sup><http://dbis.informatik.unifreiburg.de/forschung/projekte/SP2B/>

RD, was created to store integrated data from datasets DBLP, Kisti, DBpedia and Opus Swego<sup>9</sup>. From the *RD* repository, it was possible to assess whether the decomposition process proposed in this paper was working properly. The idea was to evaluate if the results obtained when executing a query *Q* directly on *RD* was similar to the results obtained when executing the subqueries obtained from the decomposition of *Q* in each one of the corresponding data sources. It is important to note that integration of the subqueries results were performed manually. At the end of the experiments, thirteen (13) queries were properly decomposed and the results obtained from the execution of queries directly on RD and the execution on the local data sources were the same in all the cases. Figure 8 shows an example of a query decomposed properly, treating both structural heterogeneity aspect as showing that some data sources may provide partial answers for the original SPARQL query. Two (2) queries could not be decomposed because they are queries that request properties that are not in the scope of the domain ontology and five (5) queries were incorrectly decomposed because they have UNION and OPTIONAL builders with more than one triple pattern.

Compute all authors that have published with Daniela Florescu and return the biography for each author, if possible.		
<pre> SELECT DISTINCT ?name ?bio WHERE{   ?daniela rdf:type s:Person .   ?daniela s:name "Daniela Florescu" .   ?document swrc:publisher ?daniela .   ?document swrc:publisher ?author .   ?author s:name ?name .   OPTIONAL {?author swrc:description ?bio .}   FILTER (?author!=?daniela) } </pre>		
DBLP	Kisti	DBpedia
<pre> SELECT DISTINCT ?name ?bio WHERE{   ?daniela rdf:type a:Person .   ?daniela a:full-name "Daniela Florescu" .   ?document a:has-author ?daniela .   ?document a:has-author ?author .   ?author a:full-name ?name .   FILTER (?author!=?daniela) } </pre>	<pre> SELECT DISTINCT ?name ?bio WHERE{   ?daniela rdf:type k:Person .   ?daniela k:engNameOfPerson "Daniela Florescu" .   ?document k:hasCreatorInfo ?t1 .   ?t1 k:hasCreator ?daniela .   ?document k:hasCreatorInfo ?t2 .   ?t2 k:hasCreator ?author .   ?author k:engNameOfPerson ?name .   FILTER (?author!=?daniela) } </pre>	<pre> SELECT DISTINCT ?name ?bio WHERE{   ?daniela rdf:type d:Person .   ?daniela d:name "Daniela Florescu" .   ?author d:name ?name .   OPTIONAL {?author d:abstract ?bio .}   FILTER (?author!=?daniela) } </pre>

Fig. 9: Query decomposed by OLinDa.

Importantly, the experiments demonstrated the feasibility of our approach, but it is still necessary to conduct a more formal proof, demonstrating that our approach is correct.

## 6. RELATED WORK

In this section, we briefly present some of the research literature related to our proposal. The works from [Correndo et al. 2010; Leme et al. 2009; Makris et al. 2012] are mainly focused on the problem of query rewriting, and not on query decomposition, as we do. These approaches do not deal with filter expressions on SPARQL queries, and consider only homogenous mappings between the ontologies that describe the datasets. In the former work, a query is first parsed, its basic graph pattern (BGP) is extracted and every triple pattern of the BGP is translated. This translation is done according to the correspondences that have been established to specify the semantic relationships between concepts of the source and the target datasets.

However, if a single triple pattern does not have a corresponding translation, i.e., there are no correspondences that allow the translation of this triple pattern, the entire query is discarded.

