Managing the Evolution of Mediation Queries

Mokrane Bouzeghoub[1], Bernadette Farias Lóscio[2], Zoubida Kedad[1], and Ana Carolina Salgado[2]

[1] Laboratoire PRiSM, Université de Versailles
45, avenue des Etats-Unis 78035 Versailles, France
{Zoubida.Kedad, Mokrane.Bouzeghoub}@prism.uvsq.fr
[2] Centro de Informática - Universidade Federal de Pernambuco
Av. Professor Luis Freire s/n, Cidade Universitária
50740-540 Recife – PE, Brasil
{bfl, acs}@cin.ufpe.br

Abstract. Previous works in data integration can be classified according to the approach used to define objects at the mediation level. One of these approaches is called global-as-view (GAV) and requires that each object is expressed as a view (a mediation query) on the data sources. One important limit of this approach is the management of the evolutions in the system. Indeed, each time a change occurs at the source schema level, all the queries defining the mediation objects have to be reconsidered and possibly redefined. In this paper, we propose an approach to cope with the evolution of mediation queries. Our claim is that if the definition of mediation queries in a GAV context follows a well-defined methodology, handling the evolution of the system becomes easier. These evolution problems are considered in the context of a methodology we have previously defined for generating mediation queries. Our solution is based on the concept of relevant relations on which propagation rules have been defined.

1 Introduction

The goal of data integration consists in providing a uniform view of data sources (called mediation schema or global schema) and defining a set of queries (called mediation queries or mediation mappings) which define objects of the mediation schema. These queries will later serve to rewrite users' queries prior to their execution. A mediator is a software device that supports a mediation schema [15], i.e., a collection of views over the data sources reflecting users' requirements. Beside the set of mediation queries which define the mediation schema, a set of linguistic mappings specifies correspondences between the mediation schema elements and the local schemas ones.

Previous work in data integration can be classified according to the approach used to define the mediation queries [4, 7, 14]. The first approach, called global-as-view (GAV), requires that each relation (or class) of the global schema be expressed as a view (i.e. a query) on the data sources. In the second approach, called local-as-view (LAV), mediation queries are defined in an opposite way; each relation (or class) in a given source is defined as a view on the global schema. The GAV approach is known as easy to implement but difficult to evolve; each change raised at a source schema may lead to the redefinition of possibly all the mediation queries. The LAV approach has the opposite feature; each source evolution results into the redefinition of the only mediation queries corresponding to this source, but the implementation of this approach is harder as the rewriting process of user queries is more complex.

This paper addresses the problem of evolution in the context of the GAV approach. As we know, local data sources are often autonomous and may change both in their structures and their concepts. New sources may be added to the system and other sources may be removed from the system either because of their irrelevance or because of their unavailability. We are interested in providing mechanisms to correctly update the mappings between the mediation schema and the distributed sources after source schema changes. Few research have discussed some aspects related to this problem [1, 8, 9]. We will compare our approach to these related works in section 5.

The challenge is to maintain the mediation queries consistent with source evolution. We claim that if the mediation queries are derived following a certain design methodology, the problem of their evolution will be easier to solve. Indeed, prior to the evolution problem, the definition of mediation queries is also a hard problem in the GAV approach, especially in the context of large scale systems. Defining a mediation relation over hundreds or thousands of source schemas is a very hard task, regarding the amount of metadata necessary to determine the queries. In fact, the evolution problem of the GAV approach is also related to the scalability of mediation systems. Evolution of a small-size mediation system is not as crucial as that of a large-size mediation system.

In a previous work, we have proposed a design methodology which generates mediation queries in the context of very large mediation systems based on the relational model [5]. Given a mediation relation, a set of source schemas and a set of linguistic assertions between the mediation schema and the sources schemas, we have defined an algorithm which discovers the mediation queries defining this relation. The evolution process is seen as an incremental execution of this algorithm.

This paper is organized as follows. Section 2 defines the evolution problem and the main notations. Section 3 recalls the principles of our design methodology for determining the mediation queries. Section 4 is devoted to the propagation of source changes to the mediation level using this methodology. Section 5 presents the related works and finally, section 6 concludes with some open problems.

2 Problem Statement

In mediation systems, the evolution problem is mainly related to changes raised at the data source level: adding or removing a relation schema, an attribute or a constraint. The mediation schema itself is supposed to be relatively stable, that is, not subject to intensive changes. Moreover, advantages and drawbacks of GAV and LAV approaches are given with respect to this assumption. Handling source evolution is an essential feature as it implies modularity and scalability of the mediation system. As mentioned before, the GAV approach suffers from the lack of evolution because each change at the source schema level may lead to the reconsideration and possibly the change of all mediation queries. One way to cope with this problem is to isolate only those mediation queries which are impacted by the source change and to automatically redefine these queries by applying only the necessary changes.

