A solution for XQuery queries rewriting in LAV approach in a mediation system

Ahmed Zellou[1], Dalila Chiadmi[2]

Ecole Mohammadia d'Ingénieurs
Department of Computer Science
BP765, Rabat Agdal

MOROCCO

[1] zellou@emi.ac.ma, [2] chiadmi@emi.ac.ma

RÉSUMÉ. La médiation d'information suscite aujourd'hui un intérêt particulier en vue de son application dans de nombreux domaines (télé-enseignement, bibliothèques électroniques, etc.). La médiation vise à intégrer un ensemble de sources d'information autonomes, hétérogènes, réparties et évolutives via la définition d'un schéma global qui représente le contenu des sources. Deux approches (Global As View - GAV et Local As View - LAV) sont couramment utilisées pour définir la relation de liaison entre le schéma global et le contenu des sources. Dans l'approche GAV, la réécriture est simple. En revanche, dans l'approche LAV, la réécriture est difficile et nécessite le recours à des algorithmes. Plusieurs algorithmes ont été proposés dans le modèle relationnel (Bucket, règles d'inversions, MiniCon, etc.). Nous proposons dans cet article une adaptation de l'algorithme Bucket pour la réécriture des requêtes XQuery dans le modèle semi-structuré selon l'approche LAV.

ABSTRACT. The information mediation attracts today a particular interest because its application in many fields (e-learning, electronic libraries, etc.). The mediation allows integrating a set of autonomous, heterogeneous and distributed information sources through the definition of a global schema which represents the sources content. There are two main approaches (Global As View - GAV and Local As View - LAV) that allow defining the mapping between the global schema and the sources content. In the GAV approach, rewriting is simple. However, in the LAV approach, the rewriting is difficult and requires the use of algorithms. Several algorithms have been proposed in the relational data model (Bucket, inverse rules, MiniCon, etc.). We propose in this paper an adaptation of the Bucket algorithm for the XQuery query rewriting in semi-structured data model according to the LAV approach.

MOTS-CLÉS : médiation, réécriture, LAV, Bucket.

KEYWORDS: mediation, rewriting, LAV, Bucket.

1. Introduction

The importance of query rewriting is not to demonstrate due to its use in numerous application fields: mediation information, data warehouses, design websites, etc [1, 2]. In the information mediation context, users do not formulate their queries in terms of the schema within the data is stored. The queries are formulated in terms of a global schema (mediation schema). However, the data is stored in sources using other schemas (sources schemas). Accordingly, the mediation system must rewrite users queries into queries formulated in terms of sources schemas.

The information mediation concept dating back to 70 years, Levy defines mediation as a transparent access service to a huge number of autonomous, heterogeneous, dynamics and distributed information sources [2]. We define the information mediation as an intermediate tool between a user or application, and a set of information sources; this tool provides a transparent access to sources by a unique interface and query language.

By providing to the user a unified global view on the information sources, the role of the user is limited to put its query to the mediation system. The mediation system interrogates thereafter all sources before returning the result to the user. The query is done in a transparent manner to the user who has the illusion of facing a simple and unique information source which is the mediation system.

Since the data is stored really in sources, the mediation system must have all sources descriptions in order to rewrite the user's queries in terms of sources schemas. These descriptions provide a mapping between relations in sources schemas and global schema. We refer by schemas integration, the construction process of a unified global schema that establishes a connection between the different sources of a mediation system. This schema presents to the users a global, uniform and centralized view on the data distributed in sources, it's the only information available to the user.

This paper is structured into six parts. After this introduction, the second part presents approaches to link schemas in a mediation system as well as the queries rewriting algorithms. We present in the third part the pretreatment needed to rewrite queries for our solution. The fourth part presents the solution that we propose for XQuery queries rewriting in LAV through the adaptation of the Bucket algorithm and a prototype developed for this purpose. We present thereafter at the fifth part the limits of our approach and we finish with a conclusion and the prospects for this work.

2. The schemas mapping approaches

The mapping specification between schemas will determine the difficulty of queries rewriting, as well as the ease of adding or deleting sources within the mediation system.

