King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

Reformulating XQuery queries using GLAV mapping and complex unification

Saber Benharzallah [*], **Hammadi Bennoui, Okba Kazar**

Smart Computer Science Laboratory, Biskra University 07000, Algeria

Received 16 March 2014; revised 18 December 2014; accepted 3 June 2015
Available online 21 October 2015

KEYWORDS

Data integration;
Mediator;
XML;
XQuery;
GLAV mapping

Abstract This paper describes an algorithm for reformulation of XQuery queries. The mediation is based on an essential component called mediator. Its main role is to reformulate a user query, written in terms of global schema, into queries written in terms of source schemas. Our algorithm is based on the principle of logical equivalence, simple and complex unification, to obtain a better reformulation. It takes XQuery query, global schema (written in XMLSchema), and mappings GLAV as input parameters and provides resultant query written in terms of source schemas. The results of implementation show the proper functioning of the algorithm.

1. Introduction

Now the Web is presented as the most favored means to disseminate information. Many companies and organizations, whatever their field of activity (e-commerce, education, geographical or historical applications, etc.), make this choice for disseminating information.

The diversity of distributed information sources and their heterogeneity are one of the main difficulties encountered by users of the Web. It requires the user to respect the access methodology for each data source, implying to know the location of the base, the description of their content, the possibilities of interrogation and the format of results, in order to receive the expected response (Hacid and Reynaud, 1998).

The mediator-based systems offer interesting solutions for the integration of heterogeneous data. Accordingly, most recent works have taken this approach including the Internet-Oriented Systems (Moussa, 2002; Elazami et al., 2007; Mustafa and Rahman, 2013). The mediator acts as an interface between users and data sources. It is composed of a global schema, which provides a unified view of data sources and a set of views describing the content of sources. Queries are then expressed on the global schema, giving users the illusion of querying a single database. Based on information provided by the views, the mediator analyzes and reformulates the queries into sub-queries that would be executed by data sources. Before being sent to the target data source, each sub-query is translated into the native language of the source by the corresponding wrapper. The mediator uses the schema mappings to reformulate queries. Schema mappings establish a correspondence between data stored in two databases, called source and target respectively. Query processing under schema mappings has been investigated extensively in the two cases

[*] Corresponding author.
E-mail addresses: sbharz@yahoo.fr (S. Benharzallah), bennoui@gmail.com (H. Bennoui), kazarokba@yahoo.fr (O. Kazar).
Peer review under responsibility of King Saud University.

where each target atom is mapped to a query over the source (called GAV, Global-As-View), and where each source atom is mapped to a query over the target (called LAV, Local-As-View). The general case, called GLAV, in which queries over the source are mapped to queries over the target, has recently attracted a lot of attention (Calvanese et al., 2012). The mediator approach has the advantage of being able to build a query data sources system without touching the data remaining in their original sources.

XML is an extremely versatile markup language, capable of labeling the information content of diverse data sources including structured and semi-structured documents, relational databases, and object repositories (XQuery 1.0, 2007). A query language which uses the structure of the XML can intelligently express queries in all types of data that are physically stored in XML or viewed as XML via middleware. Because the query languages were traditionally designed for specific data types, the majority of existing proposals for the XML query languages are robust for some types of data sources, but weak for others. The specification of XQuery (XQuery 1.0, 2007) describes a new query language, which is conceived to be largely applicable to all types of XML data sources.

Most query reformulation algorithms (Koch, 2002; Arenas et al., 2004; Libkin and Sirangelo, 2008) using GLAV mapping approach exploit conjunctive queries; and consequently are not expressive or are applied in the fields of data exchange.

In this paper we describe a reformulation algorithm of XQuery queries for mediator based systems. The main role of the mediator is to reformulate a user query, written in terms of global schema, into queries written in terms of source schemas. Our algorithm is based on the principle of logical equivalence and simple/complex unification to obtain a better reformulation. The algorithm avoids the shorts reformulations. It takes XQuery query, global schema (written in XMLSchema), and expressive mappings GLAV (written in XQuery language) and provides resultant query written in terms of source schemas. The results of implementation show the proper functioning of the algorithm.

The rest of this paper is organized as follows: Section 2 presents the related work, some solutions presented in the literature and the characteristics of our solution. Section 3 outlines some concepts used in this paper. Section 4 presents the proposed architecture of our system of mediation and describes the reformulation algorithm. The programming environment and implementation are presented in Section 5. Finally, Section 6 concludes and prospects the paper.

2. Related work

The two main problems posed by the construction of a mediator are (Rousset et al., 2002): (i) the choice of both the language used to model the global schema, and the languages used to model, according to this schema, the views on the sources to be integrated as well as queries of users. And, (ii) the choice of query reformulation algorithm in terms of views in order to get all the answers to a query.

Studies have focused on the languages for modeling the global schema to represent the views of the sources to integrate and those used to express queries from human users or computing entities (Reynaud and Safar, 2008; Goasdoue et al., 2000). Others have focused on the design and implementation of algorithms for query rewriting in terms of views on relevant data sources and, more recently, some research focus on designing intelligent interfaces assisting the user in Query formulation (Maiz et al., 2006; Charlet et al., 2003).