In a mediation architecture (as shown in figure 1), a mediation schema represents the reconciliation between users' requirements and the sources capabilities. In other words, if we assume that the mediation schema is expressed using the relational model, it contains all the relations needed in a specific business domain to answer users' queries which can be computed from the data sources. In the GAV approach, each relation R_i in the mediation schema is defined by a mediation query Q_i which computes the relation R_i over a subset of data sources.

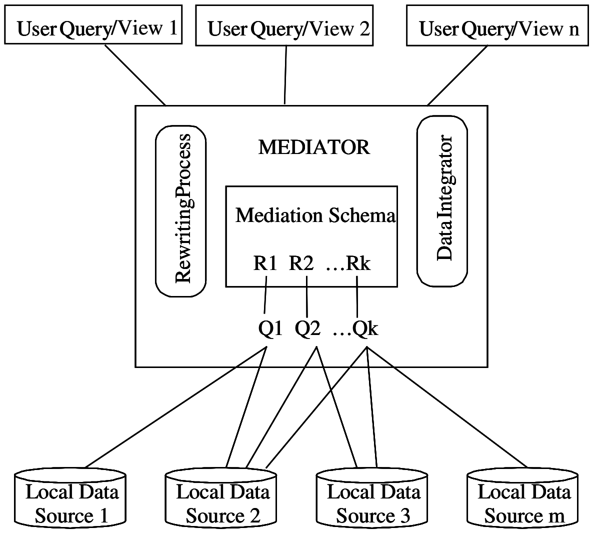

Fig. 1. A mediation architecture

To illustrate the problem we aim to solve, consider the example of two mediation relations R_1 and R_2 having the following schemas: R_1(#K, A, B) and R_2(#K', C, D). These relations are defined over the following source relations: S_1(#K, A, @X), S_2(#X, B), S_3(#K', C, D), S_4(#K', C, D). Primary key attributes are prefixed by # and foreign key attributes are prefixed by @. S_1 and S_2 belong to the same data source 1. S_3 and S_4 belong respectively to data sources 2 and 3. The mediation queries associated with the mediation relations R_1 and R_2 are the following:

$R_1 = \Pi_{K, A, B}(S_1 \bowtie_{S1.X=S2.X} S_2)$ and $R_2 = S_3 \cup S_4$.

Suppose some changes occur at the source level, consisting in removing the source relations S_1 and S_4. Our goal is to propose a methodology allowing to propagate the changes occurring at the source level to the mediation queries. As a consequence of these changes, mediation queries have to be checked, and if possible, to be rewritten. In our example, after the source changes, there is no way to compute the relation R_1 and the associated mediation query becomes invalid. The relation R_2 is still computable, and the new mediation query is $R_2 = S_3$.

The key issues of this process are first, how to propagate a given source change into a single mediation query, which is done through a set of evolution rules; second, how to define the global evolution process, which will consists in propagating a set of changes issued by several data sources to the set of mediation queries defining relations of the mediation schema.

In the following sections, we assume the relational model being the common model for both the local schemas and the mediation schema. We use the following notations to represent relations and attributes.

- S_{ij} denotes a relation i in the data source j.
- R_m denotes a relation of the mediation schema.
- $S_{ij}.A_k$ denotes the attribute A_k of the source relation S_{ij}; similarly, $R_m.A_k$ denotes the attribute A_k of the relation R_m in the mediation schema.

We also assume that some meta data describing the sources and the mediation schema is available. We will use the following definitions:

- Referential constraints: they are defined within a single schema, either the mediation schema or a local source schema; they are denoted $Ri.A \rightarrow Rj.K$.
- Semantic equivalence between attributes: when two attributes A_1 and A_2 represent two equivalent concepts, they are said to be semantically equivalent; this link is denoted $A_1 \cong A_2$ (or $R_i.A_1 \cong R_i.A_1$ to avoid confusions).
- Schema assertions, which are one of the following:

  $\underline{R}_1 = \underline{R}_2$ if the schema of the relation R_1 corresponds to the schema of the relation R_2 through a 1:1 mapping such that each attribute of R_1 has a semantically equivalent attribute in R_2 and each attribute of R_2 has a semantically equivalent attribute in R_1.

  $\underline{R}_1 \subset \underline{R}_2$ if each attribute of the relation R_1 has a semantically equivalent attribute in the relation R_2.

  $\underline{R}_1 \cap \underline{R}_2 \neq \emptyset$ if some attributes of R_1 have semantically equivalent attributes in R_2.

  $\underline{R}_1 \cap \underline{R}_2 = \emptyset$ if there is no attribute in R_1 having a semantically equivalent attribute in R_2.

In the remaining of this paper, equality, union and intersection operations stand for schema assertions when relation symbols are underlined; they stand for regular relational algebra when the relation symbols are not.

3 Generation of Mediation Queries

One of the difficult problems met when designing mediation-based systems is the definition of mediation queries. Indeed, defining queries over tens or hundreds of heterogeneous data sources requires a complete and perfect understanding of the semantics of these sources. In [5], we have proposed an approach which discovers mediation queries over a set of heterogeneous sources in a GAV context. This section recalls, through a simple example, the general principle of the Mediation Query Generation approach (MQG), which will serve as a support to propagate changes from the source level to the mediation level.

