Flexible Database Transformations: The SERF Approach *

Kajal T. Claypool and Elke A. Rundensteiner
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01609–2280
{kajal|rundenst}@cs.wpi.edu

Abstract

Database transformation is a critical task that occurs in many different domains. Schema evolution is one important class of problems for database transformations. In our work, we use existing technology and standards (ODMG, OQL, basic schema evolution primitives) to bring flexibility, extensibility and re-usability to current schema evolution systems, thus allowing the users to conveniently specify any customized transformation of their choice. We also investigate the re-usabilty of our framework to other applications beyond schema evolution such as web re-structuring.

Keywords: Schema Evolution, Transformation Templates, Object-Oriented Databases,Modeling Database Dynamics, OQL, ODMG, Schema Consistency.

1 Introduction

The age of information management and with it the advent of increasingly sophisticated technologies have kindled a need in the database community and others to *transform* existing systems and move forward to make use of these new technologies. Legacy application systems are being transformed to newer state-of-the-art systems, information sources are being mapped from one data model to another, a diversity of data sources are being transformed to load, cleanse and consolidate data into modern data-warehouses.

One important class of data transformations are schema evolution tools that do on-line transformation of database systems by modifying both the schema as well as the underlying data objects without bringing the system down [Zic92]. For this, most object-oriented database systems (OODB) today support a *pre-defined* taxonomy of *simple fixed-semantic* schema evolution operations [BKKK87, Tec94, BMO^{+}89, Inc93, Obj93]. More advanced changes such as combining two types have also recently been looked at by Breche [Bré96] and Lerner [Ler96], but are still limited to being a *fixed* set. Anything beyond the *fixed* taxonomy often requires application users to

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*This work was supported in part by the NSF NYI grant #IRI 94-57609. We would also like to thank our industrial sponsors, in particular, IBM for the IBM partnership award and Informix for software contribution. Special thanks also goes to the PSE Team specifically, Gordon Landis, Sam Haradhvala, Pat O'Brien and Breman Thuraising at Object Design Inc. for not only software contributions but also for providing us with a customized patch of the PSE Pro2.0 system that exposed schema-related APIs needed to develop our tool.

write ad-hoc programs to accomplish such transformations. Such programs are very specific and in general cannot be shared across applications and since there is no system-level support for maintaining the consistency of the system, they are more prone to errors. To address these limitations of the *current* transformation technology, we have proposed the SERF framework which aims at providing a rich environment for doing complex user-defined transformations *flexibly*, *easily* and *correctly* [CJR98b]. In this paper we give an overview of the SERF framework, its current status and the enhancements that are planned for the future. We also present an example of the application of SERF to a domain other than schema evolution, i.e., the web re-structuring.

The rest of the paper is organized as follows. Section 2 gives an overview of the key concepts of the SERF Framework. Section 3 discusses some of the more advanced features which are now being added on to SERF to increase the usability and dependability of the SERF system. Section 3.3 outlines the application of SERF to other domains. We conclude in Section 4.

2 The Basic SERF Framework

The SERF framework addresses the limitation of current OODB technology that restricts schema evolution to a *predefined* set of simple schema evolution operations with *fixed* semantics [BKKK87, Tec94, BMO$^+$89, Inc93, Obj93]. With the SERF framework we can offer *arbitrary user-customized* and possibly *very complex* schema evolution operations such as *merge*, *inline* and *split* [Ler96, Bré96] without users having to resort to writing ad-hoc code. Moreover, for each transformation type there can be many different semantics based on user preferences and application needs. For example two classes can be merged by doing a union, an intersection or a difference of their attributes. Similarly the deletion of a class that has a super-class and several sub-classes can be done by either propagating the delete of the inherited attributes through all sub-classes, or by moving the attributes up to the super-class, or by moving them down to all the sub-classes, or any composition of the above.

