Dynamic Maintenance of an Integrated Schema

Regina Motz

Instituto de Computación
Universidad de la República
Montevideo - Uruguay
rmotz@fing.edu.uy

Abstract. This work presents a schema evolution methodology able to propagate structural and semantic modifications occurring in the local schemas of a federated database to the integrated schema. Our approach is to regard this problem from a schema integration point of view. Our theoretical framework is based on a declarative schema integration methodology, which reduces schema integration to the resolution of a set of equivalence correspondences between arbitrarily complex local subschemas.

1 Introduction

Considering the importance of schema integration methodologies in data warehousing, e-commerce as well as the growing use of cooperative engineering, the relevance of schema evolution of an integrated schema in a federated database becomes very important and necessary. Such an evolution may be due to: (a) changes in the structure of the component schemas, or (b) changes in the semantics of correspondences between the component databases. The former case is referred to as *structural modifications*, while the latter as *semantic modifications*.

A key characteristic of a federation is the autonomy of their component databases: their instances and schemas may evolve, but independently. This fact naturally leads to the introduction of the concept of *dynamic maintenance* of an integrated schema defined as the process of propagating changes of the component databases to the integrated one in an efficient, consistent and user-independent way. This means to avoid as much as possible re-integration steps during the propagation process.

In this paper, we present a methodology which reduces propagation to a form of *incremental schema integration*. In contrast with traditional integration methodologies, which always require as input the whole set of correspondences between the local schemas and a complete re-integration every time a component modification occurs, the one we present permits to provide new correspondences in an incremental manner and re-integrate only on the affected portions of the integrated schema.

Our approach is based on a declarative schema integration methodology, called SIM [6], which accomplishes schema integration by resolving a set of equivalence correspondences between arbitrary component subschemas. A relevant aspect of our work is the semi–automatic derivation of new states for the already acquired set of mappings from the local schemas to the integrated one after the occurrence of local schema changes. This is achieved by the development of a framework for the evolution of an integrated

M. Hemmje et al. (Eds.): E.J. Neuhold Festschrift, LNCS 3379, pp. 21–30, 2005.

schema based on a mechanism that incorporates *incrementality* in the integration process. Another contribution of our work is that we show how it is possible to propagate *purely structural changes*, *capacity augmenting changes* as well as *capacity reducing changes* in a uniform manner.

Motivated by the fact that some local schema changes may be considered as cases of more elaborated "semantic correspondences" between subschemas, we regard the propagation of local semantic modifications as a form of *incremental semantic* schema integration. For this purpose we extend SIM to deal with an enriched set of semantic correspondences, namely inclusion, overlapping and grouping, as well as equivalence correspondences between one class and multiple classes. From the set of semantic correspondences between the local schemas, our methodology automatically induces structural transformations that make it possible to regard the given correspondences as equivalences. On this basis, the propagation of local semantic modifications reduces to a process which identifies the original schema correspondences affected by the schema change and reintegrates the affected portions of the integrated schema.

The paper is organized as follows. Section 2 describes the schema integration methodology our work is based on. In Section 3 we make an account of local semantic and structural modifications and show, by means of examples, the problems that component schema changes may cause on the integrated schema. In Section 4 we describe our approach to the dynamic maintenance of an integrated schema. Section 5 summarizes related work. Finally, in Section 6 we draw some conclusions and present directions for further work.

2 The Schema Integration Methodology: SIM

SIM [6] is a declarative schema integration methodology that works on schema graphs. A *schema graph* $G = (V, E, S, K)$ is a directed graph that captures the inheritance hierarchies and relationships of a schema. The vertices of a schema graph are of two kinds, $V = C \cup T_I$, where C is a finite set of classes and T_I is a set of immutable types. $E = E_R \cup E_A$ is a set of labelled edges. E_R corresponds to a set of relationship edges between classes. We will denote relationship edges by $p \overset{a}{\longleftrightarrow} q$, where $p, q \in C$ and $a \equiv (\alpha, \beta)$, for $\alpha, \beta \in L_E$ (a set of edge labels). $E_A \subseteq C \times L_E \times V$ is a set of attribute edges. $S \subseteq C \times C$ is a finite set of specialization edges, which will be denoted by $p \Longrightarrow q$, meaning that p is a specialization of q. We will write $p{=\!=}q$ to denote an arbitrary edge. Finally, $K = K_R \cup K_A$ is a function which associates cardinality constraints to relationship and attribute edges. Not every schema graph specifies a valid schema. A schema graph G is said to be a *proper schema* when it satisfies the following restrictions:

1. *Uniqueness in the context of a class.* The occurrence of a relationship/attribute edge in the context of a class is unique.
2. *Acyclicity of subtyping.* Class specializations are acyclic.
3. *Monotonicity of Inheritance.* Specialization is preserved along equal attributes / relationships.