3.1 Intuitive Approach

The MQG algorithm can be roughly summarized by three steps: (i) search of contributive sources, (ii) determination of candidate operations, (iii) definition of queries. These steps are illustrated through the following example. Let R_m(#K,A,B,C) be a mediation relation and $\{S_1,S_2,S_3,S_4\}$ be a set of source relations over which R_m will be defined. S_1(#K,A,@X,Y) and S_2(#X,B,Z) are in source 1, S_3(#B,C,W) is in source 2 and S_4(#B,C,U) is in source 3. Primary key attributes are prefixed by # and foreign key attributes are prefixed by @.

- *Step 1. Search for contributive sources:* The first step consists in finding all possible sources which may contribute to the computation of R_m. Intuitively, a source relation S_{ij} contributes to the computation of a mediation relation R_m if S_{ij} includes some of the attributes of R_m. The notion of *mapping relation* T_{mij} is introduced to group all common attributes between R_m and S_{ij}. To later facilitate the discovery of operations, the definition of a mapping relation is extended with primary keys and foreign keys of its source relation S_{ij}. Figure 2 shows the mapping relations T_1, T_2, T_3, and T_4 derived from source relations S_1, S_2, S_3 and S_4 to compute the mediation relation R_m.

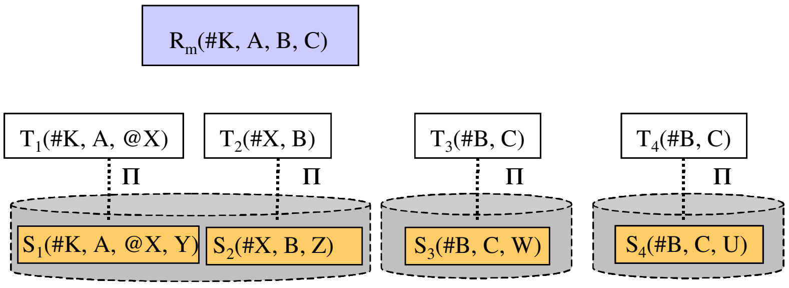

Fig. 2. Example of mapping relations

- *Step 2. Determination of candidate operations:* Given the set of mapping relations $\{T_1,T_2,T_3,T_4\}$ associated with the mediation relation R_m, the determination of candidate operations depends on operation type (join, union, intersection, difference), relation schemas and links between relations. To illustrate this, we will consider the join and union operations.

  - *The join operation is candidate in two cases:* (i) the two mapping relations are originated from the same source; in this case we consider that a join is possible only if there is an explicit referential constraint between the two source relations; (ii) the two mapping relations are originated from different sources; in this case, we consider that a join is possible if the primary key of one relation has an equivalent attribute in the other relation, either a non-key attribute or a key attribute. We consider the case of a non-key attribute as a kind of implicit referential constraint between the two source relations.

- o *The union operation is candidate if the two mapping relations have the same schema.* We can extend this rule to any intermediate relation obtained by combination of mapping relations.

Figure 3 shows some possible operations between mapping relations of figure 2. The graph of all possible operations is called the operations graph.

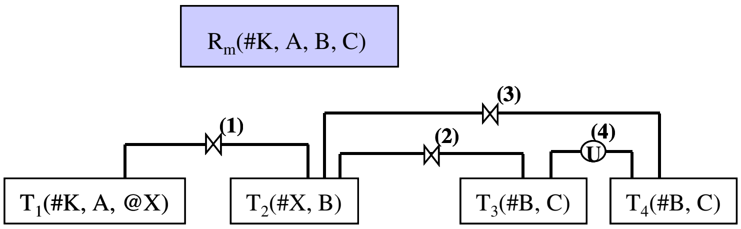

Fig. 3. Example of operations graph

One should remark that, in the case of a join operation, there is not always necessarily an explicit or implicit referential constraint between two source relations. It might be possible to join two source relations S_i and S_j through a third relation S_k which is not directly contributive to the computation of R_m. Our algorithm includes these relations as *transition relations* which contain only necessary primary keys and foreign keys which make possible a join between mapping relations. Obviously, there may exist more than one transition relation between two mapping relations.

- *Step 3. Definition of mediation queries:* Having the operations graph defined between relevant relations (mapping relations and transition relations), it becomes easier to generate mediation queries. For that purpose, we introduce the concept of *computation path*, which is a mapping relation involving all the attributes of R_m, or any connected and acyclic sub-graph of the operations graph which involves all the attributes of R_m. It may occur that no computation path exists in a given operations graph; in such case, no mediation query can be defined. Examples of computation paths in the operations graph given in figure 3 are $C_1 = (4, 2, 1)$ and $C_2 = (1, 2)$. The generation of a relational expression consists in identifying the set of all possible orderings of the operations contained in a computation path. C_1 and C_2 lead respectively to the following expressions:

  $E_1 = (T_4 \cup T_3) \bowtie T_2 \bowtie T_1$ and $E_2 = T_1 \bowtie T_2 \bowtie T_3$.