Two methods are commonly used to determine this mapping ; either the global schema is considered as view on the local sources (*GAV - Global As View*) [3, 4, 5, 6], or, conversely, local sources are considered as views on the global schema (*LAV - Local As View*) [7, 8, 9, 10, 11]. We cite also the *GLAV* [12] approach which applies in the case of necessity to put query recursive on sources, as well as the *BAV* approach (*Both As View*) [13], which adapts to the peer-to-peer mediation environments.

In the GAV approach, the global schema is defined in terms of local sources schemas. The first step to implement this approach is to define a global schema which covers data involving users. Thereafter, for each relation in global schema, a query specifying how to obtain data of this relation from sources schemas is defined. However, the LAV approach goes in the opposite direction. After defining the global schema, the content of each source is described as a view of the global schema.

In the GAV approach, the query rewriting is much simpler and straightforward, just browsing the links between the relations in global schema and theses in sources schemas to rewrite the query. However, the addition or deletion of one or more sources in the system requires redefining the global schema. In particular, the addition of a new source requires redefining the relations of global schema that can extract data from the new source.

On the other hand, knowing that each source is described separately in the LAV approach, the addition or deletion of a source is not a problem. We have just to describe the new source in the form of view (query) on the global schema. In contrast, query rewriting in the LAV approach is more difficult than in the GAV approach because each source is described separately.

The two approaches are not in contradiction, but complementary, the use depends on the mediation context. To integrate few and static sources, it is better to use the GAV method. However, in the large-scale integration context, the LAV method is preferable because a major change at a local source will have no impact on the global schema.

Knowing that the rewriting of queries is difficult in the LAV approach, solutions are needed for subqueries generation in this context. Rewrite the user's query, in this case, is to rewrite this query using a set of views. Three queries rewriting algorithms in terms of views have been developed in the information mediation context. The first algorithm named (the Bucket algorithm) has been implemented under the Information Manifold mediation system [14]. In this algorithm, queries, views and rewriting produced are conjunctive queries with comparison predicates. The second algorithm named (the inverse rules algorithm) has been implemented under the InfoMaster mediation system [15]. The accepted views and rewriting produced are conjunctive queries without comparison predicates. The last algorithm named (the MiniCon algorithm) results of a combination of the two previous algorithms [16]. In this case, the queries and views are conjunctive queries with arithmetic comparison predicates. The three algorithms are

developed in a specific context in which the query languages used in the mediation systems are relational.

Several systems use the GAV approach for schema mapping (InfoSleuth, LeSelect, TSIMMIS, TAMBIS, MOMIS, Disco, InterMed, DiscoveryLink, XMF and Xperanto) [3, 17, 18, 19, 20]. In contrast, the systems Manifold, Nimble, Agora and E-XMLMedia use the LAV approach [21, 22, 23]. In conclusion, few systems adopt the LAV approach for schema mapping. Agora uses the query language quilt. Thus, E-XMLMedia uses the query language Xpath and describes the mediation metadata as a DTD. We believe that XPath and quilt are less wealthy than the XQuery to express queries on semi-structured data. Thus, presenting the mediation metadata as DTD does not provides enough mechanisms to describe this kind of information. The purpose of this paper is to propose a solution for rewriting XQuery queries in the LAV approach with a mediation metadata description as XMLSchema.

3. Rewriting Pretreatment

We present now our contribution, namely : a solution for XQuery queries rewriting in the LAV approach. Formally, if *XSG* is the global schema of the mediator and if XSL_i are local schemas of sources S_i $(1 \leq i \leq n)$, then according to the LAV approach, $XSL_i = Vue_i\ (XSG)\ (1 \leq i \leq n)$. In order to define the mapping between the global schema *XSG* and local schemas XSL_i, we propose to associate to each source S_i, a metadata XSL_i in XMLSchema with XSL_i is a subset of *XSG*.

The rewriting requires a set of information, namely the global schema expressed in XMLSchema, local schemas of sources also expressed in XMLSchema and the mapping between the global schema and local schemas of sources. This information, once received and stored in a catalog, requiring a pre-treatment for our rewriting solution.

After defining XSG and XSL schemas in XMLSchema, a set of plans is derived from these schemas that allows : identifying each element or attribute in the XSG and XSL schemas as well as the access path to it in the catalog, establishing the mapping between the elements and attributes of the local schema and those of the global schema and finally associating with each local schema a key (KIFs) aims to specify the relation between the XML fragments after execution.