Most of research (Calvanese et al., 2012; Halevy et al., 2006) on query processing under schema mappings in data integration distinguish three approaches to establish mappings between the global schema and source schemas. In GAV mappings, for each relationship used in the global schema, we define a view written using source schemas. The main advantage of this approach is its simplicity of reformulation. Nevertheless, it lacks the flexibility with respect to the addition, deletion and modification of the sources to the data integration system. This is due to the fact that each modification of a local source schema results in a modification of the global one. The projects TSIMMIS (Garcia-Molina et al., 1997), INFORMIX (Leone et al., 2005) follow the GAV approach.

In LAV mappings, every relationship of a source schema is defined as a view on the global schema. In this approach, each source is independently specified, which permits to provide more flexibility with respect to the addition/deletion of data sources to integrate. It has no effect on the global schema, only views should be added (or deleted). On the other hand, the price to pay for this flexibility is the complexity of the construction of answers to a query in the designed mediator. The projects STYX (Amann et al., 2002), Agora (Manolescu et al., 2001), follow the LAV approach.

GLAV mappings (Reynaud and Safar, 2008; Djema et al., 2007) overcome the limitations of both GAV and LAV (Friedman et al., 1999). In the query reformulation of the GLAV approach, each mapping rule is represented by a conjunctive query written in the global schema associated with a conjunctive one written in source schemas. These queries are virtual views that do not represent the results stored on sources, rather than LAV approach where each source may be regarded as it contains a response to a query written in the global schema; and consequently, the sources represent materialized answers to written queries on the global schema. Thus, in GLAV approach, the rules allow to reformulate the query more efficiently. Additionally, it reaches the limits of the expressive power of a data source description language. And also the query reformulation is a co-NP-hard in the size of the data in the sources. Query reformulation in this approach is shown to be no harder than that of the LAV approach. In fact, most of the research on query processing under schema mappings in data integration concentrate on GAV and LAV mappings (see, for instance the surveys in Calvanese et al. (2012) and Halevy et al. (2006)).

In data integration, GLAV mappings were specifically taken into account in Cal (2004), but only in the case of relational databases. It was mainly studied in the exchange of information. In particular, the focus of Friedman et al. (1999) and Levy et al. (2000) is put on providing foundation for exchange of information based on schema mappings; whereas in Florescu (1996), Arenas et al. (2010) and Fagin et al. (2009), the goal is to study operators on schema mappings relevant to model management, notably, composition, merge, and inverse (Calvanese et al., 2012). A more general form of GLAV that accounts for XML like structures, and which we will use here, has been used to give semantics for mappings between XML schemas and to generate the data transformation scripts (in SQL, XQuery or XSLT) that implement the desired data exchange (Libkin and Sirangelo, 2008; Yu and Popa, 2004).

In our solution, we adopt GLAV approach to define mapping rules by using the XQuery language. The solution provided in this paper is characterized by the following features:

– The use of common expressive query language XQuery to express queries from human users or computing entities.
– The use of the XMLSchema model as a common data model to represent the global schema as well as the views of the sources to integrate.
– Backward integration approach and adaptation of GLAV mapping rules.
– The reformulation algorithm is based on the principle of logical equivalence and simple and complex unification to obtain a better reformulation. The GLAV mappings rules take into account the resolution of semantic conflicts.

3. Preliminaries

This section outlines briefly some basic concepts on which we will rely throughout the paper.

– Substitution: A substitution of a set of variables $X = (x_1, x_2, \ldots, x_n)$ is the finite set of the form: $\{x_1/y_1, x_2/y_2, \ldots, x_n/y_n\}$ where each y_i is a variable different to x_i but it has the same type as x_i.
– Instance: Let the substitution $\theta = \{x_1/y_1, x_2/y_2, \ldots, x_n/y_n\}$ and Q a query. Consider the following queries: $Q_1, Q_2, \ldots, Q_n$where: $Q_0 = Q$ and Q_i are obtained from$Q_{i-1}$by y_i. Q_n is called the instance of Q by the substitution and is denoted by $Q\theta$.
– Logical equivalence: Two queries Q_1, Q_2 are logically equivalent if and only if they give the same results (have the same canonical form) (Florescu, 1996).
– Simple form: A query is in a simple form if all the predicates in the *Where* clause are in conjunctive normal form. There are no imbrications in a *For* clause.
– Mapping rules: They are defined for the correspondence between the global and source schemas.
  The rules are of the form: $R_i : q_g \rightarrow q_s$, where:
  q_g: is an XQuery query relating to elements of the global schema. q_s: is an XQuery query relating to elements of source schemas.

4. The proposed architecture

4.1. General architecture of the system

Our mediator uses a common expressive query language XQuery to express queries from human users. Besides the GLAV mapping rules, it uses a XMLSchema model as a common data model to represent the global schema as well as the views of the sources to integrate. In order to obtain a better reformulation, the algorithm exploits the principle of logical equivalence and simple/complex unification. The mediator is composed of three modules (Fig. 1): user interface, query analyzer and query reformulation.

- *User interface*: The interface presents the only mean that allows direct interaction between the system and the user.