  To complete the process, expressions E_1 and E_2 should be rewritten using the definitions of mapping relations and transition relations. E_1 and E_2 become:

  $E_1 = \Pi_{K, A, B, C} [((\Pi_{B, C} S_4) \cup (\Pi_{B, C} S_3)) \bowtie (\Pi_{X, B} S_2) \bowtie (\Pi_{K, A, X} S_1)]$ and

  $E_2 = \Pi_{K, A, B, C} [(\Pi_{K, A, X} S_1) \bowtie (\Pi_{X, B} S_2) \bowtie (\Pi_{B, C} S_3)]$.

  The result of the approach is a set of mediation queries having different semantics. Given this set, one mediation query is selected either by the designer or using some heuristics.

3.2 Definitions

This subsection groups the essential definitions of the MQG algorithm which will serve for evolution purposes. Indeed, propagating changes raised at the source level will mainly consist in redefining relevant relations, operations graphs and computation paths.

Definition 3.1. Mapping relation (m-relation): An m-relation T_{mij}(K,X,X') between a mediation relation R_m and a source relation S_{ij} is defined as a relation having a schema composed of the following attributes: (X,X',K) such that: (i) $X \subseteq \underline{R}_m$, $X \subseteq \underline{S}_{ij}$ and $R_m.X \cong \underline{S}_{ij}.X$, (ii) X' is the foreign key attributes of S_{ij}, and (iii) K is the primary key attributes of S_{ij}.

Definition 3.2. Transition relation (t-relation): A t-relation T_{mij}(XY) is a projection of a non contributive source relation S_{ij} on its primary keys X and foreign keys Y such that: X is a foreign key of a mapping relation or another transition relation of R_m and Y is a primary key of a mapping relation or another transition relation of R_m.

Definition 3.3. Relevant relation (r-relation): A r-relation is either an m-relation or a t-relation.

Definition 3.4. Operations graph: An operations graph is the representation of both the relevant relations and the set of candidate operations. Each relevant relation, either m-relation or t-relation, is represented by a node in the operations graph. An edge between two nodes corresponding to relevant relations represents a candidate relational operation between these relations.

Definition 3.5. Computation path: This concept is used to generate mediation queries. A computation path in the operations graph G_{Rm} associated with the mediation relation R_m is either a relevant relation which involves all the attributes of R_m, or any connected and acyclic sub-graph of G_{Rm} which involves all the attributes of R_m.

4 Propagation of Source Changes to the Mediation Queries

As stated before, the evolution process of a GAV mediation system may take advantage of a rigorous design methodology like the one supported by the MQG algorithm. Propagating changes from the source level to the mediation level consists in detecting new contributive data sources or relations, redefining some mapping relations and/or transition relations, adding or removing operations from operations graphs, detecting new computation paths and selecting appropriate mediation queries. The evolution process consists mainly in adapting the MQG algorithm to make it more modular and more incremental. Hence, the new algorithm, called IMQG (incremental mediation query generation) makes no difference between an initial design based on a modular methodology and an evolution process.

To achieve the goal of a modular and incremental design, we assume that the mediation schema maintains a history of all design choices that have been made in a previous iteration. The operations graph plays a major role as it contains relevant relations, candidate operations, computation paths which have been selected and those which were not. Consequently, further incremental iterations of the algorithm will consist in updating only parts of this operations graph (history) which are precisely dependent of the change events raised at the sources, instead of redefining the whole mediation schema.

In the remaining of this section, we will first present the local schema change operations and the propagation primitives used to update the mediation level, then we will present the set of rules allowing to propagate the source changes into the operations graph, either on the relevant relations or on the associated operations. Finally, we will briefly discuss the global evolution process.

4.1 Local Schema Change Operations

Local schema change operations specify modifications that are performed in the local source schemas and that must be propagated to the mediation level. The possible change operations at the data source level are listed in Table 1.

Table 1. Local Schema Change Operations

Change Operation	Definition
add_relation($\underline{S}_{ij}$)	Adds a new relation schema into the source j
remove_relation($\underline{S}_{ij}$)	Removes an existing relation schema from the source j
add_attribute($\underline{S}_{ij}$, A)	Adds the attribute A into S_{ij}
remove_attribute($\underline{S}_{ij}$, A)	Removes the attribute A from S_{ij}
add_ref_constraint ($S_{ij}.A$, $S_{lj}.K$)	Adds the referential constraint $S_{ij}.A \rightarrow S_{lj}.K$ into S_j
remove_ref_constraint ($S_{ij}.A$, $S_{lj}.K$)	Removes the referential constraint $S_{ij}.A \rightarrow S_{lj}.K$ from S_j

Note that the removal or the addition of a data source can be represented in terms of a set of change operations, for example, the addition of a source can be considered as a set of *add_relation* operations and the removal of a data source can be considered as a set of *remove_relation* operations. In this paper, we will restrict ourselves to handle only some of the changes described in table 1.