One of the main distinguishing features of our approach is that the *BGP* of the generated subqueries do not correspond exactly to the same *BGP* of the original query, i.e., just triple patterns that can

<sup>9</sup><http://opus.bath.ac.uk/>

be answered by the corresponding local data source needs to be translated. Considering our example, the data source DBpedia has just information about authors, therefore, in this case, just the triple patterns concerning author were considered during the generation of the corresponding subquery. In the mentioned query rewriting approaches, the subquery  $Q_2$  (Figure 8) wouldn't be rewritten and it would not be possible to obtain data from the DBpedia. Traditional query rewriting approaches do not allow performing incomplete translations of a *BGP*, i.e., they require the translation of the whole set of triple patterns. Our main purpose is to find not only the data source(s) that may answer the original query, but also collecting complementary information and further applying a data fusion process for answering the query. Other important aspect of our approach is that we deal with triple patterns that may have FILTER and OPTIONAL clauses, as depicted in Figure 8. The data source Kisti has information about title and year, which are treated with OPTIONAL and FILTER clauses respectively; then the generated graph will contain such triple patterns. Finally, we adopt a formalism that uses the concept of property path to deal with ontologies with distinct structures.

## 7. CONCLUSIONS AND FUTURE WORK

In this paper we presented an approach to deal with the query decomposition problem in the context of data integration on the Web of Data, where local data sources are RDF datasets described by an ontology and accessed through SPARQL queries. One of the main distinguishing features of our solution is that it can deal with ontologies with distinct structures through the manipulation of homogeneous and heterogeneous mappings [Sacramento et al. 2010; Leme et al. 2009] between their concepts. We deal with triple patterns that may have FILTER and OPTIONAL and our approach retrieves complementary information available on different datasets. In order to evaluate the proposed approach, a prototype was implemented and some experiments were performed.

As a future work, we plan to develop new experiments considering larger ontologies, more queries and known datasets. We also plan to extend the scope of the SPARQL language supported by the federator with the inclusion of other types of SPARQL queries (ASK, CONSTRUCT and DESCRIBE). Finally, we want to improve the query decomposition strategy to consider the user feedback as well as to include a new task to allow the automatic generation of the query execution plan. Besides, we intend to do a formal evaluation of our approach and develop a data fusion strategy.

## REFERENCES

- BIZER, C., HEATH, T., AND BERNERS-LEE, T. Linked data – the story so far, 2009.
- BLEIHOLDER, J. AND NAUMANN, F. Data fusion. *ACM Comput. Surv.* 41 (1): 11–141, 2009.
- CORRENDO, G., SALVADORES, M., MILLARD, I., GLASER, H., AND SHADBOLT, N. Sparql query rewriting for implementing data integration over linked data. In *Proceedings of the 2010 EDBT/ICDT Workshops*. EDBT '10. New York, NY, USA, pp. 41–411, 2010.
- CUNHA, D. R. B. AND LÓSCIO, B. F. olinda: Uma abordagem para decomposição de consultas em federações de dados interligados. In *29º Simpósio Brasileiro de Banco de Dados*, M. Moro, R. Galante, C. S. Hara, V. Martins, J. A. Macedo, F. R. C. Souza, J. E. Ferreira, F. Porto, and V. Braganholo. (Eds.). 2316–5170 . Federal University of Paraná (UFPR) and Pontifical University Catholic of Paraná (PUC-PR), Brazilian Computer Society, Curitiba, PR, Brazil, pp. 137–146, 2014.
- DOAN, A., HALEVY, A., AND IVES, Z. *Principles of Data Integration*. Elsevier Science, 2012.
- DONG, X., BERTI-EQUILLE, L., AND SRIVASTAVA, D. Data fusion: Resolving conflicts from multiple sources. In *Web-Age Information Management*, J. Wang, H. Xiong, Y. Ishikawa, J. Xu, and J. Zhou (Eds.). Lecture Notes in Computer Science, vol. 7923. Springer Berlin Heidelberg, pp. 64–76, 2013.
- EUZENAT, J. AND SHVAIKO, P. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.
- FLORESCU, D. AND RASCHID, L. A methodology for query reformulation in cis using semantic knowledge, 1996.
- FRANKLIN, M., HALEVY, A., AND MAIER, D. From databases to dataspace: a new abstraction for information management. *SIGMOD Rec.* 34 (4): 27–33, 2005.
- GODFREY, P. AND GRYZ, J. A framework for intensional query optimization. In *Proceedings of the Workshop on Deductive Databases and Logic Programming*, GMD-Studien Nr. 295. pp. 57–68, 1996.

