Problems in the Maintenance of a Federated Database Schema

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

Abstract

In this paper, we characterize the problem of maintenance of a federated schema to cope with local schema evolution in a tightled coupled federation. By means of an example, we present the problems that local schema changes could cause on the federated schema and show proposed solutions.

1. Introduction

For many organizations, *federated databases* [14] have become the preferred strategy for reconciling the proliferation of private and independent databases, and for increasing the productivity of their information technology investment. A federated database, or *federation* for short, consists of a collection of possible heterogeneous, interoperating but autonomous component databases. A key characteristic of a federation is the preservation of local database autonomy, in the sense that the component databases can not be modified for the purpose of integration, and their instances and schemas may evolve, but independently.

One approach to building federated databases is by building a "tightly-coupled" federated schema [5]. A critical task in this approach is that of schema integration. Schema integration is the process by which overlapping information represented in heterogeneous forms within the component databases of a cooperative environment is unified. However the few techniques on object oriented schema evolution that have been developed focus on centralized database management systems, and have not characterized the result they should produce in the federated case with exception of evolution of views [2, 9] and the use of knowledge database or metamodelling [10, 4, 12].

In this paper, we characterize the problem of maintenance of a federated schema to cope with local schema evolution in a tightled coupled federation and present a directly approach, without the use of views or metamodelling, for the maintenance of the federated schema.

The remainder of the paper is organized as follows. Section 2 presents a motivating example the paper is based on. In Section 3 we review the different local schema changes and show, by means of our running example, the problems that local schema changes could cause on the federated schema. Section 4 presents our approach to solve the problem of maintenance of a federated schema Section 5 concludes the paper giving some final remarks.