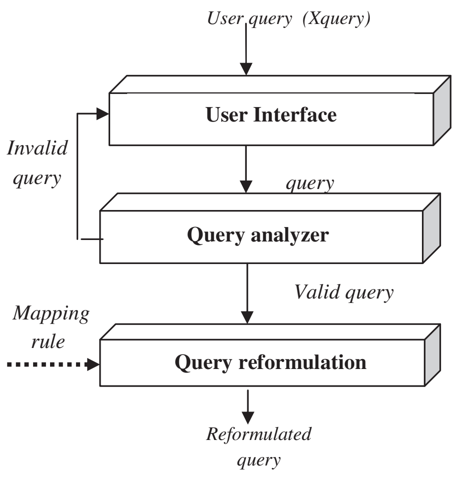


Figure 1 General architecture of our system.

- *Query analyzer* This analyzer allows a lexical and syntactic analysis of the query to verify its validity.
- *Query reformulation*: This module decomposes a query Q written in global schema into a recomposition query and sub-queries. Each sub-query $q_i$is written in terms of a source schema.

The global schema offers the illusion that the user asks a centralized system. In general, when the user puts his query (in terms of the global schema) via the user interface, the analysis module checks its lexical and syntactic validity (the query must be expressed in terms of the global schema and respects the adopted syntax). If all goes well, the reformulation module tries to find a reformulation for the query. First, it must transform the query to a simpler form (the canonical form); next it exploits the available mapping rules to reformulate the query. If the query is reformulated successfully, then the module identifies the participant sources in the execution of Q.

4.1.1. Query analyzer

This module analyzes the query, knowing that it is written in a restriction of the XQuery language. This restriction is generated by the following grammar (Yu and Popa, 2004):

```
Q: = For $x1 in C1,…,$xn in Cm
Where B
Return R
R: = [A1: = R1,…,Ak = Rk] | E | Q
E: = S | $x | E/L
Ci: = E | Q
$x: is a variable
S: is the root of schema
L: is a label
E/L: recording of projection
```

In fact, this grammar is the heart of XQuery (Yu and Popa, 2004). The analyzer decomposes the user query into an internal structure that can be easily manipulated by the various components of the mediator. It also checks whether the query is valid, both syntactically and in relation to the surveyed data types.

4.1.2. Query reformulation module

For each relation in the global schema, we will define a view consisting of the terms of source schemas relations. The reformulation (Algorithm 1) consists itself of two sub components (Florescu, 1996): simple and complex unification.

a - Simple unification

Two queries Q_1 and Q_2 are unifiable if Q_1 is an instance of by the substitution θ; this means that: $Q_1 = Q_2\theta$. We say in this case that Q_1 is logically equivalent to $Q_2\theta$.

We adapt the algorithm defined in Florescu (1996) which allows verifying the unification of two OQL queries.

The unification is simply divided into three main stages:

- The unification of collections (*Ci*).
- The unification of predicates.
- The unification of projections (return).

If all goes well, the unification succeeds and returns the substitution θ. The substitution θ is calculated iteratively and we obtain a θ such that: $Q_1 = Q_2\theta$.

b - Complex unification

If two queries Q_1 and Q_2 are not unifiable by the simple unification, then it may be possible that there exists a query Q_3 which is logically equivalent to Q_2 and it contains $Q_1\theta$ as a sub query.

We say that Q_3 is written in terms of $Q_1\theta$ which is unified by the substitution θ. Thus, Q_3 is the reformulation of Q_2 by using Q_1. We adapt the algorithm defined in Florescu (1996) which allows verifying from two queries Q_1 and Q_2, if it's possible to reformulate Q_2 in a query Q_3 containing $Q_1\theta$ such as a sub query.

The complex unification is divided into three main stages:

- The unification of collections.
- The unification of predicates.
- The construction of the new query Q_3 and the substitution θ.

```
Algorithm 1 Reformulation
Input: Q (written in XQuery), M // where
$M = \{r_1, r_2, \ldots, r_n\}$ and $r_i : q_{g_i} \rightarrow q_{s_i}$.
Output: E1
{
$E_1 = \{Q\}$/* Q is in simple form */
While not exist reformulated query in E1
        AND E1 not empty do
  {
$E_2 = \{\}$;
For each $q \in E_1$ and $r_i \in M${
ifUnificationSimple ($q$, $q_{g_i}$)==true then
replace $q$ by $q_{g_i}\theta$/*$\theta$ is the substitution*/
Else: If UnificationComplexe($q_{g_i}$,$q$)==true
/* successful with the substitution $\theta$ and
the query $q'$ */then
    Replace $q$ by $q'$ (as $q'$ contains $q_{g_i}\theta$ like sub
query).
    Add the result to $E_2$
    }
              $E_1 = E_2 - E_1$
}
}
```

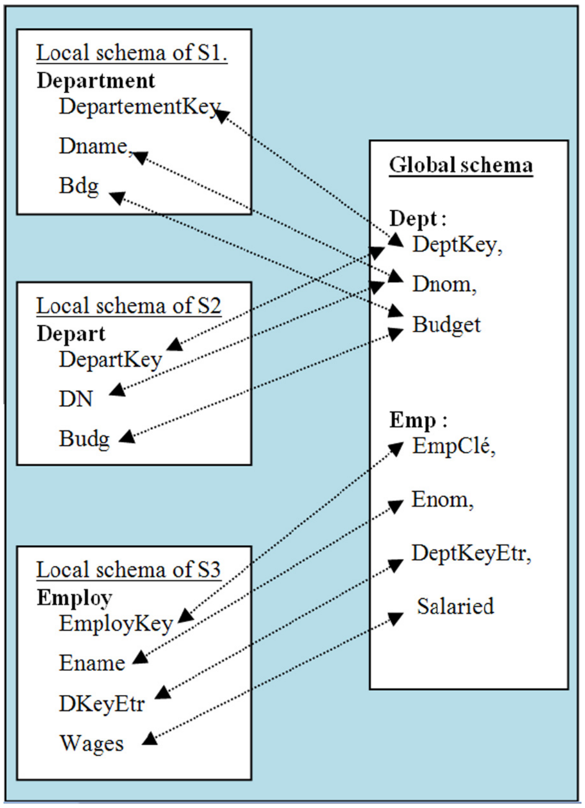


Figure 2 Relations between different schemas.