4.2 Propagation Primitives

We consider that each mediation relation R_m is associated with an operations graph G_{Rm} corresponding to the mediation query defining R_m. If a change occurs in the data sources, some checking operations have to be performed on this graph to test if the computation path and therefore the mediation query associated with R_m are still valid. If not, new computation paths have to be searched and a new query has to be defined. A given computation path is invalid if one or more operations in this path is no longer valid. For example, if this path contains a join operation and if an attribute involved in the join predicate is removed from the data source, the join and consequently the computation path become invalid.

The propagation primitives presented in Table 2 specify modifications and verifications which must be performed in the operations graphs and the corresponding mediation queries to reflect local schema change operations.

Table 2. Mediation queries propagation primitives

Propagation Primitive	Definition
valid_operation_graph(G_{Rm})	Checks the validity of the operations graph G_{Rm} associated with the relation R_m and removes the invalid operations.
search_operation(G_{Rm})	Searches new operations for combining pairs of relevant relations in the operations graph G_{Rm}.
remove_operation(G_{Rm}, T_{ij}, A)	Removes all edges in the operations graph G_{Rm} that become invalid because of the removal of the attribute A from the relevant relation T_{ij}.
add_relevant_relation(T_{ij}, G_{Rm})	Adds relevant relation T_{ij} into the operations graph G_{Rm}.
remove_relevant_relation(T_{ij}, G_{Rm})	Removes relevant relation T_{ij} from the operations graph G_{Rm}.
search_relevant_relation(G_{Rm}, S)	Searches new relevant relations associated with relation R_m from the set of data sources S.
search_computation_path(G_{Rm})	Determines the computation paths associated with the operations graph G_{Rm}.
generate_query(G_{Rm}, Q)	Generates the set Q of relational expressions to compute relation R_m using the operations graph G_{Rm}.
select_query(Q, q)	Takes as input a set of possible queries Q and produces as output a single query q (the choice is made either by the designer or using some heuristics)

4.3 Evolution Rules for Relevant Relations

Given a source change represented by one of the local schema change operations described in section 4.1, we will first propagate these changes into the set of relevant relations associated with each mediation relation. To specify this propagation, we use event-condition-action (ECA) rules. The rules are classified according to the type of local schema change operations. Each rule has a name and a parameter denoted R_m which represents a mediation relation. G_{Rm} represents the operations graph associated with R_m. Table 3 gives a sample of such rules that we will discuss hereafter. Due to space limitations, only some of the evolution rules are presented in this table. A complete description is given in [3].

Table 3. Evolution rules

Rule 1 (R_m) **Event**: add_attribute($\underline{S}_{ij}$, A) **Condition**: $A \in \underline{R}_m$ $\exists T_{ij} \in M_{Rm} \mid \underline{T}_{ij} \subseteq \underline{S}_{ij}$ /*M_{Rm} is the set of relevant relations corresponding to the relation R_m */ **Action**: $\underline{T}_{ij} := \underline{T}_{ij} \cup \{A\}$, valid_operation_graph($G_{Rm}$), search_operation($G_{Rm}$).	**Rule 2 (R_m)** **Event**: add_attribute($\underline{S}_{ij}$, A) **Condition**: $A \in \underline{R}_m$ $\nexists T_{ij} \in M_{Rm} \mid \underline{T}_{ij} \subseteq \underline{S}_{ij}$ **Action**: /* X is the set of key attributes and foreign keys of S_{ij} */ $T_{ij} = \Pi_{X \cup A} S_{ij}$, add_relevant_relation($T_{ij}$, G_{Rm}), search_operation(G_{Rm}).
Rule 3 (R_m) **Event**: remove_attribute($\underline{S}_{ij}$.A) **Condition**: $\exists T_{ij} \in M_{Rm} \mid \underline{T}_{ij} \subseteq \underline{S}_{ij}$ $A \in \underline{T}_{ij}$ **Action**: $\underline{T}_{ij} := \underline{T}_{ij} - \{A\}$, remove_operation($G_{Rm}$, T_{ij}, A).	**Rule 4 (R_m)** **Event**: add_relation($\underline{S}_{ij}$) **Condition**: $\exists A \in \underline{S}_{ij} \mid A \in \underline{R}_m$ **Action**: /* X is the set of key attributes and foreign keys of S_{ij} */ $T_{ij} = \Pi_{X \cup A} S_{ij}$, add_relevant_relation($T_{ij}$, G_{Rm}), search_operation(G_{Rm}).
Rule 5 (R_m) **Event**: remove_relation($\underline{S}_{ij}$) **Condition**: $\exists T_{ij} \in M_{Rm} \mid \underline{T}_{ij} \subseteq \underline{S}_{ij}$, **Action**: remove_relevant_relation(T_{ij},G_{Rm}).	**Rule 6 (R_m)** **Event**: add_ref_constraint(S_{ij}.A, S_{kj}.K) **Condition**: $\exists T_{ij} \in M_{Rm} \mid \underline{T}_{ij} \subseteq \underline{S}_{ij}$ $\exists T_{kj} \in M_{Rm} \mid \underline{T}_{kj} \subseteq \underline{S}_{kj}$ $A \in \underline{T}_{ij}$ **Action**: search_operation(G_{Rm}).