SIM reduces schema integration to the resolution of a set of equivalence vertex/path correspondences between subschemas of the local databases. From such a set of correspondences, SIM semi-automatically derives mappings, called *schema augmentations*, from each local schema to the integrated one, in such a way that corresponding data among local databases is mapped to the same structure in the integrated database. The generated schema augmentations enhance the schemas with classes and paths, such that the resulting integrated schema is a non-redundant proper schema graph.

Formally, this means that it is the least upper bound $(G_1 \sqcup_{\mathcal{A}} G_2)$ of the local schemas G_1 and G_2 under an augmentation ordering $\sqsubseteq_{\mathcal{A}}$. Two proper schema graphs $G = (V, E, S, K)$ and $G' = (V', E', S', K')$ are in an augmentation ordering, denoted by $G \sqsubseteq_{\mathcal{A}} G'$, iff $V \subseteq V'$, $S \subseteq S'$ and there exists a mapping, denoted by $\mathcal{A} : G \to G'$ and called an *augmentation*, which maps each vertex/edge of G to a proper subschema of G', satisfying the following conditions:

1. The mapping of the empty graph is the empty graph.
2. Each class $p \in V$ is mapped to a subschema of G' which contains at least p.
3. The augmentation of two distinct classes in G do not contain a common class in G'.
4. Each specialization $p \Longrightarrow q$ in G is mapped to a specialization path $p \overset{*}{\Longrightarrow} q$ in G'.
5. Each relationship $p \overset{a}{\longleftrightarrow} q$ in G is mapped to a path in G' with begin and end given by p and q, respectively, and of this form: $p \overset{*}{\Longrightarrow} p' \overset{r_1}{\longleftrightarrow} A_1 \overset{r_2}{\longleftrightarrow} \ldots \overset{r_n}{\longleftrightarrow} A_n \overset{r_{n+1}}{\longleftrightarrow} q' \overset{*}{\Longleftarrow} q, n \geq 0$.

   If, in addition, the relationship occurs in the path of G' as $p' \overset{a}{\longleftrightarrow} q'$, then $p \overset{*}{\Longrightarrow} p'$ and $q \overset{*}{\Longrightarrow} q'$, and $K(p \overset{a}{\longleftrightarrow} q) \leq K'(p' \overset{a}{\longleftrightarrow} q')$.
6. Each attribute edge $p \overset{a}{\rightarrow} q$ in G is mapped to a path in G' of the form: $p \overset{*}{\Longrightarrow} p' \overset{r_1}{\longleftrightarrow} A_1 \overset{r_2}{\longleftrightarrow} \ldots \overset{r_n}{\longleftrightarrow} A_n \overset{a}{\rightarrow} q$, with $n \geq 0$, such that the path $p \overset{*}{\Longrightarrow} p' \overset{r_1}{\longleftrightarrow} A_1 \overset{r_2}{\longleftrightarrow} \ldots \overset{r_n}{\longleftrightarrow} A_n$ is included in the mapping of p.

An important aspect of SIM is that mappings are injective functions. The methodology recognizes inconsistent correspondences for which there is no augmentation possible, and identifies ambiguous correspondences, for which there exists no unique integrated schema. For consistent correspondences, the methodology generates appropriate augmentations and an integrated schema automatically. An important aspect is that all path correspondences that overlap in some subpath are grouped as a pair of trees and resolved together recursively. A complete description of SIM can be found in [6].

3 Component Modifications

The aim of this section is to illustrate the impact that component schema changes may produce on an already integrated schema. We consider two types of schema changes: semantic and structural modifications.