4.1.2.1. Process of reformulation. We describe in our solution the decomposition process of a query Q written in global schema into a recomposition query and sub-queries. Each sub-query q_i is written in a source schema S_i.

Our process of reformulation is accomplished in four stages: transformation of the query Q into a more simple form to be processed, reformulation, identification of sources involved in the execution of the query and the generation of sub-queries.

- Stage 1: The transformation of the query is to write it in the canonical form or approximate it to the canonical form.
- Stage 2: The reformulation of a query Q (Algorithm1): our algorithm consists to reformulate a query Q (using mapping rules M) into a query logically equivalent to Q and written in terms of $q_{g_i}\theta$. There are three cases: the case in which there exists a rule $r_i : q_{g_i} \rightarrow q_{s_i}$ such that $Q = q_{g_i}\theta$; the case where $Q$is in terms of $q_{g_i}\theta$; and the case where there doesn't exist a reformulation of the query because of the lack of mapping rules. In this last case, the algorithm gives failure as a result. We propose that the mapping rules follow an order of priority to ensure proper reformulation and also allow to take into account the constraints on the sources that are defined in the mapping rules. So the algorithm should avoid short reformulations.
- Stage 3: The identification of data sources involved in the execution of the query (Algorithm 2) is performed using the mapping defined in the mapping rules between q_{g_i} et q_{s_i}.

Table 1 Mapping rules.

r_1	q_{g_1}	for $zincollection(“GlobalSchema”)/Dept
		where $z/Dnom = $a
		return $z

	q_{s_1}	for $zin(for $t in collection(“LocalSchema-Source1”)/Department where return [DeptKey = $t/DepartmentKey, Dnom = $t/Dname, Budget = $t/bdg]) where $z/Dnom = $a return $z **union** for $zin(for $t in collection(“LocalSchema-Source2”)/Depart where return [DeptKey = $t/DeparKey, Dnom = $t/DN, Budget = $t/budg]) where $z/Dnom = $a return $z

r_2	q_{g_2}	for $zincollection(“GlobalSchema”)/Emp
		where $z/Salaried = $a
		return [A1 = $z/Enom, A2 = $z/DeptKeyEtr]

	q_{s_2}	for $zincollection(“LocalSchema-Source3”)/Employ
		where $z/wages = $a
		return [A1 = $z/Ename, A2 = $z/DKeyEtr]

```
Algorithm 2 Identification Sources Participating
Input: a reformulated Q;
$M = \{r_1, r_2, \ldots, r_n\}$ and $r_i : q_{g_i} \rightarrow q_{s_i}$;
Output: Q in term of source schemas;
{
  For each $r_i \in M$ {
  If $q_{g_i}$ appears in Q then replace $q_{g_i}$ by $q_{s_i}$ in Q;
  }
returnQ
}
```

- Stage 4: The generation of sub-queries from the query Q. We distinguish between two cases: – if the query Q has the form $q_{s_i}\theta$ (special case) then Q is the union of sub-queries, each of which is written on the elements of a source schema; – If the query Q is written in terms of $q_{s_i}\theta$ then, in this case, the query Q is considered as recombining a query and $q_{s_i}\theta$ are considered as sub-queries, each of which can contain unions of sub-queries. The sub-queries for each source involved in the execution of Q are grouped altogether to be sent to these sources.

4.2. Experimental results

We have implemented our prototype using the environment C+ + Builder. We present the following case study to demonstrate the operation of the algorithm. We have the global schema *GS* and source schemas S_1, S_2 et S_3 representing databases ûDepartment, Employersý. The global schema is as follows:

Global Schema:
Dept(DeptKey, Dnom, Budget);
Emp(EmpClé, Enom, DeptKeyEtr, Salaried);

The Global Schema is written in XMLSchema and interrogated by XQuery.

Source schemas are:
Local schema of the source S_1:
Department(DepartmentKey, Dname, Bdg);
Local schema of the source S_2 :
Depart(DepartKey, DN, Budg);
Local schema of the source S_3:
Employ(EmployKey, Ename, DKeyEtr, wages);

Fig. 2 presents relations between the different schemas. In our case study, we used the subset of GLAV mapping rules represented in Table 1.

For example, consider the following user query Q:
Q = for $x in collection (“GlobalSchema”)/Dept, $y in collection (“GlobalSchema”)/Emp
Where $x/DeptKey = $y/DeptKeyEtrand $x/Dnom = “department1” and
  $y/Salaried = “20000,00 DA”
return [Name = $y/Enom]

The user wants to know the names of employers in department1 who have wages equal to 20000,00 DA.