- HAASE, P., MATHASS, T., AND ZILLER, M. An evaluation of approaches to federated query processing over linked data. In *Proceedings of the 6th International Conference on Semantic Systems*. I-SEMANTICS '10. ACM, New York, NY, USA, pp. 51–59, 2010.
- HALEVY, A., RAJARAMAN, A., AND ORDILLE, J. Data integration the teenage years. In *Proceedings of the 32nd international conference on Very large data bases*. VLDB '06. VLDB Endowment, pp. 9–16, 2006.
- HARTIG, O. AND LANGEGER, A. A database perspective on consuming linked data on the web, 2010.
- LANGEGER, A. *A Flexible Architecture for Virtual Information Integration based on Semantic Web Concepts*. Ph.D. thesis, Johannes Kepler Universitt Linz, 2010.
- LEME, L. A. P. P., CASANOVA, M. A., BREITMAN, K. K., AND FURTADO, A. L. Instance-based owl schema matching. In *Enterprise Information Systems*. Vol. 24. Springer Berlin Heidelberg, pp. 14–26, 2009.
- LENZERINI, M. Data integration: A theoretical perspective. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS '02. ACM, New York, NY, USA, pp. 233–246, 2002.
- LÓSCIO, B. F. *Managing the Evolution of XML-Based Mediation Queries*. Ph.D. thesis, Federal University of Pernambuco, Center of Informatic, 2003.
- MAGALHÃES, R. P., MONTEIRO, J. M., VIDAL, V. M. P., DE MACÊDO, J. A. F., MAIA, M., PORTO, F., AND CASANOVA, M. A. Qef-ld - a query engine for distributed query processing on linked data. S. Hammoudi, L. A. Maciaszek, J. Cordeiro, and J. L. G. Dietz (Eds.). SciTePress, pp. 185–192, 2013.
- MAKRIS, K., BIKAKIS, N., GIOLDASIS, N., AND CHRISTODOULAKIS, S. Sparql-rw transparent query access over mapped rdf data sources. In *Proceedings of the 15th International Conference on Extending Database Technology*. EDBT '12. ACM, New York, NY, USA, pp. 610–613, 2012.
- MUNIR, K., ODEH, M., AND McCLATCHEY, R. Ontology assisted query reformulation using the semantic and assertion capabilities of owl-dl ontologies. In *IDEAS*. pp. 81–90, 2008.
- QUILITZ, B. AND LESER, U. Querying distributed rdf data sources with sparql. In *Proceedings of the 5th European Semantic Web Conference on The Semantic Web: Research and Applications*. ESWC'08. Springer-Verlag, Berlin, Heidelberg, pp. 524–538, 2008.
- SACRAMENTO, E. R., VIDAL, V. M. P., DE MACÊDO, J. A. F., LÓSCIO, B. F., LOPES, F. L. R., AND CASANOVA, M. A. Towards automatic generation of application ontologies. *JIDM* 1 (3): 535–550, 2010.
- SALGADO, A. C., LÓSCIO, B. F., DA CONCEIÇÃO MORAES BATISTA, M., BELIAN, R. B., PIRES, C. E. S., AND SOUZA, D. The data integration research group at ufpe. *JIDM* 2 (2): 109–122, 2011.
- SCHWARTE, A., HAASE, P., HOSE, K., SCHENKEL, R., AND SCHMIDT, M. Fedx a federation layer for distributed query processing on linked open data. In *ESWC (2)*. pp. 481–486, 2011.
- SUNG, L. G. A., AHMED, N., BLANCO, R., LI, H., SOLIMAN, M. A., AND HADALLER, D. A survey of data management in peer-to-peer systems, 2005.
- WIEDERHOLD, G. Mediators in the architecture of future information systems. *IEEE COMPUTER* 25 (3): 38–49, 1992.
- XIAO, H. *Query Processing for Heterogeneous Data Integration Using Ontologies*. Ph.D. thesis, University of Illinois at Chicago, 2006.