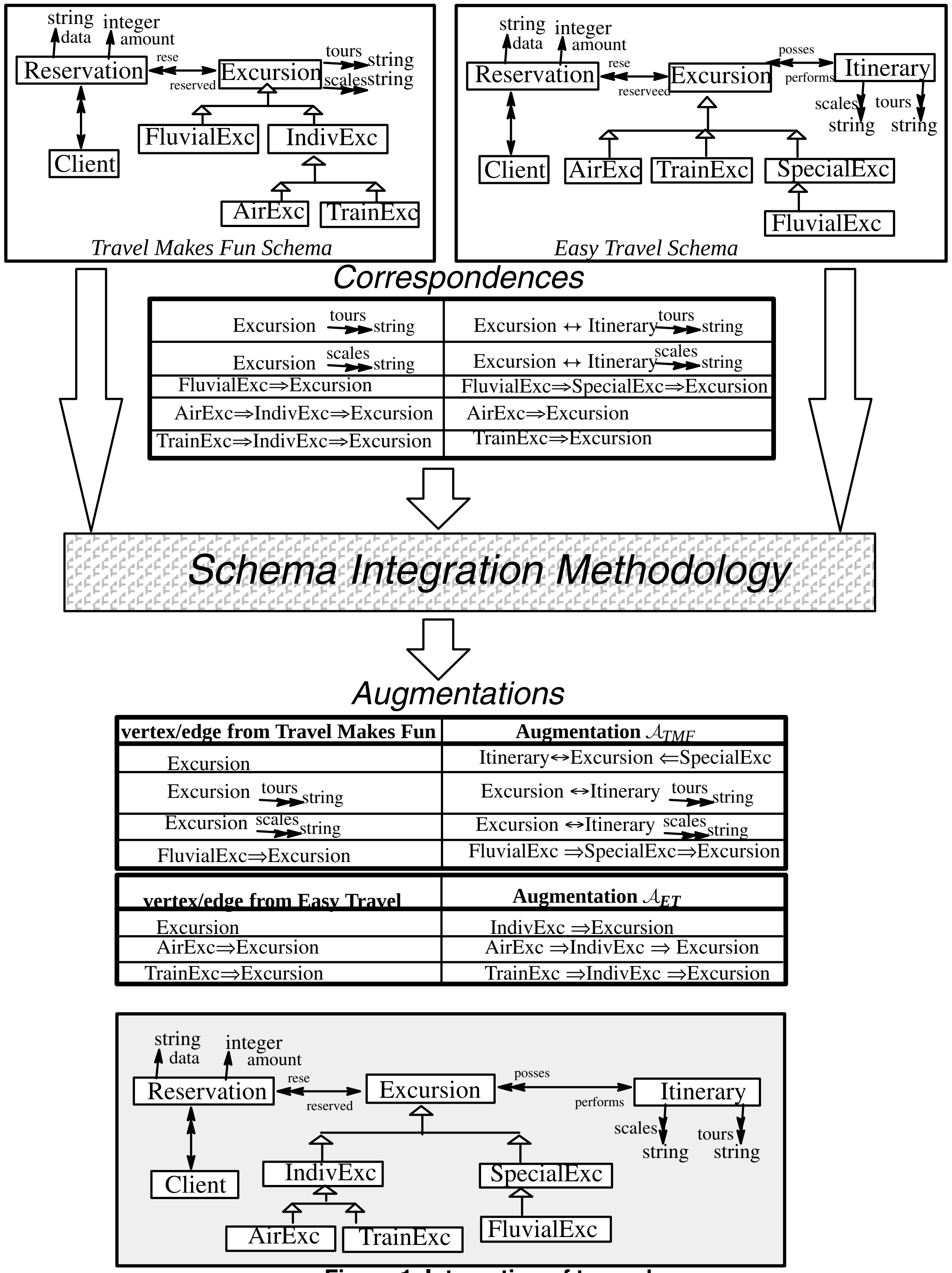
2. A Motivating Example

In this section, we present a running example and briefly show how a federated schema is generated. As our running example we consider the application depicted in Figure 1.

The example model the databases of two travel agencies, Travel Makes Fun and Easy Travel, which offer excursions. The domain of expertise of each travel agency is different. Easy Travel mainly concentrates on young people, with emphasis on individual excursions, whereas Travel Makes Fun concentrates on business persons, with emphasis on a special kind of excursions. Both agencies have decided to cooperate with each other offering excursions in an integrated way, while maintaining independence in their specialized domains. To this end, their databases have been coupled in a federation that maintains a global integrated database.

The Federated Schema

Apart from modelling their own specialized real world portions, the schemas of both databases share some common situations which are modelled with different granularity. In both travel agencies, excursions follow predefined itineraries. Each itinerary consists of a set of tours and a set of scales. While the Easy Travel schema models itineraries as objects, in the Travel Makes Fun schema they are given by the values of the attributes *tours* and *scales* that belong to the class Excursion. Excursions need to be reserved in



Excursion $\xrightarrow{tours}$ string	Excursion ↔ Itinerary $\xrightarrow{tours}$ string
Excursion $\xrightarrow{scales}$ string	Excursion ↔ Itinerary $\xrightarrow{scales}$ string
FluvialExc⇒Excursion	FluvialExc⇒SpecialExc⇒Excursion
AirExc⇒IndivExc⇒Excursion	AirExc⇒Excursion
TrainExc⇒IndivExc⇒Excursion	TrainExc⇒Excursion

vertex/edge from Travel Makes Fun	**Augmentation $\mathcal{A}_{TMF}$**
Excursion	Itinerary↔Excursion ⇐SpecialExc
Excursion $\xrightarrow{tours}$ string	Excursion ↔Itinerary $\xrightarrow{tours}$ string
Excursion $\xrightarrow{scales}$ string	Excursion ↔Itinerary $\xrightarrow{scales}$ string
FluvialExc⇒Excursion	FluvialExc ⇒SpecialExc⇒Excursion

vertex/edge from Easy Travel	**Augmentation $\mathcal{A}_{ET}$**
Excursion	IndivExc ⇒Excursion
AirExc⇒Excursion	AirExc ⇒IndivExc ⇒ Excursion
TrainExc⇒Excursion	TrainExc ⇒IndivExc ⇒Excursion

Figure 1. Integration of two schemas.