To decompose Q into sub-queries in our mediator, we apply the following steps:

- Simplification of the query: Q is in a simple form.
- Reformulation: we apply our algorithm of query reformulation. The query is reformulated using q_{g_1} and q_{g_2}.

Q = for $x0 in $q_{g_1}\theta$, $x01 in $q_{g_2}\theta'$
where $x0/DeptClé = $x01/A2
return [Nom = $x01/A1]
where: $\theta = \{\$\ z/\$\ x, \$\ a/\text{“department1”}\}$ and
$\theta' = \{\$\ z/\$\ x, \$\ a/\text{“20000, 00DA”}\}$.

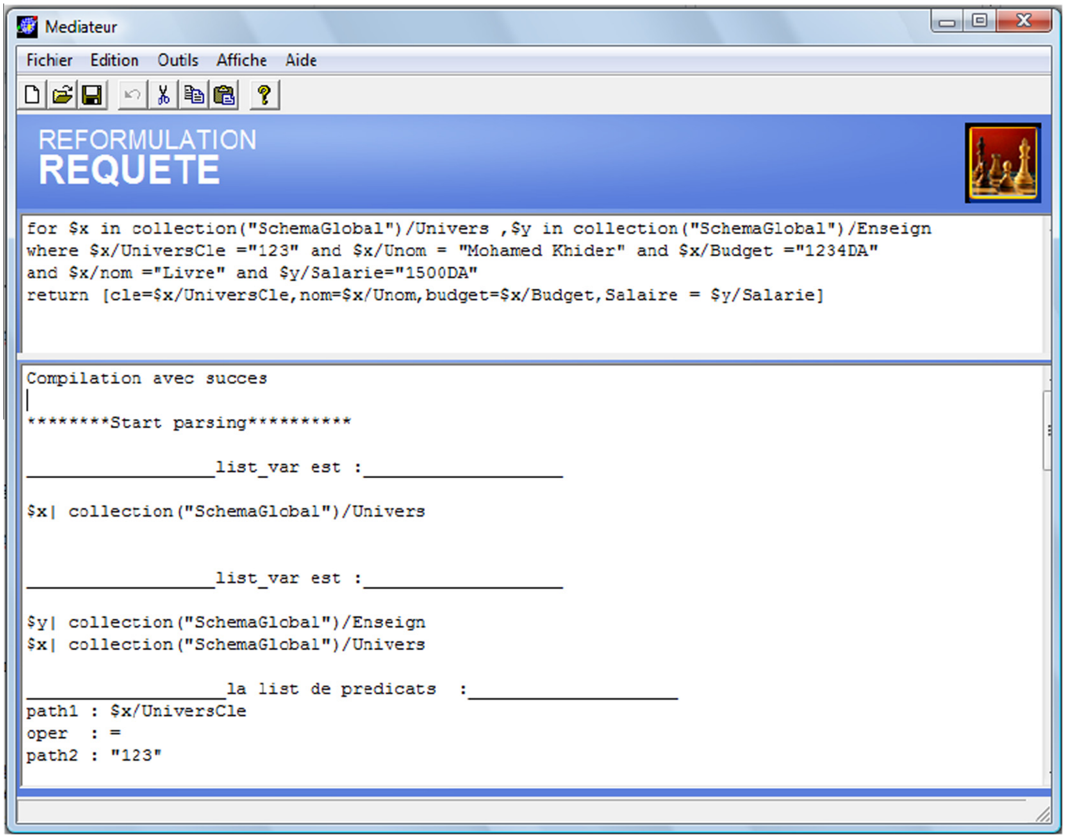
Table 2 Sub queries sent to S_1, S_2, and S_3.

Sub query	Sent to

for $x in (for $t in collection(“LocalScheam-source1”)/Department
where return [DeptKey = $t/DepartmentKey, Dnom = $t/Dname, Budget = $t/bdg])
where $x/Dnom = “department1”
return $x
	S_1

for $x in (for $t in collection(“LocalScheam-source2”)/Depart
where return [DeptKey = $t/DeparKey, Dnom = $t/DN, Budget = $t/budg])
where $x/Dnom = “department1”
return $x
	S_2

