An Algorithm Reformulation of XQuery Queries Using GLAV Mapping for Mediator-based System

Benharzallah SABER, 2Kazar OKBA
1,2Computer sciences Department, Biskra University, Algeria
1sbharz@yahoo.fr, 2kazarokba@yahoo.fr

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

Keywords: Data integration, Mediator, XML, XQuery, GLAV mapping.

I. INTRODUCTION

Now the Web is presented as the most favored mean 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 information sources distributed and their heterogeneity are one of the main difficulties encountered by users of the Web today. It requires the user to respect the access methodology for each data source, this implies to know the location of the base, the description of their content, the possibilities of interrogation, the format of results, in order to receive the expected response [1].

The Mediator-based System offer interesting solutions for the integration of heterogeneous data. For these reasons, the most recent works have taken this approach include the Internet-Oriented Systems [2] [3] [4]. 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 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 analysis and reformulates the queries into sub-queries 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 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 attracted a lot of attention recently [5]. 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.

In this paper we describe an algorithm for reformulation of XQuery queries. The main role of the mediator is to reformulate a user query, written in terms of global schema, in queries written in terms of sources schemas. Our algorithm is based on the principle of logical equivalence, simple and complex unification, to obtain a better reformulation. It takes as parameter the query XQuery, the global schema (written in XMLSchema), mappings GLAV and gives as a result a query written in terms of sources 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 problems studied, some solutions presented in the literature and the characteristics of our solution. Section 3 describes some concepts used in this paper. Section 4 presents the proposed architecture of our system of mediation and describes the algorithm reformulation. The programming environment and implementation are presented in Section 5. Finally a conclusion and prospects.

II. PROBLEMS STUDIED

The two main problems posed by the construction of a mediator are [6] - The choice of the language used to model the global schema, and the choice of the languages for modeling, according to this schema, the views on the sources to be integrated, and users queries. - And, depending on the choice and implementation of algorithms for query rewriting 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 to express queries from human users or computing entities [7] [8]. 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 focuses on designing intelligent interfaces assisting the user in query formulation [9] [10]. Most of the research on query processing under schema mapping in data integration has concentrated on GAV mappings, and on LAV mappings under the label of view-based query processing (see, for instance the survey in [11]) [5]. GLAV mapping overcome the limitations of both GAV and LAV [12].

The solution provided by this paper is characterized by the following points:
- The use of common expressive query language XQuery to express queries from human users or computing entities.
- The use of the model XML Schema as a common data model to model the global schema and to represent the views of the sources to integrate.
- Backward integration Approach and adaptation of GLAV mapping rules.
- The algorithm reformulation is based on the principle of logical equivalence, simple and complex unification, to obtain a better reformulation.
- The algorithm takes into account semantic conflicts, resolve them in the GLAV mappings rules.

III. DEFINITIONS

We describe some concepts used in this paper.

- **Substitution:** A substitution of the set of variables \( X = (x_1, x_2, ..., x_n) \) 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. Considering the following queries: \( Q_1, ..., Q_i, ..., Q_n \) where: \( Q_0 = Q \) and \( Q_i \) is obtained by \( Q_{i-1} \) by \( y_i. \) \( Q_n \) is called the instance of \( Q \) by the substitution \( \theta \), and is denoted by \( Q^\theta \).

- **Logical Equivalences:** Two queries \( Q_1, Q_2 \) are called logically equivalent if and only if they give the same results (have the same canonical form) [13].

- **Simple form:** A query is in simple form if all the predicates in the *Where* clause are in conjunctive normal form. There is no imbrication in clause *For*.

- **Mapping rules:** they are defined for the correspondence between the global and sources schemas. They also intervene in the reformulation of queries. The rules are of the form: \( R : q_g \rightarrow q_s \). Where: \( q_g \) is an XQuery query relating to elements of the global schema. \( q_s \) is an XQuery query relating to elements of sources schemas.

IV. THE PROPOSED ARCHITECTURE

A. General architecture of the system

In our solution we adopt as:

- **Query reformulation approach GLAV:** There are basically two approaches to build the link between a mediated schema and sources schemas. The LAV approach (Local-As-View) and the GAV (Global-As-View) [14] [15]. We chose the GLAV approach [7] [16] which presents a combination of both approaches, GLAV offer more flexibility to the updates of users or local sources.