Our approach is based on the hypothesis that complex schema evolution transformations can be broken down into a sequence of basic evolution primitives, where each basic primitive is an invariant-preserving atomic operation with fixed semantics provided by the underlying OODB system. In order to effectively combine these primitives and to be able to perform arbitrary transformations on objects within a complex transformation, we rely on a standard query language namely OQL [Cea97]. The alternative approach to define a new language for specifying the database transformations has been explored in the literature [DK97] (also see this issue). In our work, we demonstrate that a language such as OQL is sufficient for accomplishing schema evolution.

We illustrate the steps involved in a schema evolution transformation using the example of *Inlining* which is defined as the replacement of a referenced type with its type definition [Ler96]. For example in Figure 1 the `Address` type is inlined into the `Person` class, i.e., all attributes defined for the `Address` type (the referenced type) are now added to the `Person` type resulting in a more complex `Person` type. Figure 2 shows the *Inline* transformation expressed in our framework using OQL, schema modification primitives such as *add_attribute()*, and system-defined update methods such as *obj.set()*.

In general in a SERF transformation there are three types of steps:

- **Step A: Change the Schema.** We require that all structural changes, i.e., changes to the schema, are exclusively made through the schema evolution primitives. This helps us guarantee schema consistency after the application of a transformation [CJR98b]. For example, **Step A** in Figure 2 shows the addition of the attributes `Street, City` and `State` via the *add_attribute* schema evolution (SE) primitive to the `Person` class.

- **Step B: Query the Objects.** Prior to performing object transformations, we need to must obtain the handle for objects involved in the transformation process. This may be objects from which we copy object values (e.g., `Address` objects in **Step B**), or objects that get modified themselves (e.g., `Person` objects in **Step C**).

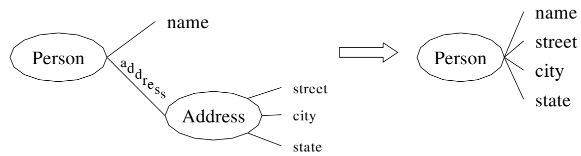

Figure 1: Example of an Inline Transformation.

```
// Add the required attributes to the Person class
add_attribute (Person, Street, String," ");
add_attribute (Person, City, String," ");                    Step A
add_attribute (Person, State, String," ");

// Get all the objects for the Person class
define extents() as                                          Step B
   select c
   from Person c;

// Update all the objects
for all obj in extents():                                    Step C
   obj.set (obj.Street, valueOf(obj.address.Street)) AND
   obj.set (obj.City, valueOf(obj.address.City))     AND
   obj.set (obj.State, valueOf(obj.address.State))

// Delete the address attribute                              Step B
delete_attribute (Person, address);
```

Figure 2: Inline Transformation Expressed in OQL with Embedded Evolution Primitives.

- **Step C: Change the Objects.** The next step to any transformation logically is the transformation of the objects to conform to the new schema. Through **Step B**, we already have a handle to the affected object set. **Step C** in Figure 2 shows how a query language like OQL and system-defined update methods, like *obj.set*(...), can be used to perform object transformations.

The transformation uses the query language to invoke the schema evolution primitives for schema changes and the system-defined functions for object updates, as in **Steps A** and **C**. Thus we require the capability to invoke method calls as part of a query specification, which is indeed supported by OQL [Cea97].

SERF Template. A SERF transformation as given in Figure 2 *flexibly* allows a user to define different semantics for any type of schema transformation. However, these transformations are *not re-usable* across different classes or different schemas. For example, the *inline* transformation shown in Figure 2 is valid only for the classes `Person` and `Address`. To address this, we have introduced the notion of templates in the SERF framework [CJR98b]. A template uses the query language's ability to query over the meta data (as stated in the ODMG Standard) and is enhanced by a name and a set of parameters to make transformations *generic* and *re-usable*. Figure 3 shows a templated form of the transformation presented in Figure 2. The section of the template marked **Step D** shows the steps required to achieve the effect of **Step A** in Figure 2 in a general form. Thus when this inline template shown in Figure 3 is instantiated with the variables `Person` and `address` it results in the SERF transformation in Figure 2. A template is thus an arbitrarily complex transformation that has been encapsulated and generalized via the use of the ODMG Schema Repository, a name and a set of parameters.