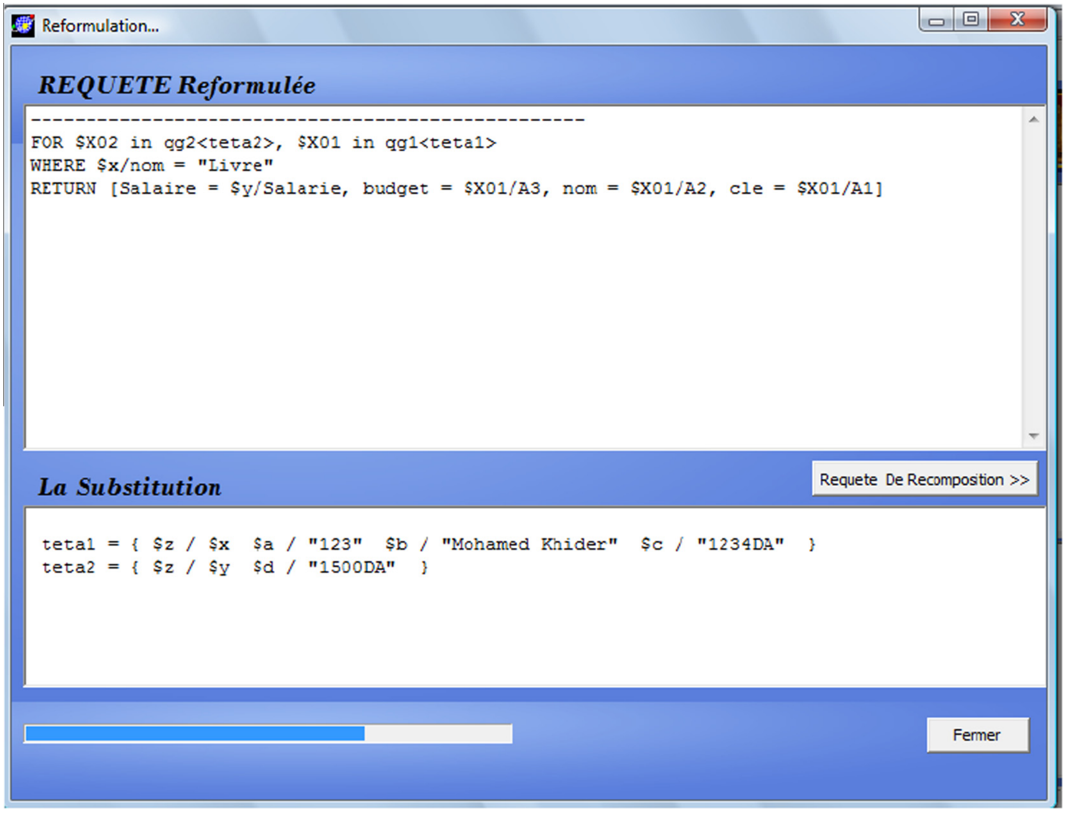
for $x in collection(“LocalScheam-source3”)/Employ
where $x/wages = “20000,00DA”
return [A1 = $x/Ename, A2 = $x/DCléEtr]
	S_3



```
for $x in collection("SchemaGlobal")/Univers ,$y in collection("SchemaGlobal")/Enseign
where $x/UniversCle ="123" and $x/Unom = "Mohamed Khider" and $x/Budget ="1234DA"
and $x/nom ="Livre" and $y/Salarie="1500DA"
return [cle=$x/UniversCle,nom=$x/Unom,budget=$x/Budget,Salaire = $y/Salarie]
```

```
Compilation avec succes

********Start parsing**********

_________________list_var est :__________________

$x| collection("SchemaGlcbal")/Univers


_________________list_var est :__________________

$y| collection("SchemaGlcbal")/Enseign
$x| collection("SchemaGlcbal")/Univers

__________________la list de predicats   :___________________
path1 : $x/UniversCle
oper  : =
path2 : "123"
```

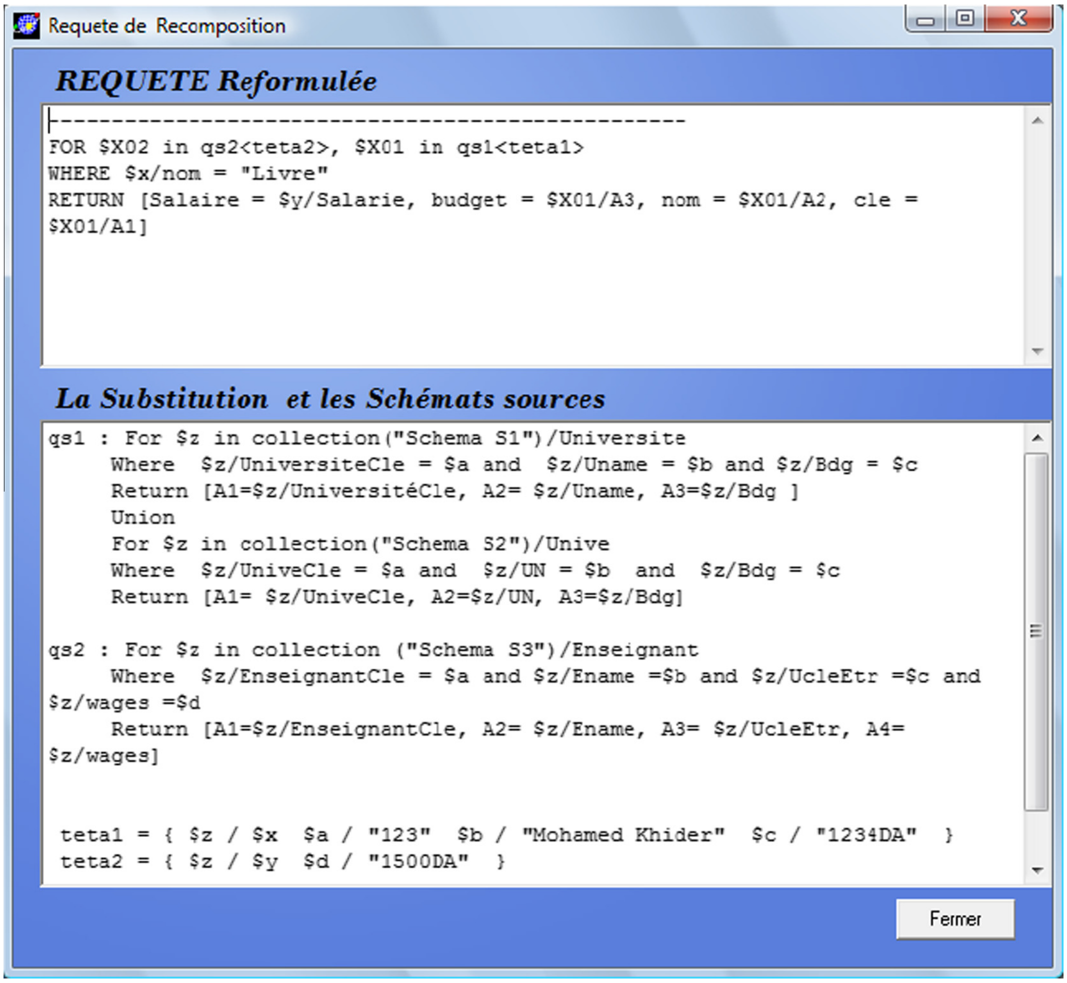
Figure 3 The result of compilation.



```
--------------------------------------------------
FOR $X02 in qg2<teta2>, $X01 in qg1<teta1>
WHERE $x/nom = "Livre"
RETURN [Salaire = $y/Salarie, budget = $X01/A3, nom = $X01/A2, cle = $X01/A1]
```

```
teta1 = { $z / $x  $a / "123"  $b / "Mohamed Khider"  $c / "1234DA"  }
teta2 = { $z / $y  $d / "1500DA"  }
```