- **Data Model:** The model of semi-structured data XML schema has been designed to easily represent irregular data from heterogeneous sources, structured or not. We also note that it is a flexible model used to represent irregular data by mixing structure and data [17]. And that's exactly the model suitable to our case, where data sources are heterogeneous, structured, semi structured or unstructured.

- **Query Language, XQuery:** The W3C proposed the XQuery language [18] which is more adopted for an interrogation of an XML schema. The XQuery language incorporates the benefits of XPath, XML-QL and XQL this language was designed to allow to create specific queries and can be adapted to any type of XML data source, whether databases or documents. For all these reasons and also because we have chosen XML schema as a common model, we have chosen the XQuery language.

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

- **Query analyzer:** This analyzer allows a lexical analysis, syntax and semantics on the query to verify its validity.

Our mediator is composed of three modules (figure 1):

![Fig. 1. General architecture of our system](image-url)
1) Query analyzer

Can analyze the query, knowing that it is written in a restriction of the XQuery language. This restriction is generated by the following grammar [19]:

\[
Q: = \text{For } x_1 \text{ in } C_1, \ldots, x_n \text{ in } C_n \\
\text{Where } B \\
\text{Return } R
\]

\[
E: = s \mid S x \mid E/L
\]

\[
C_i: = E \mid Q
\]

\[
\$x: \text{ is a variable}
\]

\[
S: \text{ is the root of schema}
\]

\[
L: \text{ is a label}
\]

\[
E/L: \text{ recording of projection}
\]

In fact, this grammar is the heart of XQuery [19]. The analyzer decomposes the query user 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 data types surveyed...

2) Query reformulation Model

For each relation in the global schema we will define a view consisting of the terms of the relations of source schemas. The reformulation consists itself of two sub components [13]:

- a - Simple unification
- b - Complex unification

Our reformulation algorithm (algorithm1 reformulation) is based on the tow concepts: simple and complex unification.

a - Simple unification:

Two queries Q1 and Q2, are unifiable if Q1 is an instance of Q2 by the substitution \( \theta \), that means that : \( Q_1 = Q_2^\theta \). we say that Q1 is logically equivalent to Q2^\theta.

We adapt the algorithm defined in [13] which allow to verify 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 \( \theta \). The substitution – is calculated iteratively and we obtain a \( \theta \) 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, it is possible to verify that if it exist a query Q_3 logically equivalent to Q_2 by the substitution \( \theta \). we say that Q1 is logically equivalent to Q2^\theta.

We say that Q_3 is written in terms of Q_1 by the substitution \( \theta \) which unify with Q_1 by the substitution \( \theta \), and Q_1 is the reformulation of Q_2 by using Q_1. We adapt the algorithm defined in [13] which allow to verify from two queries Q_1 and Q_2, if it’s possible to reformulate Q_1 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.
- Construction the new query Q_3 and the substitution \( \theta \).

Algorithm1 Reformulation

Input: Q (written in XQuery), M // where \( M = \{ r_1, r_2, \ldots, r_i, \ldots, r_n \} \) and \( f_i: q_{g_i} \rightarrow q_{s_i} \).

Output: E1

\[
E1 = \{ Q \} \\
\text{While not exist reformulated query in } E1 \text{ and } E1 \text{ not empty do}
\]

\[
E2 = \{ \} \\
\text{For each } q \in E1 \text{ and } r_i \in M \{ \}
\]

\[
\text{if UnificationSimple}(q, q_{g_i})=\text{true then}
\]

- replace q by \( q_{g_i}^\theta \) (\( \theta \) is the substitution)

[...]

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 source schema \( S_{ii} \).

Our process of reformulation will be in four stages: transform the query Q into a more simple form to process, 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 to the canonical form.
- Stage 2: The reformulation query Q (fig 3 algorithm reformulation): our algorithm consists to reformulate a query Q (using mapping rules M) into a query logically equivalent to Q and written in terms (s) \( q_{s_i}^\theta \). This stage prepare to the stage of identification of information sources participant to the execution of query Q. There are three cases:
  - The case which exist a rule \( r_i: q_{g_i} \rightarrow q_{s_i} \), as: \( Q = q_{s_i}^\theta \). 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 case the algorithm gives failure. 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 shorts reformulations.

- Stage 3: The identification of data sources involved in the execution of the query (algorithm 2 IdentificationSourcesPraticipating) is performed using the mapping defined in the mapping rules between $q_{g_1}$ et $q_{g_2}$.