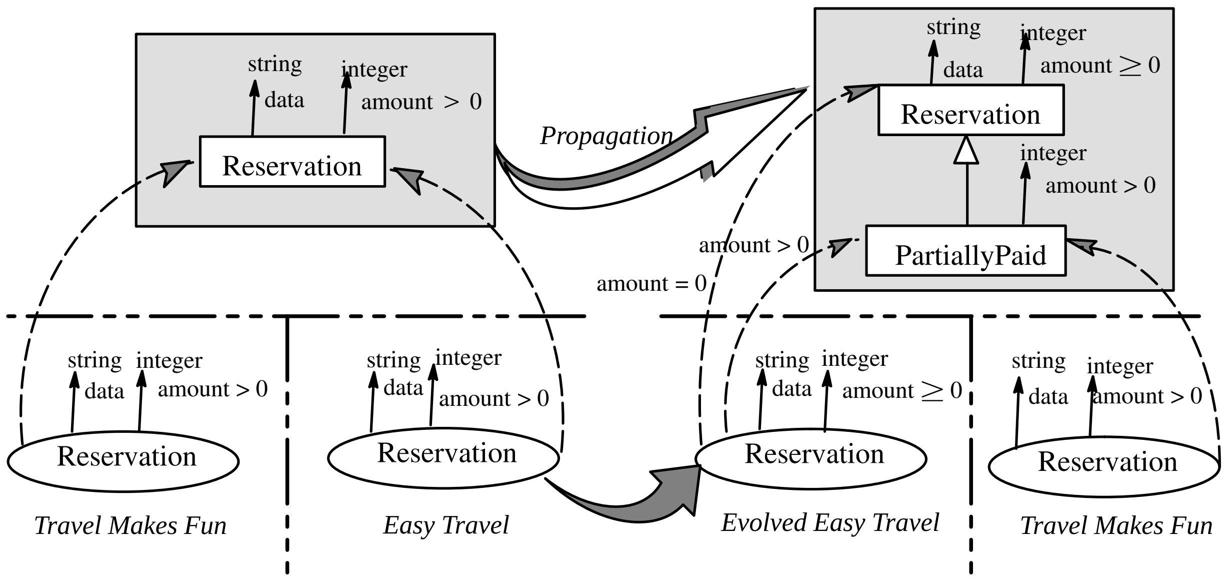


Figure 2. Dynamic Maintenance when Semantic Modifications.

advance. In both schemas this fact is represented by a class Reservations with attributes: *date* of the reservation and *amount* already paid. The specialization hierarchies for excursions are expressed at different levels of abstraction in each schema.

To overcome the representation differences of the same real world portions, the integration process needs to transform both schemas, so that the representations of the common situations are made equal. Declarative schema integration methodologies require as input the source schemas and a set of equivalence correspondences between portions of the schemas to be integrated. Suppose that the schema correspondences for our running example are those listed in Figure 1. The table of correspondences is formed by two columns. Each entry stands for an equivalence correspondence between a subschema coming from Travel Makes Fun (left column) and one coming from Easy Travel (right column).

For simplicity reasons, we assume that elements with equal names in both schemas correspond with each other. Using these correspondences, the schema integration methodology derives the mappings $\mathcal{A}_{TMF}$ and $\mathcal{A}_{ET}$ (for Travel Makes Fun and Easy Travel, respectively), listed in Figure 1. They map the corresponding local schema to the richer integrated one. Each entry of an augmentation table specifies the subschema of the integrated schema which a vertex/edge of the respective local schema is mapped to.

3. Maintenance of a Federated Schema

The aim of this section is to illustrate, by means of an example, the impact that local schema changes may produce on an already integrated schema. Such changes could be automatically determined by comparing the local schema with the evolved one, for example. Work in this direction has been performed by Ambite and Knoblock [1] and Goñi *et al.* [7]. Therefore, we assume that the specification of the local modifications is supplied as input.

We consider two types of schema changes: semantic and structural modifications.

Semantic Modifications: They refer to changes in the semantic properties of the local schemas. For instance, changes in the (semantic) properties of a class. The occurrence of such modifications may in turn affect the semantics of the existing set of correspondences between the local schemas. Consider the corresponding classes Reservation (Easy Travel) and Reservation (Travel Makes Fun). Suppose that, initially, there was a constraint in the definition of both classes which stated that reservations could only be carried out with advance the payment of an amount (i.e. the payment of a strictly positive quantity). However, after some time the Easy Travel agency changes its policy allowing reservations without any payment. This modification in the semantics of Reservation (Easy Travel) must have the following effect on the integrated schema (see Figure 2): A new class PartiallyPaid, subclass of Reservation, must be added to the integrated schema. This class models those reservations that have been made after paying

Travel Makes Fun	Easy Travel
Excursion $\overset{scales}{\dashrightarrow}$ string	Excursion $\leftrightarrow$ Itinerary $\overset{scales}{\dashrightarrow}$ string
Excursion $\leftrightarrow$ Tour $\overset{tours}{\dashrightarrow}$ string	Excursion $\leftrightarrow$ Itineray $\overset{tours}{\dashrightarrow}$ string

Table 1. New correspondences.