3.1 Semantic Modifications

A common criterion to identify the differences between two databases is to compare their intentions [7,12,8]. The *intention* of a schema graph G is defined as the set of all

its possible valid instances. Semantic modifications on a schema may then be identified by inspecting the relationship that holds between its intentions before and after the change. On this basis, we can identify the following kinds of semantic modifications: *inclusion* (increment in the set of possible valid instances), *exclusion* (reduction in the set of valid instances), *overlapping* (some instances are maintained, others dismissed and new ones included) and *disjoint* (the new set of valid instances does not intersect with the old one). These local semantic modifications require the specification of new semantic correspondences between the local schemas. The enriched set of semantic correspondences contains equivalence ($\equiv$), inclusion ($\subset$), overlapping ($\odot$) and grouping ($\gg$). An important feature is that all kind of semantic correspondences can be expressed as a structural schema modification plus a set of equivalence correspondences. Based on this feature our approach is to treat semantic modifications as structural modifications.

Let us see this with an example. Consider the schemas of two travel agencies, Travel Makes Fun (TMF) and Easy Travel (ET). In both travel agencies, excursions need to be reserved in advance and this is represented by a class $Reservation$ in each schema. Suppose that, initially, there is a constraint in both schemas which states that reservations can only be carried out with advance the payment of an amount. After some time the Easy Travel agency changes its policy allowing reservations without any payment. This modification in the semantics of $Reservation_{ET}$ induces the following new semantic correspondence: $Reservation_{ET} \subset Reservation_{TMF}$. By applying a structural transformation this inclusion correspondence can be treated as follows:

- *Structural modification*: A class $PartiallyPaid_{ET}$, modeling those reservations that are made after paying some amount, is added as subclass of $Reservation_{ET}$. This means that $Reservation_{ET}$ permits doing reservations without initial payment. The definition of $Reservation_{ET}$ as a generalization of $PartiallyPaid_{ET}$ models the inclusion relationship between the extensions of $Reservation_{TMF}$ and $Reservation_{ET}$.
- *New equivalence correspondence*: With the definition of $PartiallyPaid_{ET}$, an equivalence correspondence between $Reservation_{TMF}$ and $PartiallyPaid_{ET}$ now holds.

This example shows the necessity of addressing the resolution of structural modifications for the propagation of local semantic modifications.

3.2 Structural Modifications

These are modifications to the structure of local schemas. They are more complex than semantic modifications because they can not only affect the instances of a class or relationship, but rather the whole schema. Therefore, as a consequence of a local structural modification a set of correspondences between local schemas can also be affected. The problem is that there are some cases in which it is not possible to automatically obtain an evolved integrated schema from the old one due to the occurrence of ambiguity or inconsistency in the new set of correspondences.

Modifications that lead to ambiguity: Consider the previous two travel agencies. In both schemas, excursions, represented by a class *Excursion*, follow predefined itineraries.

Each itinerary consists of a set of tours and a set of stops. While Easy Travel models itineraries as objects, in Travel Makes Fun they are given by the values of the attributes *tours* and *stops* that belong to the class *Excursion*. For simplicity reasons, we assume that elements with equal names in both schemas correspond with each other. Therefore, the following correspondences hold initially:

$$\textit{Excursion} \xrightarrow{stops} \text{string} \equiv \textit{Excursion} \leftrightarrow \textit{Itinerary} \xrightarrow{stops} \text{string}$$
$$\textit{Excursion} \xrightarrow{tours} \text{string} \equiv \textit{Excursion} \leftrightarrow \textit{Itinerary} \xrightarrow{tours} \text{string}$$

Now, suppose that the attributes *tours* and *stops* of $\textit{Excursion}_{TMF}$ evolve to a new class *Tour*. This modification affects the set of correspondences between both local schemas. The new correspondences are:

$$\textit{Excursion} \leftrightarrow \textit{Tour} \xrightarrow{stops} \text{string} \equiv \textit{Excursion} \leftrightarrow \textit{Itinerary} \xrightarrow{stops} \text{string}$$
$$\textit{Excursion} \leftrightarrow \textit{Tour} \xrightarrow{tours} \text{string} \equiv \textit{Excursion} \leftrightarrow \textit{Itinerary} \xrightarrow{tours} \text{string}$$

Traditional schema integration methodologies can not propagate this change automatically due the following ambiguity: Is the class *Tour* corresponding to the class *Itinerary*? Interaction with the user is necessary to solve this ambiguity. However, the main problem is to identify the correspondences affected by the ambiguity.