**Algorithm 2 IdentificationSourcesPraticipating**

**Input** : $Q$ reformulated , $M = \{r_1,r_2,\ldots,r_i,\ldots,r_n\}$ and $r_i: q_{g_1} \rightarrow q_{g_2}$ .

**Output** : $Q$ in term of sources schemas.

For each $r_i \in M$

If $q_{g_1}$ apprers in $Q$ them replace $q_{g_1}$ by $q_{g_2}$ in $Q$

return $Q$

- Stage 4 : The generation of sub-queries from the query $Q$. We distinguish two cases: if the query $Q$ has the form (case): $q_{g_1} \theta$ (special case) : $Q$ 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 $q_{g_1} \theta$ in this case the query $Q$ is considered as recombining query and $q_{g_1} \theta$ are considered as sub-queries, each can contain unions of sub-queries. The sub-queries for each source involved in the execution of $Q$ are grouped to be sent to these sources.

V. PROGRAMMING ENVIRONMENT AND RESULTS OF IMPLEMENTATION

We have implemented our prototype using the environment C++ Builder. Among the different categories of applications of mediation systems include applications of information retrieval on the Web, those of decision support online, more generally, knowledge management in the broad sense [1]. We present the following case study to demonstrate the operation of the algorithm. We have the global schema and source schemas S1, S2 et S3 represent databases «Department, Employers». The global schema is as follows: Global Schema:

- Dept(DeptKey, Dnom, Budget);
- Emp(EmpClé, Enom, DeptKeyEtr, Salaried);

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

Fig. 2. the relation between different schemas

Sources schema are:

- Local schema of the source S1:
  Department(DeptKey, Dname, Bdg);
- Local schema of the source S2:
  Depart(DeptKey, DN, Budg);
- Local schema of the source S3:
  Employ(EmployKey, Enom, DKeyEtr, wages);

The relation between different schemas is presented in the figure fig 2. In our case study we used the subset of GLAV mapping rules represented in the table 1.

For example, either a user query $Q$:

```
Q = for $x$ in collection("GlobalSchema")/Dept, $y$ in collection("GlobalSchema")/Emp
    Where $x/DeptKey= y/DeptKeyEtr$ and $x/Dnom= \"department1\"$ and $y/Salaried= \"20000,00 DA\"
    return [ Name = $y/Enom]
```

The user wants to know the name of the employers in the 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 and $q_{g_1}$ and $q_{g_2}$.
TABLE 1. MAPPING RULES

<table>
<thead>
<tr>
<th>$q_{g_1}$</th>
<th>for $z$ in collection(&quot;GlobalSchema&quot;)/Dept where $z$/Dnom= $a$ return $z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1$</td>
<td>for $z$ in ( for $t$ in collection(&quot;LocalSchema-Source1&quot;)/Department where return [DeptKey=$t$/ DepartmentKey, Dnom=$t$/Dname, Budget=$t$/bdg ] ) where $z$/Dnom= $a$ return $z$ union</td>
</tr>
<tr>
<td>$q_{S_1}$</td>
<td>for $z$ in ( for $t$ in collection(&quot;LocalSchema-Source2&quot;)/Dept where return [DeptKey=$t$/ DeparKey, Dnom=$t$/DN , Budget=$t$/budg ] ) where $z$/Dnom= $a$ return $z$</td>
</tr>
<tr>
<td>$q_{g_2}$</td>
<td>for $z$ in collection(&quot;GlobalSchema&quot;)/Emp where $z$/Salaried= $a$ return [ A1= $z$/Enom, A2= $z$/DeptKeyEtr ]</td>
</tr>
<tr>
<td>$r_2$</td>
<td>for $z$ in collection(&quot;LocalSchema-Source3&quot;)/Employ where $z$/wages= $a$ return [ A1= $z$/Ename, A2= $z$/DKeyEtr ]</td>
</tr>
<tr>
<td>$q_{S_2}$</td>
<td>for $x$ in collection(&quot;LocalSchema-source1&quot;) where $x$/Salaried= $a$ return $x$.</td>
</tr>
</tbody>
</table>

For example, either a user query Q:

$Q = \{ \text{for } x \text{ in collection("GlobalSchema")/Dept, } y \text{ in collection("GlobalSchema")/Emp} \text{ where } x/\text{DeptKey}= a, y/\text{Salaried}= 20000,00 \text{ DA and } x/Dnom= \text{"department1"); }$ return $[ \text{Name }= y/\text{Enom}]$ |

The user wants to know the name of the employers in the 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 and $q_{g_1}$ and $q_{g_2}$ and $q_{S_1}$ and $q_{S_2}$.