4. The XQuery queries rewriting in LAV

We present in this part our solution of XQuery queries rewriting in LAV. The solution we are proposing is an adaptation of the *Bucket* algorithm in semi-structured data model.

The query canonization

The user formulates his query in terms of XSG schema using the XQuery query language. The first phase, called canonization, consists of removing in the user query the set of representation tags.

The query atomization

Each user query contains two types of variables, variables that are in the condition "*where*" (we call them conditional variables) and variables that are in the return block "*return*" (we call them effective variables). The atomization consists to decompose the query into a set of subqueries (named atomic subqueries), each concerns one variable. The atomization is done by determining the two types of variables on which the query find. Thereafter, we generate a subquery for each atomic variable, effective or conditional, in the user query (named respectively effective atomic subquery (denoted AQ_Effe_i) and conditional atomic subquery (denoted AQ_Cond_i)).

The identification

For each effective atomic subquery AQ_Effe_i, the return expression $Expression_{2i}(X_i)$ can be reduced to an expression like $path_{2i}/var_i$ $(1 \leq i \leq n)$. The next step is to identify the effective variables var_i in the atomic subqueries as well as the sources where atomic subqueries should be sent from plans. Thereafter, we fetch from the catalog the elements and attributes $idelement_{i,j}$ are equivalent to each variable var_i in local schemas. In this context, an atomic subquery can be rewritten into several subqueries, each covering a variable in a local schema. The next step is to determine the access path to each variable $idelement_{i,j}$ in the local schema of source $idsource_{i,j}$, from the root, named $path_{i,j}$.

On the other hand, for each conditional atomic sub-query AQ_Cond_k, the expression $Condition_k(\{X_j\})$ $(1 \leq k \leq m,\ 1 \leq j \leq n)$ can be rewritten into an expression like $(\{path_j/var_j\ op\ const\}\ Op_j)$ $(1 \leq j \leq m)$ where *op* is a comparison operator $op \in \{<, \leq, \geq, >, ==, ! =, contain\}$, Op_j is a link operator $Op_j \in \{and, or, !, xor\}$ with $Op_m = NULL$ and const is a constant. Thereafter, we treat the variables in the conditional atomic subqueries. The objective is to identify the sources involved in constraint definition in each conditional atomic subquery definition. The processing of conditional atomic subqueries variables carried out in three steps. In the first, we identify conditional variables var_j in the atomic query $AQ\_Cond_i : \{var_j\ op\ const\}$. In the second step we determine the equivalent elements and/or attributes to each conditional variable in the local schemas as well as their sources and in the third step we determine the access path for each variable from the root source.

Adaptation of the Bucket algorithm

The solution that we propose is an adaptation of the Bucket algorithm. For each effective atomic subquery, a green bucket is constructed in which we put the rewriting result of the subquery. In some way, for each conditional atomic subquery, a red bucket is constructed in which we put the rewriting result of the subquery. Moreover, a bucket blue is constructed in which we put a set of subqueries allowing us to retrieve the keys

(KIFs) from all sources. Each subquery, named identification subquery (denoted AQ_KIF_i), allows extract the KIF key from only one source.

The generation of the execution plan

An execution plan is a tree structure incorporating at leaves level the atomic subqueries and at nodes level the operators (joints, restriction, projection, etc.) as well as the function operators. At the end of the preceding step, we get a set of atomic subqueries : effective, conditional and fragments identification.

Each bucket can contain several rewriting of the atomic subquery in which it is associated. Thereafter, we choose a single rewriting by bucket. The choice can be justified by optimization reasons (the quickest source, the most efficient source, etc.).

Suppose that a rewriting R_E_j $(1 \leq j \leq n_i)$ was chosen for the effective atomic subqueries EAQ_i $(1 \leq i \leq n)$. Suppose also that a rewriting R_C_k $(1 \leq k \leq m_i)$ is chosen for the conditional atomic subqueries Rg_C_i $(1 \leq i \leq m)$. The execution plan is generated in three steps :

– Step 1 : determine for each rewriting R_E_i (respectively R_C_i) for an effective atomic subquery R_E_i (respectively R_C_i), which retrieves results from a source S_i, the fragments identification query AQ_KIF_i of the source S_i (available in the blue bucket). Thereafter, we merge each rewriting with its fragments identification query AQ_KIF_i.