Modifications that lead to inconsistency: Consider now the evolution of Easy Travel presented in Figure 1. The class *TrainExc* is moved through the hierarchy of classes to appear as a subclass of *SpecialExc*. In this case there is no possible propagation of the local change because of an inconsistency between the correspondences that represent the modification and the initial ones, see Figure 1. The reason of the inconsistency is that for the new set of correspondences:

$$\textit{Excursion} \leftarrow \textit{IndivExc} \leftarrow \textit{AirExc} \equiv \textit{Excursion} \leftarrow \textit{AirExc}$$
$$\textit{Excursion} \leftarrow \textit{IndivExc} \leftarrow \textit{TrainExc} \equiv \textit{Excursion} \leftarrow \textit{SpecialExc} \leftarrow \textit{TrainExc}$$
$$\textit{Excursion} \leftarrow \textit{FluvialExc} \equiv \textit{Excursion} \leftarrow \textit{SpecialExc} \leftarrow \textit{FluvialExc}$$

it is not possible to fulfill all of them simultaneously. The direct propagation of such a schema change is usually rejected by most methodologies because they maintain previous decisions taken by the user.

Our methodology, in contrast, supplies the user the set of correspondences that lead to the inconsistency in order to eliminate one of them, obtaining in that way a consistent set of correspondences. In the next section we describe how to identify the set of correspondences that lead to the ambiguity or inconsistency.

4 Dynamic Maintenance

A key characteristic of the maintenance of an integrated schema is the fact that a local schema evolution may invalidate correspondences established in some previous integration steps. Our solution to this problem is to identify the augmentations affected by the

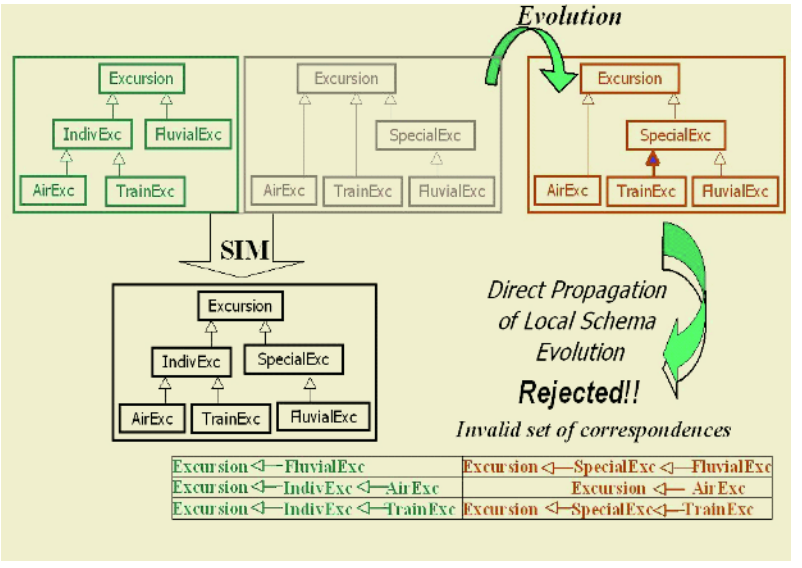

Fig. 1. Structural Modification that leads to inconsistency

local structural modification, and either modify them directly or deduce from them the involved corresponding subschemas where re-integration is required.

Schema changes may produce the addition or deletion of correspondences between local subschemas. In this sense we can regard the propagation of local schema changes as a form of *incremental schema integration*. To achieve this, we provide SIM with a mechanism able to treat ambiguities or inconsistencies incrementally. Such a mechanism essentially consists of two phases: first, it identifies the subschema of the integrated schema that has been affected by a given set of correspondences, and second, it manipulates this subschema so that it becomes consistent.

Semantic modifications are regarded as semantic correspondences between local schemas. As mentioned previously, semantic correspondences can in turn be expressed as structural modifications plus a set of equivalence correspondences. Given a vertex or path correspondence $p_1 \star p_2$ between two local schemas G_1 and G_2, with $\star \in \{\equiv, \subset, \odot, \gg\}$, $p_1 \in G_1$ and $p_2 \in G_2$, we define a structural transformation $\mathcal{T}(p_1 \star p_2) = (\mathcal{T}_1, \mathcal{T}_2, EC)$ given by structural mappings $\mathcal{T}_1 : p_1 \rightarrow G_1'$ and $\mathcal{T}_2 : p_2 \rightarrow G_2'$, and a set of equivalence correspondences EC between the modified schemas G_1' and G_2'. The new set of equivalence correspondences are then handled using the techniques developed to the propagation of structural modifications.