Travel Makes Fun	Easy Travel
Excursion $\leftrightarrow$ Tour $\overset{scales}{\dashrightarrow}$ string	Excursion $\leftrightarrow$ Itinerary $\overset{scales}{\dashrightarrow}$ string
Excursion $\leftrightarrow$ Tour $\overset{tours}{\dashrightarrow}$ string	Excursion $\leftrightarrow$ Itineray $\overset{tours}{\dashrightarrow}$ string

Table 2. New Correspondences (ambiguity case).

some amount. The class Reservation, in contrast, permits to do reservations without initial payment. That is, the definition of Reservation as a generalization of PartiallyPaid models the inclusion relationship between the extensions of the classes Reservation (Travel Makes Fun) and Reservation (Easy Travel). Note that as a consequence of local schema changes, modifications might occur in some of the correspondences between subschemas that led to the existing integrated schema.

Structural Modifications: These are modifications to the structure of local schemas. Consider the attribute *tours* of the class Excursion (Travel Makes Fun) and suppose that it is transformed into a new created class Tour (with an attribute *tours*) which holds a relationship with Excursion. As a consequence of this modification the set of correspondences between both local schemas is also affected. After the modification the correspondences depicted in Table 1 hold. The first correspondence is the same as before, while the second reflects the appearance of the new class Tours. The new integrated schema is depicted in Figure 3.

Intuitively, it is the result of propagating the local structural modification to the corresponding correspondence. However, the problem is to find the original correspondences in the integrated schema. Moreover,there are some cases in which it is not possible to automatically obtain an evolved integrated schema due to the occurrence of ambiguity or inconsistency.

Modifications that lead to ambiguity: Consider the attributes *tours* and *scales* of the class Excursion (Travel Makes Fun) and suppose that they are transformed into a new created class Tour (also with attributes *tours* and *scales*), which holds a relationship with Excursion. The new correspondences are depicted in Table 2.
Traditional schema integration methodologies can not automatically propagate this change due the following ambiguity: Is the class Tour corresponding to the class Itinerary or not? In order to solve this ambiguity, the interaction with the user is needed.

Modifications that lead to inconsistency: Local schema changes may cause inconsistencies, i.e. situations in schemas that cannot be simultaneously satisfied in the integrated schema. Consider, for instance, the evolution of Easy Travel presented in Figure 4, where the class TrainExc is moved through the hierarchy of classes to appear as a subclass of the class SpecialExc. In this case there is no possible propagation of the change because of an inconsistency between the correspondences that represent the modification and the initial ones, see Figure 4. The reason of the inconsistency is that for the new set of correspondences:

$$\begin{aligned} &\mathsf{Excursion} \Leftarrow \mathsf{IndivExc} \Leftarrow \mathsf{AirExc} \\ &\quad\equiv \mathsf{Excursion} \Leftarrow \mathsf{AirExc} \qquad (1) \\ &\mathsf{Excursion} \Leftarrow \mathsf{IndivExc} \Leftarrow \mathsf{TrainExc} \\ &\quad\equiv \mathsf{Excursion} \Leftarrow \mathsf{SpecialExc} \Leftarrow \mathsf{TrainExc} \qquad (2) \\ &\mathsf{Excursion} \Leftarrow \mathsf{FluvialExc} \\ &\quad\equiv \mathsf{Excursion} \Leftarrow \mathsf{SpecialExc} \Leftarrow \mathsf{FluvialExc} \qquad (3) \end{aligned}$$

there is no possibility to have an integrated schema that fulfills all of them simultaneously. For example, to achieve an integrated schema it is sufficient that the paths $\mathsf{Excursion} \Leftarrow \mathsf{IndivExc} \Leftarrow \mathsf{TrainExc}$ and $\mathsf{Excursion} \Leftarrow \mathsf{SpecialExc} \Leftarrow \mathsf{TrainExc}$ appear intact in the integrated schema. However, observe that correspondence (2) indicates that these two paths must be unified. Furthermore, no equivalence correspondence between IndivExc and SpecialExc may hold. In fact, an integration based on this correspondence would introduce a direct relation between the classes FluvialExc and IndivExc in the integrated schema, as the path $\mathsf{Excursion} \Leftarrow \mathsf{FluvialExc}$ from Travel Makes Fun would be transformed into $\mathsf{Excursion} \Leftarrow \mathsf{IndivExc} \Leftarrow \mathsf{FluvialExc}$, and