- *Adding an attribute into a relation of a local schema:* Rules 1 and 2 update the relevant relations corresponding to the mediation relation R_m after the insertion of a new attribute A into a local source relation S_{ij}. The Rule 1 checks if the new attribute A belongs to the set of attributes of R_m and if there is a relevant relation T_{ij} associated with R_m over S_{ij}. If the attribute A belongs to the relation R_m, this means that there is at least one relevant relation T_{xy} containing the attribute A and derived from a source relation S_{xy} which is distinct from S_{ij}. If the conditions are true, the attribute A must be inserted into the relevant relation T_{ij}. As a consequence of this, the operations graph G_{Rm} must be validated and new operations must be searched.

The validation of the operations graph consists in verifying the validity of the existing operations after the insertion of the attribute A into the relevant relation T_{ij}, and removing the invalid operations. The Rule 2 checks if the new attribute A belongs to the set of attributes of R_m and if there is no relevant relation T_{ij} derived from the source relation S_{ij}. In this case, a new relevant relation should be added to the operations graph G_{Rm}. As a consequence of this, new operations must be searched in G_{Rm}.

- *Removing an attribute from a relation of a local schema:* Rule 3 updates the set of relevant relations and the set of candidate operations that define the mediation relation R_m after the deletion of the attribute A from the local relation S_{ij}. The condition part checks if there is a relevant relation T_{ij} associated with R_m over S_{ij} and if the removed attribute A belongs to the relevant relation T_{ij}. The attribute A must be removed from the relevant relation T_{ij} and all edges in G_{Rm} representing operations which are no longer valid must be removed. It may occur that no computation path can be found after the propagation; in such case, the mediation relation R_m becomes no longer computable.

- *Adding a relation into a local schema:* Rule 4 updates the set of relevant relations defining the mediation relation R_m after the addition of a local relation S_{ij}. The condition part checks if there is a set of attributes A in the source relation S_{ij} that belongs to the schema of R_m. If such set exists, then a relevant relation T_{ij} must be added into the operations graph G_{Rm}. As a consequence of this, new operations must be searched in G_{Rm}.

- *Removing a relation from a local schema:* Rule 5 updates the set of relevant relations associated with the relation R_m in the mediation schema after the deletion of a local relation S_{ij}. The condition part checks if there is a relevant relation T_{ij} associated with R_m over S_{ij}. To reflect the deletion of the local relation S_{ij}, the corresponding relevant relation T_{ij} must be removed from the operations graph G_{Rm}, along with all the operations involving T_{ij}. It may occur that no computation path can be found after the propagation; in such case, the mediation relation R_m becomes no longer computable.

- *Adding a referential constraint involving two relations in the same data source:* Considering that the constraint involves the source relations S_{ij} and S_{kj}, rule 6 checks if there are two mapping relations T_{ij} and T_{kj} derived from S_{ij} and S_{kj} respectively and if the attribute A belongs to T_{ij}. In such case, new operations are searched between the two mapping relations.

The propagation of source changes in the relevant relations and the query generation are the two main steps of the global evolution process which is described in the next section.

4.4 The Global Evolution Process

This section describes the global evolution process used in our approach to propagate source schema changes to the mediation level. A general overview of this process is presented in figure 4.

The global evolution process receives events (*e*) representing source schema changes and propagates them to the mediation level. An event is a source change represented by one of the local schema operations described in section 4.1. We consider that the events are produced by the data sources participating in the data integration system (or by a specific process called *Lookup* which detects these changes automatically).