The propagation of structural modifications is performed by our methodology as follows. Given two local schemas G_1 and G_2 and a local schema change in G_1 represented as a subschema C_1, we identify the corresponding portions of the local schemas as well as of the integrated schema affected by the schema change. We start finding the image of C_1 in the integrated schema by applying the augmentation mapping $\mathcal{A}_1$, $IC_1 = \mathcal{A}_1(C_1)$.

Then, by applying the inverse of the other augmentation[1] on that image, we obtain a subschema of G_2, $C_2 = \mathcal{A}_2^{-1}(IC_1)$. The subschema of G_1 that is in correspondence with C_2 is obtained by applying the mappings in the other direction, i.e. $C_3 = \mathcal{A}_1^{-1}(\mathcal{A}_2(C_2))$. Finally, we have found the corresponding subschemas affected by the schema change. It is worth mentioning that the correspondence between the subschemas is expressed as a set of equivalence vertex/path correspondences.

The local structural modifications we handle are within the following categories:

1. Modifications that extend an edge to a path
   (a) *Refinement of a specialization hierarchy*
   An edge $x \Leftarrow y \in S_i$ evolves to a path : $x \Leftarrow z \Leftarrow y$.
   (b) *Generalization of multiple classes*
   Edges $x \Leftarrow y_j \in S_i$ evolve to paths: $x \Leftarrow z \Leftarrow y_j$ with $j = 1 \cdots n$.
   (c) *Objectification of attributes/relationships*
   An edge $x{=\!=}y_j \in E_i$ evolves to a path: $x{=\!=}z{=\!=}y$ with $j = 1 \cdots n$.
   (d) *Generalization of relationships*
   An edge $x \overset{r}{\longleftarrow} y \in E_i$ evolves to a path:
   (e) *Specialization of relationships*
   An edge $x \overset{r}{\hookleftarrow} y \in E_i$ evolves to a path:
2. Addition of a new vertex or edge
3. Deletion of an existing vertex or edge
4. Modifications that compress a path to an edge
   (a) *Flatting of a specialization hierarchy*
   A path $x \overset{+}{\Leftarrow} z \overset{+}{\Leftarrow} y \in G_i$ evolves to the edge $x \Leftarrow y$.
   (b) *De-Objectification of attributes/relationships*
   A path $x \overset{+}{=\!=} z \overset{+}{=\!=} y \in G_i$ evolves to the edge $x{=\!=}y$.

In modifications of type (1) and (4) the crucial point is how to identify path correspondences affected by an edge. Given an edge $x{=\!=}y \in G_1$, the path correspondences in the affected subschemas are given by the *proper extension* of the edge, which is calculated as the componentwise concatenation of the paths pairs belonging to the right and left extension of the edge. By *right extension* of an edge $x{=\!=}y$, we understand the set of path pairs (p_1, p_2), with $p_1 \in G_1,\ p_2 \in G_2$ and $p_1 = (x{=\!=}y) \cdot p_1'$, which are derived by simultaneously traversing the graphs G_1 and G_2 following those paths indicated by the augmentations $\mathcal{A}_1$ and $\mathcal{A}_2$. The *left extension* of $x{=\!=}y$ is defined analogously except that, instead of at the beginning, $x{=\!=}y$ appears at the end in all paths derived from G_1 in the extension, i.e. for each path pair (p_1, p_2), $p_1 = p_1' \cdot (x{=\!=}y)$. The existence of such a proper extension of an edge is ensured by construction of augmentations. In fact, augmentations $\mathcal{A}_1$ and $\mathcal{A}_2$ are built from sets of path correspondences, where each set can be regarded as a pair of corresponding trees with roots in vertex correspondences. When building the extension of an edge $x{=\!=}y$, we are essentially walking in parallel on such a pair of trees, determining those path pairs, connecting the root of the corresponding tree with a leaf, such that they contain the edge $x{=\!=}y$.