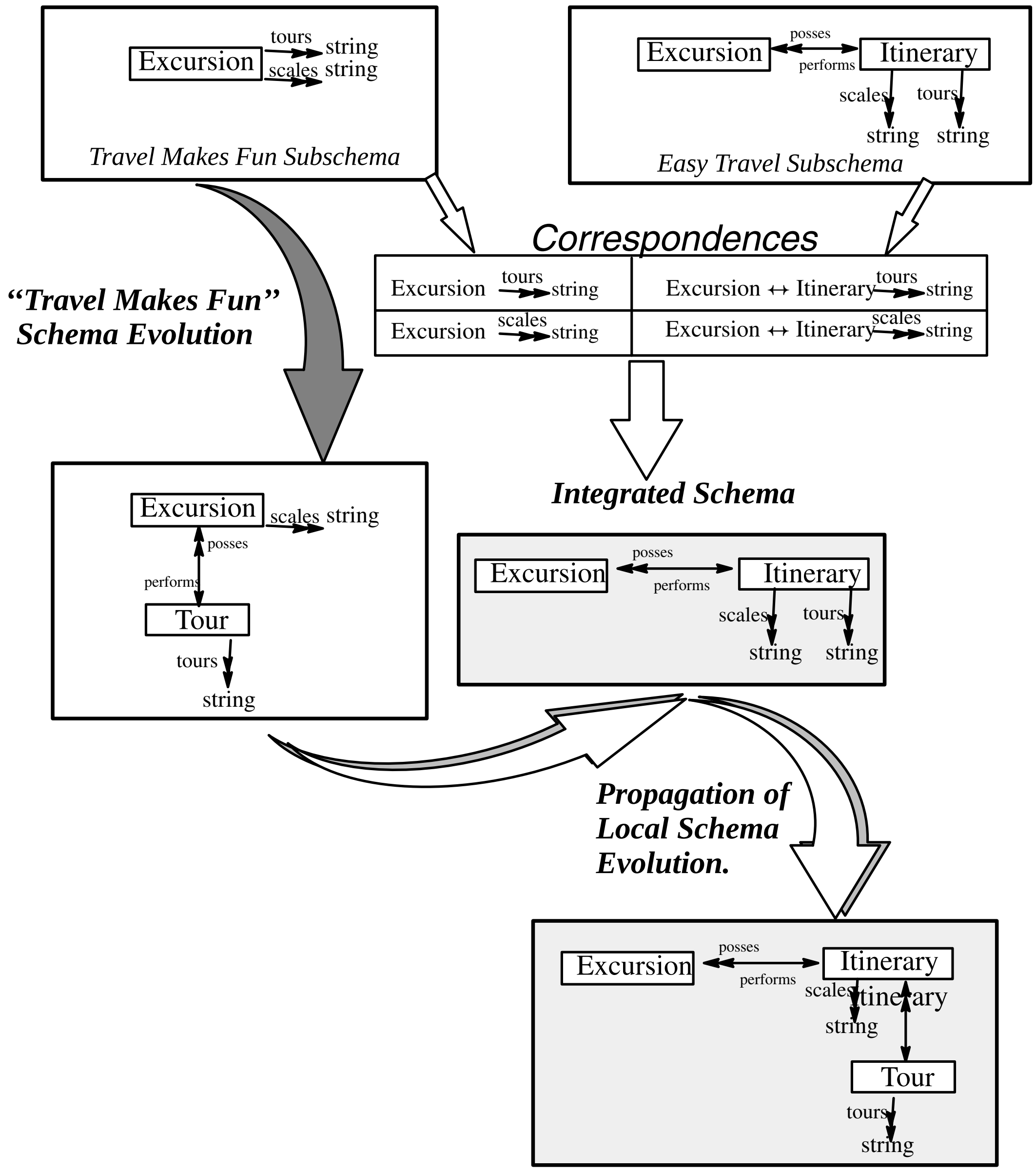


Figure 3. Dynamic Maintenance when Structural Modifications.