$Q = \text{for } x_0 \text{ in } q_{g_1} \theta_1, \text{for } x_01 \text{ in } q_{g_2} \theta_2$, where $x_0/\text{DeptClé}= x_01/A2$ return $[ \text{Name }= x_01/A1]$ 

where: $\theta = \{ z/\text{x}, x/a/\text{department1} \}$ and $\theta = \{ z/\text{x}, x/a/20000,00 \text{DA} \}$

- The identification of sources involved in the execution of the query Q: $q_{g_1}$ corresponds $q_{S_1}$ (so the sources A1 and A2 are involved in the execution of Q) and $q_{g_2}$ corresponds $q_{S_2}$ (so the source A3 participate in the execution of Q).

<table>
<thead>
<tr>
<th>Sub query</th>
<th></th>
<th>Sent to</th>
</tr>
</thead>
<tbody>
<tr>
<td>for $x$ in ( for $t$ in collection(&quot;LocalSchema-source1&quot;)/Department where return [DeptKey=$t$/ DepartmentKey, Dnom=$t$/Dname , Budget=$t$/bdg ] ) where $x$/Dnom= “department1”</td>
<td>$S_1$</td>
<td></td>
</tr>
<tr>
<td>for $x$ in ( for $t$ in collection(&quot;LocalSchema-source2&quot;)/Depart where return [DeptKey=$t$/ DeparKey, Dnom=$t$/DN , Budget=$t$/budg ] ) where $x$/Dnom= “department1”</td>
<td>$S_2$</td>
<td></td>
</tr>
<tr>
<td>for $x$ in collection(&quot;LocalSchema-source3&quot;)/ Employ where $x$/wages= “20000,00DA” return [ A1= $x$/Ename, A2= $x$/DCléEtr ]</td>
<td>$S_3$</td>
<td></td>
</tr>
</tbody>
</table>

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 the algorithm 2, the sub-queries that will be executed at the sources (table 2). Finally the mediator recomposes the results.

Figures 3, 4 and 5 present the results of our system: The Figure 3 presents the compilation of a user query who’s been passed successfully and it displays the steps of lexical and syntactic verification. the figure 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 of complex type because the algorithm could not find a direct reformulation for Q.

Figure 5 also shows the reformulated query with sub queries to be sent to data sources.

The quality of reformulation depends on a manner diecte by the mappings rules, more than the mappings rules are numerous more than the algorithm takes time to find reformulations.

TABLE 2. THE SUB QUERIES SENT TO S1, S2 AND S3
VI. CONCLUSION

A mediation system is a powerful mean allowing an easy access to various information collected from data sources 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 an algorithm reformulation of XQuery queries for mediation systems using GLAV mappins and unification.

The implementation of the prototype mediation system, illustrates the operation of the algorithm. We advise 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 shorts reformulations. As a prospect our mediation system will be improved by taking into account the following points: The use of all possibilities of XQuery language, Building a module that lets to add or remove rules mapping, building adapters to resolve structural conflicts of heterogeneous sources (XML schema model, relational model ...).

REFERENCES

[4] Rashed Mustafa, Hasan Hafizur Rahman, Mediator based architecture to address data heterogeneity, daffodil international university journal of science and technology, volume 8, issue 2, july 2013
Dr. Benharzallah Saber  Received the Baccalaureate diploma in 1997, engineering in computer sciences from the university of  Batna in 2002 and the Master in computer sciences from the university of Mentouri Constantine in 2005. He obtained his PhD degree from the  Biskra university in 2010. He is the leader of the research team named Semantic Interoperability and its applications to Web Services and Information Systems. Since 2006 he is an assistant professor at computer sciences department, university of Mohamed Kheider Biskra , Algeria. His main research concern data and services integration, cooperative information systems, multi agents systems and semantic web.

Pr. Okba Kazar  was born at Biskra (Algeria) in 1962. He obtain his master degree in 1997 from the Constantine University (Algeria) working on artificial intelligence field. He obtained his PhD degree from the same university in 2005. He is member of editorial board of the SJI Journal. He is an author of some publication in international journals and session chair in international conferences. He is the chief of Smart Computing Laboratory. He is interested to the multi-agents systems and their applications, advanced information systems, Web services, semantic Web. Pr. Okba KAZAR actually is an associate professor at computer science department of Biskra university. www.e-monsite.com/kazarokba.