– Step 2 : with an inspection of sources for each atomic subquery, we can combine those of the same source. The combination of two effective atomic sub-queries R_E_i and R_E_j (res. R_C_i and R_C_j) will generate an effective atomic sub-query denoted $R\_E_{(i/j)}$ (res. R_C_{Si}).

– Step 3 : merge the all new effective rewriting R_E_i $(1 \leq i \leq n)$ and the all new conditional rewriting R_C_k $(1 \leq k \leq m_i)$ using the operators Op_k $(1 \leq k \leq m_i)$ in a single execution plan.

Prototype

We have developed a prototype of our solution for Xquery queries rewriting in LAV approach by adopting the Bucket algorithm. This prototype is based on PHP, Mysql, javascript and XML technologies. We present below a screen that presents the atomic subqueries generated for an example query.

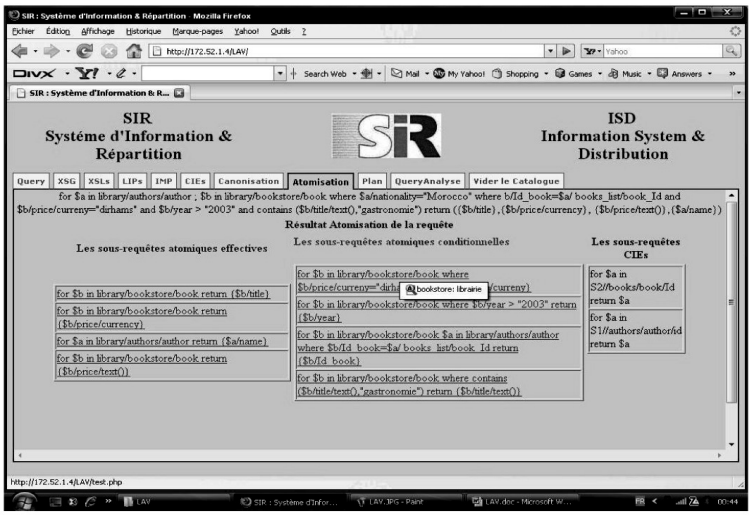

Figure 1. *The query atomization.*

5. Our approach limitations

The solution that we have presented is an adaptation of the Bucket algorithm for XQuery queries rewriting in LAV and in a semi-structured data model. We faced two major problems in the semi-structured data model :

– in the relational model, for which the Bucket algorithm was proposed, the user query is in advance divided into several relations, which facilitates the association of buckets to the query relations. In XQuery, the query is not decomposed, and that is why we generate the atomic subqueries.

– in the original Bucket algorithm, the views to put in a bucket are calculated and materialized in advance. In our case, the views are not calculated in advance (except the KIFs views) and that is why we rewrite each atomic subquery before putting the result rewriting in buckets.

Our solution can be used in the majority of cases, in spite of some limits related to the nature of the query and the choice of rewriting. On the nature aspect of the query, our solution supports the selection, projection and joint queries. For the selection queries, our solution supports queries whose selection criteria is defined as *{var op var/const} Op* where *var* is a variable, *const* is a constant, *op* is a comparison operator *(<, ≤, ≥, >,==,! =, contain)*, and *Op* is a link operator *(and, or,!, xor)*. But our solution is not running for queries finding some functional expressions as *sum*, *avg*, co*u*nt, *min* and *max*. In the other hand, our solution assumes that a choice is to do at each bucket one choice only among several possible rewritings thus enabling our solution to use an

optimization strategy. For lack of choice, our algorithm remains blocked at this level. We believe that it's possible to join all rewritings of a bucket to form a single rewriting thus avoiding the selection process. Once adopted, this solution would deprive our algorithm of its efficiency to choose a single rewriting by bucket.

The keystone of a mediation system is its global schema. A rich and optimal schema remains a heavy task for the catalog administrator. A light schema facilitates queries processing, but does not deal all sources characteristics and does not allow the addition of certain sources in the system. A rich schema, however, complicates queries definition. Managing keys in this case must be done carefully to return consistent results.