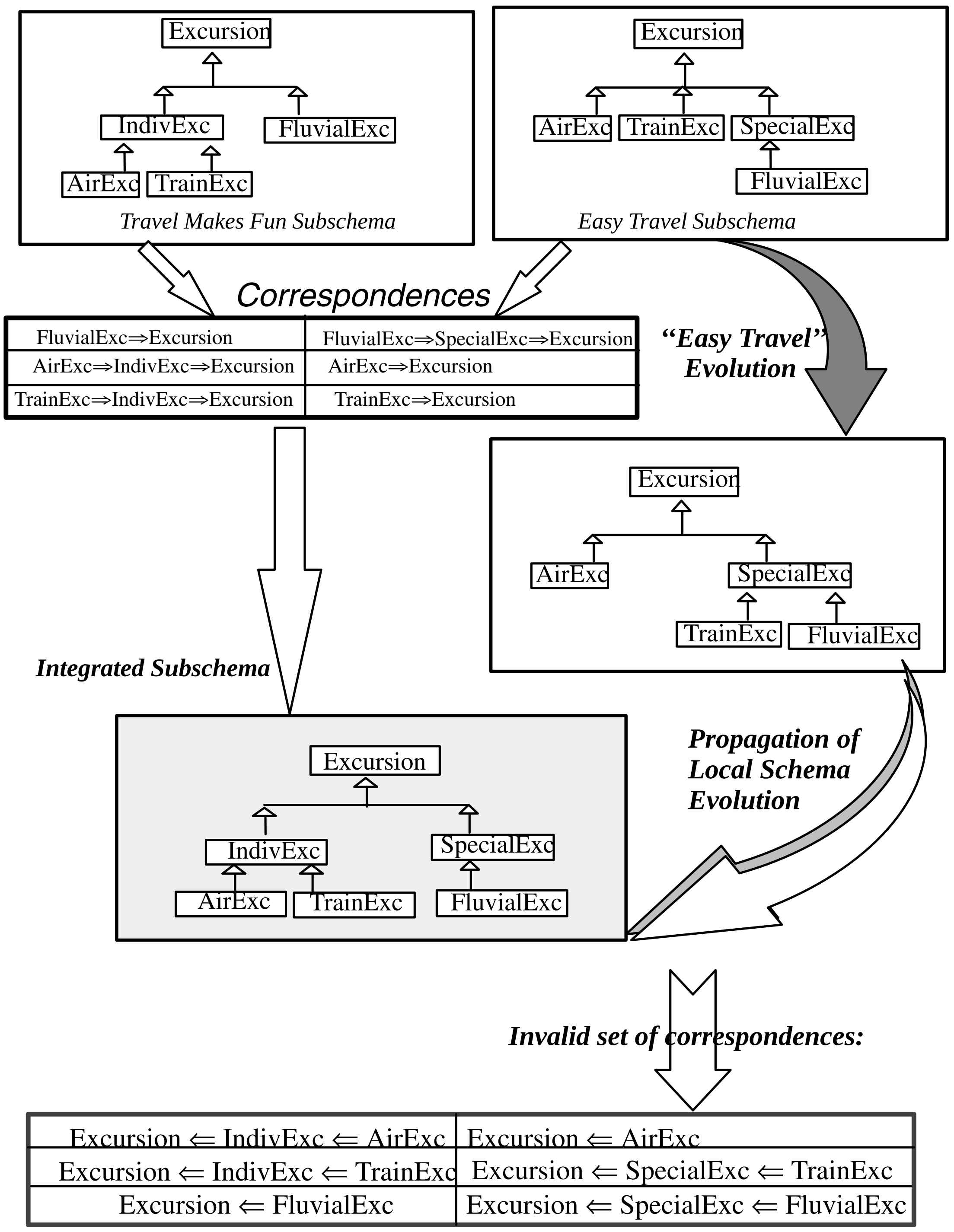


Figure 4. Structural Modification that leads to inconsistency.

this is incompatibe with the semantics these classes possess in Travel Makes Fun.

Some powerfull schema integration methodologies (i.e. [6]), are able to warn the user about the inconsistency problem but are not powerfull enough to present the invalid correspondences in order to dismiss one of them.

4. Our Approach

A detailed examination of local schema evolution, as the cases depicted in the previous section, shows us that a great amount of schema changes may also be expressed by means of correspondences between local schemas. However, a particular challenge of our approach to maintenance of an integrated schema is the fact that a local schema evolution may invalidate correspondences established in some previous integration steps. Moreover, we aim at an *efficient* propagation of local schema evolution to an integrated schema. This means that we would like to avoid as much as possible reintegration steps in the propagation process.

We observe that, in general, the propagation of local structural modifications to the integrated schema can be accomplished by adopting one of the following solutions:

1. Re-integrate from the beginning using the whole set of schema correspondences; or

2. Identify the augmentations affected by the evolution and modify them directly without any new integration step.

Solution (1) amounts to a brute force strategy, as it implies to redo the whole integration process each time a local schema change is produced. Solution (2) is clearly the most desirable solution, as it does no imply any extra integration step. However, it is applicable only in case of modifications that do not generate inconsistent or ambiguous correspondences.