Figure 4 The result of reformulation.



```
|-------------------------------------------------
FOR $X02 in qs2<teta2>, $X01 in qs1<teta1>
WHERE $x/nom = "Livre"
RETURN [Salaire = $y/Salarie, budget = $X01/A3, nom = $X01/A2, cle =
$X01/A1]
```

```
qs1 : For $z in collection("Schema S1")/Universite
     Where  $z/UniversiteCle = $a and  $z/Uname = $b and $z/Bdg = $c
     Return [A1=$z/UniversitéCle, A2= $z/Uname, A3=$z/Bdg ]
     Union
     For $z in collection("Schema S2")/Unive
     Where  $z/UniveCle = $a and  $z/UN = $b  and  $z/Bdg = $c
     Return [A1= $z/UniveCle, A2=$z/UN, A3=$z/Bdg]

qs2 : For $z in collection ("Schema S3")/Enseignant
     Where  $z/EnseignantCle = $a and $z/Ename =$b and $z/UcleEtr =$c and
$z/wages =$d
     Return [A1=$z/EnseignantCle, A2= $z/Ename, A3= $z/UcleEtr, A4=
$z/wages]


 teta1 = { $z / $x  $a / "123"  $b / "Mohamed Khider"  $c / "1234DA"  }
 teta2 = { $z / $y  $d / "1500DA"  }
```

Figure 5 The result of the queries sent to sources.

- The identification of sources involved in the execution of the query$Q$: q_{g_1}corresponds to q_{s_1}(so the sources A1 and A2 are involved in the execution of Q) and q_{g_2}corresponds toq_{s_2}(so the source A3 participates in the execution of Q). Therefore the three sources take part in the execution of the query.
- After the identification of the sources participating in the execution of the query Q, our mediator sends, using Algorithm 2, the sub-queries that will be executed at the sources (Table 2). Finally, the mediator recomposes the results.

Figs. 3–5 present the results of our system. Fig. 3 presents the compilation of a user query which has been passed successfully and it displays the steps of lexical and syntactic verification. Fig. 4 presents the query reformulated in terms of q_{g_i} and it also displays the two substitutions where they helped to reformulate the request of the user to the correct way. In this case, the reformulation is of a complex type, since the algorithm could not find a direct reformulation of Q. Fig. 5 also shows the reformulated query with sub queries to be sent to data sources.

5. Conclusion

A mediation system is a powerful means allowing an easy access to various information collected from data sources that can be quite disparate. It must integrate diverse data in order to provide to the user a centralized and uniform view of data by hiding the features specific to their location, access method and formats. We presented in this paper a reformulation algorithm of XQuery queries for mediation systems using GLAV mappings and unification.

The implementation of the prototype illustrates the operation of the algorithm. We tested the algorithm (the prototype) on a case study across multiple queries, and we showed in this paper an example of algorithm execution through a query. This example shows in detail the running of the algorithm on a real case. Through our experimentation, we advise that the GLAV mapping rules follow an order of priority to ensure proper reformulation and also allow to take into account the constraints on the sources that are defined in the mapping rules. So the algorithm should avoid short reformulations. In fact, the efficiency of the algorithm is related to the efficiency of the mapping rules and their order of priority. These priorities are defined by the administrator.

GLAV mapping combines the expressive power of GAV and LAV, and the query reformulation is a co-NP-hard in the size of the data in the sources. Query reformulation in our approach is no harder than that of the LAV approach.

As a prospect, our mediation system will be improved by taking into account the following points: the use of all possibilities of XQuery language and the construction of adapters to resolve structural conflicts of heterogeneous sources (XML schema model, relational model, ... etc).

References

Amann, B., Beeri, C., Fundulaki, I., Scholl, M., 2002. Querying xml sources using an ontology-based mediator. In: CoopIS/DOA/ODBASE, pp. 429–448.

Arenas, M., Barcelo, P., Fagin, R., Libkin, L., 2004. Locally consistent transformations and query answering in data exchange. In: Proc. of the 23rd ACM Symp. on Principles of Database Systems (PODS 2004), pp. 229–240.