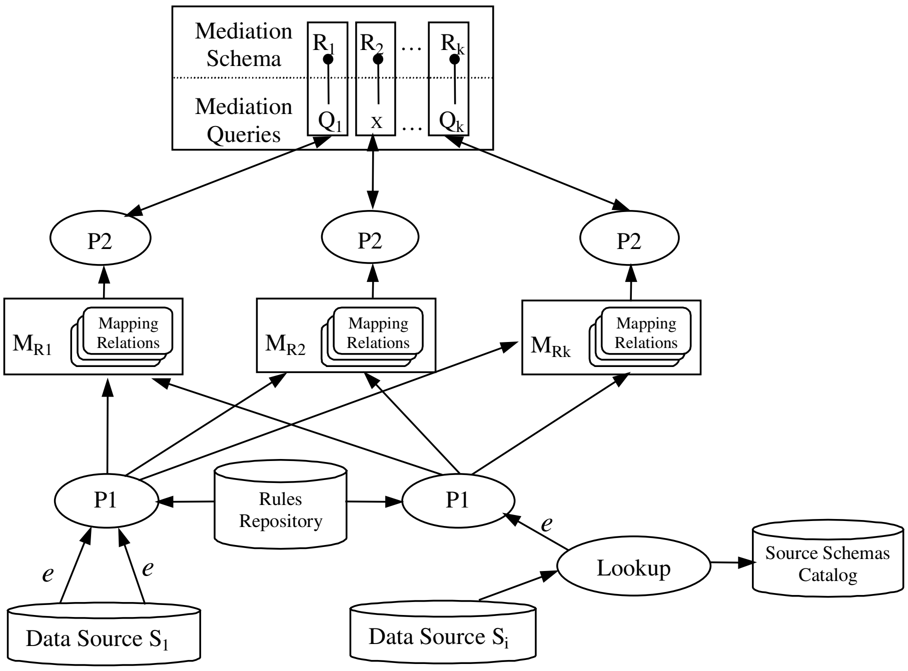

Fig. 4. Global Evolution Process

The evolution process can be divided into two main steps: a relevant relation evolution step and a mediation queries generation step. The relevant relations evolution consists in propagating source schema changes to the relevant relations associated with each mediation relation using the set of ECA rules described in section 4.3. This step is executed by a set of processes, where each process, called relevant relations evolution process and denoted *P1* is associated with a distinct data source S_i and is responsible for updating the relevant relations derived from S_i. In this way, different processes can be executed concurrently updating distinct sets of relevant relations.

The mediation queries generation step consists in generating a mediation query for each mediation relation when the corresponding set of relevant relations is modified. Each mediation relation R_m is associated with a process, called mediation queries generation and denoted *P2*, which is responsible for generating the mediation query used to compute R_m. Note that a mediation relation may become invalid after a source change; in figure 4, the symbol *X* linked to a mediation relation represents the fact that no mediation query has been found for this relation. There are two possible strategies for the global evolution process: (i) *Query driven evolution*, in which new mediation queries are generated when user queries are posed to the system and (ii) *Source driven evolution*, in which new mediation queries are generated when a set of changes is notified by a data source.

One of the problems in the query driven strategy is that, since the new mediation query is generated at evaluation time, the response time may increase. Besides, the process of mediation queries generation can result in a set of mediation queries, and it is therefore necessary to interact with the system administrator in order to ask him to choose one of them. The main problem with the source driven evolution strategy is that we have to find the best way of synchronizing the propagation of events notified by different data sources in order to minimize the effort of generating the mediation queries.

In our work, we have adopted the source driven evolution strategy and the synchronization between processes *P1* and *P2* is done as follows: we consider that for a given mediation relation R_m, the process *P2* is executed only when all processes *P1* have finished the evolution of the relevant relations. This means that the query generation process will start only when all the events notified by all the data sources have been propagated in the corresponding relevant relations. The relevant relations corresponding to the relation R_m are locked by the processes *P1*; when these locks are released, the process *P2* can lock the relevant relations and start its execution. Another possible strategy would consists in executing the process *P2* whenever a process *P1* signals the end of the evolution of the relevant relations. In this case, the generation process will take place as many often as the number of sequences of events notified by the data sources.

5 Related Works

In [1] an approach is presented which minimizes the cost of query processing for a mediator by pre-compiling the source descriptions into a minimal set of integration axioms. An integration axiom specifies a particular way in which available sources can be combined to provide the data for a class belonging to the global model. The current version of the proposed algorithm to discover integration axioms is incremental, which means that when new sources are added, the system can efficiently update the axioms, but no details on how this could be achieved nor examples are given. In the current version of the algorithm, in case of deleting a source the algorithm must start from scratch. A LAV approach is used to define the mappings between the global model an the local sources, while we have adopted a GAV approach in this paper.

The problem of evolution is also discussed in [8], which provides an approach to handle both schema integration and schema evolution in heterogeneous database architectures. The proposed approach is based on a framework for schema transformation, which consists of a hypergraph-based common data model and a set of primitive schema transformations defined for this model. Source schemas are integrated into a global schema by applying a sequence of primitive transformations to them. This set of transformations can also be used to systematically adapt the global schema and the global query translation pathways after changes to the source schemas. Instead of defining mediation queries as in many mediation systems, they use some transformations to automatically translate queries posed to the global schema to queries over the local schemas. In the proposed framework the schemas are defined in a hypergraph-based data model (HDM) and queries are expressed using Datalog.

The work presented in [9] is one of the few which studies the view evolution problem in information integration systems. The authors propose the Evolvable View Environment framework (EVE) as a generic approach to solve issues related to view evolution under schema changes for both view definition adaptation and view extent maintenance after synchronization. EVE uses materialized views for data integration. They propose some view synchronization algorithms to evolve a view definition by finding appropriate replacements for affected view components based on available meta-knowledge, and by dropping non-essential view components. Unlike the strategies proposed for query rewriting using views [6, 13], the proposed algorithms find view rewritings that are not necessarily equivalent to the original definition. Similarly to our approach they use the relational data model as the common data model and they also propose an extended view definition language (derived from SQL), which allows users to explicitly define evolution preferences for changing the semantics of the view. In this way, it is possible to accept view rewritings that preserve only the required attributes if preserving all is not possible.