6. Conclusion and future works

We have proposed in this paper a solution for the XQuery query rewriting on the LAV approach. Given the fact that queries rewriting can be done only according to the information stored in the catalog. We declined our solution into two basic constituents. The first is how we manage the information (required for rewriting) stored in the catalog through defining a set of plans and. The second is how we rewrite queries through an adaptation of the Bucket algorithm.

As future works, we will extend our solution to support the different types of XQuery queries (min, max, count, etc.). We will also proposing a solution to support applications that require recursive queries on the sources.

7. References

[1] Xiaolei, Q., "Query folding". In Proc. *of Int. Conf. on Data Engineering (ICDE)*, New Orleans, LA, 1996, PP. 48-55.

[2] Levy, A. Y., "Logic-based techniques in data integration", *Logic Based Artificial Intelligence*, 2000.

[3] Garcia-Molina, and al., "The TSIMMIS project: Integration of heterogeneous information sources". *Journal of Intelligent Information Systems*, 1997, pp. 8(2):117-132.

[4] Papakonstantinou, Y., and al., "Object fusion in mediator systems". *In Proceedings of International Conference on Very Large Databases (VLDB)*, Bombay, India, September 1996, pp. 413-424.

[5] Adali, S., and al., "Query Caching and Optimization in Distributed Mediator Systems". *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, ACM, New York, June 1996, pp. 137–148.

[6] Florescu, D., and al., "Answering queries using OQL view expressions". *In Workshop on Materialized Views, in cooperation with ACM SIGMOD*, Montreal, Canada. 1996.

[7] Levy, A. Y., "The Information Manifold Approach to Data Integration", *IEEE Intelligent Systems*, 1312-16, 1998.

[8] Kwok, C. T. and Weld, D. S., “Planning to gather information". *In Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence*. 1996.

[9] Duschka, O. M., and Genesereth, M. R., “Answering recursive queries using views". *In Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, Tucson, Arizona. 1997.

[10] Friedman, M., and Weld, D. “Efficient execution of information gathering plans". *In Proceedings of the International Joint Conference on Artificial Intelligence*, Nagoya, Japan. 1997.

[11] Ives, Z., and al., ““An adaptive query execution engine for data integration". *In Proc. of ACM SIGMOD Conf. on Management of Data*. 1999.

[12] Friedman, M., and al., “Navigational plans for data integration". *In Proceedings of the National Conference on Artificial Intelligence*. 1999.

[13] Boyd, M., and al. “AutoMed: A BAV Data Integration System for Heterogeneous Data Sources". *In Advanced Information Systems Engineering 16th International Conference*, CAiSE, Riga, Latvia, June 7-11, Proceedings .Springer-Verlag, 2004.

[14] Levy, A. Y., and al., “Querying heterogeneous information sources using source descriptions". *In Proc. Of the Int. Conf. on VLDB*, Bombay, India. 1996.

[15] Duschka, O. M., and Genesereth, M. R., “Query planning in infomaster". *In Proceedings of the ACM Symposium on Applied Computing*, San Jose, CA. 1997.

[16] Pottinger, R., Halevy, A. Y., "MiniCon: A scalable algorithm for answering queries using views". *VLDB Journal*, 10(2-3):182--198, 2001.

[17] Bayardo, R. J., and al., “InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments", *ACM SIGMOD*, 1997.

[18] Beneventano, D., and al., “The MOMIS approach to information integration”. *In AAAI International Conference on Enterprise Information Systems*, ICEIS 2001.

[19] Haas, L., and al., “DiscoveryLink : A System for Integrating Life Sciences Data", *IBM Systems Journal*, vol. 2, 2001.n, 40.

[20] Kangchan, L., and al., “A Design and Implementation of XML-based mediation Framework (XMF) for Integration of Internet Information Resources". *In Proceedings of the 35th Hawaii International Conference on System Sciences*, 2002.

[21] Draper, D., and al., “The Nimble Integration Engine". *In SIGMOD Record (ACM Special Interest Group on Management of Data)*, volume 30, 2001.

[22] Manolescu, I., and al., “Agora: Living with XML and relational". *In Proc. of the VLDB Conf. Software demonstration*. 2000.

[23] Dang-Ngoc, T. T., and Gardarin, G., “The XML Mediator", Document technique interne à e-XMLMedia, 26 pages, 2002.