We adopt a combination of both solutions. We identify the augmentations affected by the local structural modification and either modify them directly or deduce from them the involved corresponding subschemas. Then, re-integrate only on these subschemas.
This solution reduces the number of correspondences used for re-integration, since only those correspondences deducible from the affected schema augmentations are considered.
The crucial point is then how to identify the augmentations that an evolution affects and then deduce from them the affected overlapping subschemas (which are presented in the form of a set of path correspondences). In other words, we regard the process of propagating local schema changes as a form of *incremental schema integration*.

Investigation on this approach requires a theoretical framework. In this sense we choose an existing schema integration methodology as the universe of discourse. Our work is based on a schema integration methodology, henceforth called SIM, developed by Fankhauser [6], which presents a novel and concisely defined treatment of schema integration. SIM operates with a formal notion of schema transformations, called *schema augmentations*, which are automatically derived from a set of "equivalence correspondences" between (arbitrary) subschemas of the component databases. Schema augmentations map, without information loss and in a non-redundant manner, corresponding information between two component schemas to a unified representation in the integrated schema.

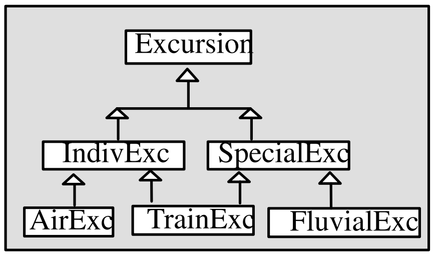
Based on this observation, we work out an extension of SIM that permits to deal with flexibility with the set of correspondences. Flexibility in the integration process is accomplished by basing the process on a complete independence of the ordering in which correspondences occur.
An essential aspect of our approach is that, from the augmentation mappings for the already acquired 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.

In case of ambiguity, all possible alternatives are presented to the user in order to select or dismiss one. Our approach is to adopt the complete schema level as the granularity of change, allowing changes to involve multiple classes or relationships in the local schemas. This is achieved by regarding the propagation of local schema evolutions from an integration point of view.

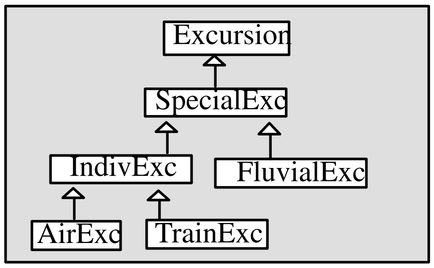
In case of inconsistency, the user is presented with the minimal set of correspondences that produce the inconsistency. Our approach is to make SIM incremental. That is, correspondences between the local schemas can be added or deleted at any time. In this sense, we provide SIM with a procedure able to treat ambiguities or inconsistencies incrementally. Essentially, such a procedure consists of two phases: (i) it identifies the subschema of the integrated schema affected by the given correspondence, and (ii) it treats this subschema so that it becomes consistent.

Figure 5 shows the integrated schema that results from each set of accepted correspondences presented to the user

Excursion⇐ IndivExc⇐ AirExc	Excursion ⇐ AirExc
Excursion⇐FluvialExc	Excursion⇐SpecialExc⇐FluvialExc



Excursion⇐ IndivExc⇐ AirExc	Excursion ⇐ AirExc
Excursion⇐IndivExc⇐TrainExc	Excursion ⇐ SpecialExc⇐ TrainExc



Excursion⇐ IndivExc⇐ AirExc	Excursion ⇐ AirExc
Excursion⇐IndivExc⇐TrainExc	Excursion ⇐ SpecialExc⇐ TrainExc

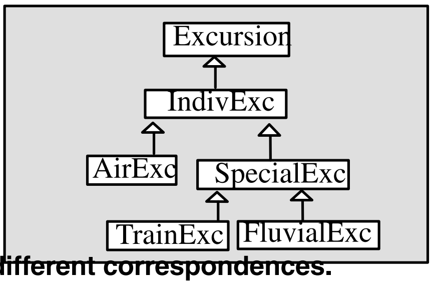


Figure 5. Integrated Schema for different correspondences.

for the inconsistent example depicted in Figure 4.

Apart from using SIM, our methodology performs the identification of the local subschemas affected by the local modification in order to derive the affected correspondences. Essentially, this process consists of walking through the augmentation corresponding to the schema augmentations that refer to the schema change (either as argument or as part of its result). This can be done because augmentations are injective functions. Once the schema correspondences are identified, the associated augmentations are replaced by the trivial ones, preserving that way schema consistency. Then, new correspondences are derived to cope with the schema change and re-integration on these new correspondences is eventually performed by using SIM.