Our approach differs from the approach proposed in [9], because the modifications are not directly executed in the mediation query definition but in the metadata that describes the mediation query. We consider that a mediation query must be modified as a consequence of every kind of source schema change. In [9], a view must evolve only when a source schema change occurs that can make the view definition obsolete; i.e., only the cases of removal of relations or attributes are dealt with. In the current version of our algorithm, we consider that all attributes of a mediation query are required, i.e., when an attribute cannot be found then the mediation query is invalid.

6 Conclusion

In this paper, we have presented a solution to the problem of evolution of mediation queries in a GAV approach. This solution is an improvement of the MQG algorithm which has been defined to generate mediation queries. The improvement consists in adapting the MQG algorithm to make it more incremental by introducing a set of propagation rules that operate on operations graphs and reflect the changes made at the source schema level. Propagation rules are formalized as event-condition-action (ECA) rules whose events correspond to the change operations and the actions to a set of propagation primitives. Conditions specify the semantic context in which the transformations are valid.

One interesting aspect to investigate in future works is the impact on the mediation level (schema and queries) of user requirements changes. In the same way as changes in the source schemas are, changes in the users' requirements must be reflected in the mediation level. In that case, the assumption on the invariance of the mediation schema is relaxed and the advantage of the LAV approach compared to the GAV approach is lost, because a change in the mediation schema will result in reconsidering and possibly rewriting all the mediation queries. Consequently, the evolution process in this case will mainly concern the LAV approach, because in the GAV approach, a change at the mediation schema results in the redefinition of one mediation query. One perspective of this work in the near-future is to propose an adaptation of our approach to handle user requirements changes in a LAV context.

References

1. J. Ambite, C. Knoblock, I. Muslea and A. Philpot, Compiling Source Description for Efficient and Flexible Information Integration, Journal of Intelligent Information Systems (JIIS), vol. 16, no.2, Mar. 2001.
2. J. Banerjee, W. Kim, H. J. Kim, and H. F. Korth, Semantics and Implementation of Schema Evolution in ObjectOriented Databases, in Proc. of SIGMOD, pp. 311–322, 1987.
3. M. Bouzeghoub, B. Farias-Loscio, Z. Kedad, A.C. Salgado, Design and Evolution of Large Scale Mediation Systems, Technical report, september 2003.
4. Y. Halevy, Theory of answering queries using views, SIGMOD Record, vol. 29, no.4, pp.40–47, 2000.
5. Z. Kedad and M. Bouzeghoub, Discovering View Expressions from a Multi-Source Information System, in Proc. of the Fourth IFCIS International Conference on Cooperative Information Systems (CoopIS), Edinburgh, Scotland, pp. 57–68, Sep. 1999.
6. A. Y. Levy, A. Rajaraman, J. D. Ullman, Answering Queries Using Limited External Processors, in Proc. of Symposium on Principles of Database Systems – PODS, pp. 227–237, 1996.
7. A. Y. Levy, Logic-based techniques in data integration, in J. Minker, editor Logic Based Artificial Intelligence. Kluwer Publishers, 2000.
8. P. McBrien and A. Poulovassilis, Schema Evolution in Heterogeneous Database Architectures, A Schema Transformation Approach, in Proc. of CAiSE'02, pp. 484–499, Toronto, May 2002.
9. A. Nica and E. A. Rundensteiner, View maintenance after view synchronization, in Proc. of International Database Engineering and Application Symposium (IDEAS'99), Apr. 1999.
10. Y. G. Ra and E. A. Rundensteiner, A Transparent Schema Evolution System Based on ObjectOriented View Technology, IEEE Transactions on Knowledge and Data Engineering, pp. 600–624, September 1997.
11. E.A. Rundensteiner, A. Lee, and Y.G. Ra, Capacityaugmenting schema changes on object-oriented databases: Towards increased interoperability, in ObjectOriented Information Systems, 1998.
12. D. Sjoberg, Quantifying Schema Evolution. Information and Software Technology, vol. 35, no.1, pp. 35–54, Jan. 1993.

13. D. Srivastava, S. Dar, H. V. Jagadish, A. Y. Levy, Answering Queries with Aggregation Using Views, in Proc. of 22th International Conference on Very Large Data Bases, pp. 318–329, 1996.
14. J. D. Ullman, Information integration using logical views, in Proc. of ICDT'97, vol.1186 of LNCS, pp.19–40, Springer-Verlag, 1997.
15. G. Wiederhold, Mediators in the architecture of future information systems, IEEE Computer, pp.38–49, Mar. 1992.