All path correspondences that overlap in some subpath may be regarded as forming a pair of trees, from which the integration process produces an augmentation tree in the

[1] The inverse always exists because augmentations are injective functions

integrated schema. By definition of augmentation, an edge may belong to at most one augmentation tree.[2] This means that we can derive those parts of the original local trees formed by the path correspondences that contain the given edge by "ideally" walking through the augmentation tree. However, in some cases the subschemas affected by an edge are formed by more path correspondences than only those in the extension of the given edge. This leads us to consider the whole augmentation tree —or, equivalently, to consider the local trees completely— in order to include also those path correspondences that do not contain the given edge. Let us see this with an example.

Consider the following corresponding schemas:

with these initial path correspondences:

$$A \xleftarrow{r} B \xrightarrow{c} C \equiv A \xrightarrow{c} C \qquad A \xleftarrow{r} B \xrightarrow{d} D \equiv A \xrightarrow{d} D$$

Vertices with equal name are in vertex correspondence. The integrated schema is then given by G_1. Suppose we perform the following local schema modification: $A \xrightarrow{c} C$ of G_2 evolves to the path $A \xleftarrow{m} M \xrightarrow{c'} C$. Let us consider that the subschema affected by $A \xrightarrow{c} C$ is only formed by the extension of the edge, i.e. by the path correspondence: $A \xleftarrow{r} B \xrightarrow{c} C \equiv A \xrightarrow{c} C$.

If we now integrate on the updated version of this path correspondence, namely

$$A \xleftarrow{r} B \xrightarrow{c} C \equiv A \xleftarrow{m} M \xrightarrow{c'} C$$

an ambiguity arises, because all these alternatives are equally valid:

$$\mathcal{A}_1(A) = A \xleftarrow{m} M \qquad \mathcal{A}_2(A) = A \xleftarrow{r} B \qquad B \equiv M$$

On the contrary, if we consider that the subschemas affected are formed by the complete augmentation tree to which belongs the given edge $A \xrightarrow{c} C$, then no ambiguity arises and it can be automatically deduced that $\mathcal{A}_2(A) = A \xleftarrow{r} B$.

To derive the whole local trees, we need to determine which vertex is the root of the calculated extension of the given edge. After having the root, we can calculate the extension of the edge starting at the root, obtaining the whole local trees, since all paths correspondences from the root towards the leaves necessarily traverse this edge. The problem then reduces to finding the root. When the extension of the given edge is formed by several path correspondences, it reduces to finding the overlapping parts of the paths at hand. On the other hand, when the extension is formed by a single path correspondence we need to test all four begin/end edges of the paths in the path correspondence in order to determine the desired root.

The description of the corresponding algorithms can be found in [14].

[2] This is not the case with a vertex in vertex correspondence, as it may belong to several trees.

5 Related Work

Our framework for the dynamic maintenance of an integrated schema permits the propagation of local schema changes in tightly coupled federated databases. It manages the impact of structural as well as semantic local changes by means of a semi-automatic mechanism that propagates the changes to the integrated schema without information loss and with a minimal amount of re-integration steps. Our work shares common subgoals with areas like *schema evolution* [5,9], *schema impact analysis* [2,3], and *federated view update mechanisms* [1]. The FEvoS project [15] uses ontologies combined with a fuzzy logic approach to identify the discrepancies produced by structural as well as semantic local schema changes. FEvoS shares with our work the interest in identifying the subschemas affected by a local schema change, but it does not proceed to re-establish the discrepancies as we do.

Schönhoff *et. al* [16] propose to use version management in the integrated schema. For each local schema change, the proposed service makes a new local version "visible" at the global layer of the federation, and vice-versa. It tries to identify automatically properties like a new version's history and predecessor. The service requires user assistance to propagate completely the information to the new version.

Recent works [11,13] follow a schema transformation approach to both schema integration and schema evolution. Source schemas are integrated into a global one by applying a sequence of primitive transformations to them. The same set of primitive transformations are used to specify the evolution of a source schema. In our work we also follow a schema transformation approach in which integration and evolution are treated uniformly. However, while [11] and [13] resolve local schema evolution in a loosely-coupled federation, we resolve the problem in a tightly-coupled federation. Another relevant difference with [11,13] is the fact that our framework is declarative, whereas theirs is operational.