Arenas, M., Fagin, R., Nash, A., 2010. Composition with target constraints. In: Proc. of the 13th Int. Conf. on Database Theory (ICDT 2010), pp. 129–142.

Cal, A., 2004. Query answering by rewriting in GLAV data integration systems under constraints. In: Proc. of the 2nd Int. Workshop on Semantic Web and Databases (SWDB 2004), Volume 3372 of Lecture Notes in Computer Science. Springer, pp. 167–184.

Calvanese, Diego. et al, 2012. Query processing under GLAV mappings for relational and graph databases. In: Proceedings of the VLDB Endowment, vol. 6, no. 2.

Charlet, J., Laublet, P., Reynaud, C., 2003. Rapport final Web sémantique, Action spécifique 32 CNRS/STIC.

Djema, L., Boumghar, F., Debiane, S., 2007. L'imagerie Médicale Dans une Base De Données Distribuée Multimédia Sous Oracle 9i, 4th International Conference: Sciences of Electronic, Technologies of Information and Telecommunications, March 25–29, Tunisi, pp. 3–5.

Elazami, I., Doukkali, D., Cherkaoui, O., 2007. Approche à base de Patterns pour la Médiation entre les Systèmes d'Information Hospitaliers, Département de Mathématiques et Informatique Faculté des Sciences, Séminaire SIM 07, Dhar EL Mahraz Fès, Maroc. FMP de Fès, 02 juin 2007, pp. 1–8.

Fagin, R., Kolaitis, P.G., Popa, L., Tan, W.C., 2009. Reverse data exchange: coping with nulls. In: Proc. of the 28th ACM Symp. on Principles of Database Systems (PODS 2009), pp. 23–32.

Florescu, D., 1996. Espaces de Recherche pour l'Optimisation de Requêtes Objet (thèse de doctorat). Université de Paris VI, France.

Friedman, M., Levy, A., Millstein, T., 1999. Navigational plans for data integration. In: Proc. of the National Conf. on Artificial Intelligence (AAAI), pp. 67–73.

Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J.D., Vassalos, V., Widom, J., 1997. The TSIMMIS approach to mediation: data models and languages. J. Intell. Inf. Syst. 8 (2), 117–132.

Goasdoue, François, Lattes, V., Rousset, M.-C.H., 2000. The use of the Carin language and algorithms for Integration Information: the PICSEL system. Int. J. Cooperative Inf. Syst. 9 (4), 383–401.

Hacid, M.S., Reynaud, C., 1998. L'intégration de sources de données (thèse de doctorat), Université Claude Bernard Lyon 1, France.

Halevy, A.Y., Rajaraman, A., Ordille, J., 2006. Data integration: the teenage years. In: Proc. of the 32nd Int. Conf. on Very Large Data Bases VLDB2006.

Koch, C., 2002. Query rewriting with symmetric constraints. In: Proc. of the 2nd Int. Symp. on Foundations of Information and Knowledge Systems (FoIKS 2002), Volume 2284 of Lecture Notes in Computer Science. Springer, pp. 130–147.

Leone, N. et al, 2005. The INFOMIX system for advanced integration of incomplete and inconsistent data. In: Proc. of the ACM SIGMOD Int. Conf. on Management of Data, pp. 915–917.

Levy, A.Y., 2000. Logic-based techniques in data integration (PhD). Department of Computer Science and Engineering, University of Washington.

Libkin, L., Sirangelo, C., 2008. Data exchange and schema mappings in open and closed worlds. In: Proc. of the 27th ACM Symposium Principles of Database Systems (PODS 2008), pp. 139–148.

Maiz, N., Boussaid, O., Bentayeb, F., 2006. Un système de médiation basé sur les ontologiesý, Laboratoire ERIC Université Lumière Lyon 2, 17 Janvier 2006 Lille, France.

Manolescu, I., Florescu, D., Kossmann, D., 2001. Answering XML queries over heterogeneous data sources. In: Proc. of the 27th Int. Conf. on Very Large Data Bases (VLDB 2001).

Moussa, L.O., 2002. Dataweb basés sur XML: modélisation et recherche d'informations pertinentes (Thèse de Doctorat d'Informatique), Université de Pau et des Pays de l'Adour, France, 17 décembre 2002.

Mustafa, Rashed, Rahman, Hasan Hafizur, 2013. Mediator based architecture to address data heterogeneity. Daffodil Int. Univ. J. Sci. Technol. 8 (2), 30–40.

Reynaud, C., Safar, B., 2008. Construction automatique d'adaptateurs guidée par une ontologie pour l'intégration de sources et de données XML, inria-00432426, version 1, Univ. Paris-Sud, France, juin 2008.

Rousset, C., Bidault, Froidevaux, C., Gagliardi, H., Goasdoué, F., Reynaud, C., Safar, B., 2002. Construction de Médiateurs pour intégré des sources d'information multiples et hétérogènes le projet PICSEL, Université Paris_Sud, Information-Interaction-Intelligence, vol. 2, no. 1, France.

XQuery 1.0, 2007. An XML Query Language, W3C working Draft.

Yu, C., Popa, L., 2004. Constraint based XML Query Rewriting for data integration, SICMOD 2004, juin 2004, Paris, France, pp. 371–382.