5. Final Remarks

One of the main difficulties associated with tightly coupled federated databases is to mantain the federation in a consistent state after its initial integration. This is due to the considerable effort required for detecting and reconciling discrepancies that arise as a result of the evolution of the local sources [13, 4, 12]. The structure of the federated schema depends directly on the correspondences between the local schemas and the integration method used [3]. Structural or semantic schema changes in the local schemas may affect some of the correspondences between the local subschemas that led to the existing integrated schema. Therefore, the problem is how to translate the schema changes to the integrated schema without the necessity of re-integrating all from the beginning.

While capturing local schema modifications is something that we are not aiming to automate, our goal has been, at the first phase, the semi-automatic propagation of local structural modifications to the integrated schema. This is achieved by a procedure that permits the automatic identification of subschemas affected by local structural modifications and their semi-automatic reconciliation. Local structural modification are regarded as a *new* set of correspondences between local schemas, which are then used for re-integration. This issue is addressed by extended a schema integration methodology (SIM) with incremetal capabilities.

At the moment, we have a prototype, developed in Java, that recognizes the minimal set of inconsistent correspondences [8]. We plan to extend this implementation in order to incorporate an undo mechanism on augmentations so that the corresponding affected subschemas become consistent. The specification of such a mechanism can be found in [11].

References

[1] J. L. Ambite and C. A. Knoblock. Reconciling Distributed Information Sources. In *Working Notes of the AAAI Spring Symposium on Information Gathering in Distributed Heterogeneous Environments*, Palo Alto, CA, 1995.

[2] Z. Bellahsene. Extending a View Mechanism to Support Schema Evolution in Federated Database Systems. In A. Hameurlain and A. M. Tjoa, editors, *8th. Int. Conf. on Database and Expert Systems Applications (DEXA)*, Toulouse, France, September 1997. Springer Verlag, Lectures Notes in Computer Science Nro. 1308.

[3] S. Busse, R.-D. Kutsche, and U. Leser. Strategies for the Conceptual Design of Federated Information Systems . In *Proc. of the Engeneering Federated Information Systems (EFIS 2000),Berlin*, 2000.

[4] S. Busse and C. Pons. Schema Evolution in Federated Information Systems . In *A. Heuer, F. Leymann, D. Priebe (eds.) Datenbanksysteme in Büro, Technik und Wissenschaft (BTW2001)*. Springer, 2001.

[5] U. Dayal and H. Hwang. View definition and Generalization for Database Integration in Multibase: a System for Heterogeneous Distributed Databases. *IEEE Transactions on Software Engineering*, 10(6), 1984.

[6] P. Fankauser. *Methodology for Knowledge-Based Schema Integration*. PhD thesis, University of Vienna, Austria, December 1997.

[7] A. Goñi, A. Illarramendi, E. Mena, and J. M. Blanco. Monitoring the Evolution of Databases in Federated Relational Database Systems. In *CAiSE'97 Workshop on Engineering Federated database Systems (EFDBS'97)*, June 1997.

[8] C. Herrera, S. Capretti, and F. Jacques. Una herramienta para la construccin de una base de datos generada a partir de datos extrados de la Web. Graduate Proyect, InCo, Facultad de Ingenieria, UdelaR,Montevideo, Uruguay, 2001.

[9] R. Kaushal and B. Eaglestone. View-Based Support for Transparent Schema Evolution in Federated Database Systems. In *Int. Workshop on Issues and Applications of Database Technology, Berlin, Germany*, July 1998.

[10] S. Kolmschlag and G. Engels. Unterstützung der Flexibilität eines Ele Workshop Integration heterogener Softwaresysteme (IHS98) im Rahmen der GI-Jahrestagung Informatik ' 98, Magdeburctronic Commerce Systems durch Evolutionstechniken. In *Workshop Integration heterogener Softwaresysteme (IHS98) im Rahmen der GI-Jahrestagung Informatik ' 98, Magdeburg*, 1998.

[11] R. Motz. Dynamic Maintenance of an Integrated Schema. In *PhD Thesis, To forthcomming*, 2002.

[12] N. Pittas, A. C. Jones, and W. A. Gray. Metadata Exploitation in Support of Federated Database Systems Evolution . In *Proc. of the Engeneering Federated Information Systems (EFIS 2001),Berlin*, 2001.

[13] M. Roantree, W. Hasselbring, and S. Conrad. Proc. of the Engineering Federated Information Systems (EFIS'2000).). In *IOS Press (ISBN 1-58603-075-2)*, 1998.

[14] A. Sheth and J. Larson. Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases. *ACM Computing Surveys*, 22(3), 1990.