6 Conclusions and Further Work

We presented a methodology for the dynamic maintenance of an integrated schema based on an *incremental schema integration* approach. This means that we avoid as much as possible re-integration steps in the propagation process. An essential aspect of this process is that, from the augmentation mappings for the already integrated schema and the local schema changes, we semi-automatically derive a new state for the augmentations. The user is guided in the process of propagating the schema changes, only requiring his assistance in case of ambiguity or inconsistency in the set of given correspondences.

To propagate a semantic modification, we first transform the respective semantic correspondence into appropriate structural modifications on the local schemas and a set of equivalence correspondences. Then we identify the subschemas affected by the semantic correspondence (deriving the equivalence correspondences that originate the affected subschemas), and finally we re-integrate using the derived equivalence correspondences plus those obtained by the transformation of the semantic correspondences [14].

One issue that still remains open is that of updating. Future work will consider the update problem, with the corresponding definition of a *write semantics*. Another

promising direction for further research is to study the application of our techniques to evolutionary environments, such as electronic commerce or data warehouses built from Internet data repositories. In [10] and [4], we report experiences in this direction.

References

1. M. Castellanos. View mechanism for schema evolution in object oriented DBMS. In *14th British National Conference on Databases (BNCOD14)*, July 1996.
2. L. Deruelle, G. Goncalves, and J.C. Nicolas. Local and Federated Database Schemas Evolution. An Impact propagation Model. In *Databases and Expert Systems Applications (DEXA)*, 1999.
3. L. Deruelle, G. Goncalves, and J.C. Nicolas. A change impact analysis approach for corba-based feerated databases. . In *Databases and Expert Systems Applications (DEXA)*, 2000.
4. A. do Carmo and R. Motz. Propuesta para integrar bases de datos que contienen información de la Web. In *Proceedings of the 4to. Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software (IDEAS'2001), Costa Rica*, April 2001.
5. K. Claypool et. al. SERF: A Schema Evolution through an extensible, re-usable and flexible framework. In *CIMK98*, 1998.
6. Peter Fankhauser. *Methodology for Knowledge-Based Schema Integration*. PhD thesis, University of Vienna, Austria, December 1997.
7. Richard Hull. Relative Information Capacity of Simple Relational Database Schemata. *SIAM Journal of Computation*, 15(3), August 1986.
8. Anthony Kosky, Susan Davidson, and Peter Bunemann. Semantics of Database Transformations. Technical report, MS-CIS-95-25, University of Pennsylvania, USA, 1995.
9. X. Li. A survey of schema evolution in object-oriented databases. In *Technology of Object Oriented Languages and Systems, IEEE Comp. Soc., Los Alamitos, CA, USA*, 1999.
10. A. Marotta, R. Motz, and R. Ruggia. Managing Source Schema Evolution in Web Warehouses. *Journal of the Brazilian Computer Society, Special Issue on Information Integration on the Web.*, 8(2), November 2002.
11. P. Mc.Brien and A. Poulovassilis. Schema Evolution in Heterogeneous Database Architectures, A Schema Transformation Approach. In A. Banks Pidduck et. al., editor, *14th International Conference on Advanced Information Systems Engineering (CAiSE 2002), Lectures Notes in Computer Science vol. 2348*, pages 484–499. Springer Verlag, 2002.
12. R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan. The use of Information Capacity in Schema Integration and Translation. In *Proc. of the 19th. VLDB Conf. Dublin, Ireland.*, 1993.
13. R.J. Miller, M.A. Hernández, L.M. Haas, L. Yan, C.T.H. Ho, R. Fagin, and L. Popa. The Clio Project: Managing Heterogeneity. *SIGMOD Record*, 30(1), March 2001.
14. Regina Motz. *Dynamic Maintenance of an Integrated Schema*. PhD thesis, Darmstadt University of Technology, January 2004.
15. N. Pittas, A. C. Jones, and W. A. Gray. Conceptual Concistency Management in Multidatabases: The FEvoS Framework. In *18th British National Conference on Databases (BNCOD18), Oxfordshire*, July 2001.
16. Martin Schönhoff, Markus Strässler, and Klaus R. Dittrich. Version Propagation in Federated Database Systems. In *Proc. Int. Databases and Applications Symposium (IDEAS'01), Grenoble, France*, July 2001.