SERF Template Library. The SERF templates can be collected into a **template library** which in turn can be grouped in many different ways, for example by domain such as templates for doing data cleansing, or by object model such as templates for the graph or the web model, thus providing a valuable *plug-and-play* resource to the transformation community. Our overall goal is thus to provide SERF as a value-added service layer on top of existing database systems as the template library can ideally be plugged in for any SERF system.

```
begin template inline (className, refAttrName)
{

  refClass = element (
               select a.attrType
               from MetaAttribute a
               where a.attrName = $refAttrName
               and   a.classDefinedIn = $className; )

   define localAttrs(cName) as
      select c.localAttrList
      from MetaClass c
      where c.metaClassName = cName;

   // get all attributes in refAttrName and add to className
   for all attrs in localAttrs(refClass)
      add_atomic_attribute ($className, attrs.attrName,
                            attrs.attrType, attrs.attrValue);

   // get all the extent
   define extents(cName) as
      select c
      from cName c;

   // set: className.Attr = className.refAttrName.Attr
   for all obj in extents($className):
      for all Attr in localAttrs(refClass)
         obj.set (obj.Attr, valueOf(obj.refAttrName.Attr))

   delete_attribute ($className, $refAttrName);
}

end template

Legend:  cName: OQL variables
         $className: template variables
         refClass   user variables
```

```
begin template inline (className, refAttrName)
{
 REQUIRES:

   className must exist in the currentSchema AND
   refAttrName must exist in className AND
   domain(refAttrName) must exist in currentSchema AND
   className != domain(refAttrName)


      Body of Inline Template


 ENSURES:
   className exists in thisSchema AND
   refAttrName does not exist in thisSchema AND
   Properties(domain(refAttrName)) is subset of className

}
end template
```

Figure 3: Left: The Inline Template; Right: The Contractual Form of the Inline Template.

SERF System - OQL-SERF. An implementation of SERF, OQL-SERF, is currently being developed at Worcester Polytechnic Institute. It is based on the ODMG standard and uses the ODMG object model, the ODMG Schema Repository definition, as well as OQL. The system is being implemented entirely in Java and uses Object Design's Persistent Storage Engine (PSE) for Java as its back-end database [RCL+99].

3 Beyond the Base SERF System

The goal of the SERF project is now to increase usability, utility and applicability of the SERF framework to transformation problems beyond OODB evolution. We have started work on providing an assurance of consistency for the users and a semantic optimizer to improve the performance of the transformations, and have also started looking at domains beyond schema evolution for the applicability of this transformation framework.

3.1 Consistency Management

Consistency management is the definition of consistency violations, re-establishment of consistency following violations, and the meaningful manipulation of objects that are not in a consistent state. This is a key problem for complex applications in general and in the face of database transformations it is an even more critical problem. One example for the use of consistency violation detection in the SERF environment where we are dealing with an expensive transformation process is the early (prior to execution) detection of erroneous templates via the consistency manager. This would help improve performance by saving the cost of rollback.

For this purpose, we have developed a model that allows for the specification of consistency constraints for the SERF templates using the *contract* model [Mey92]. Based on this model, we are developing a consistency checker that allows us to detect not only the violation of the consistency constraints but also helps in the verification of the templates. Figure **??** shows the *Inline* template written with *contracts* in easily understandable *English*. We are in the process of developing a language for the specification of *contracts*. Within a SERF template *contracts* serve both as a vehicle for a declarative specification of the behavior of a template as well as for the specification of the constraints under which a SERF template can be applied to the underlying system. Thus, beyond the advantage of violation detection, the *contracts* give us the added advantage to now provide a more sophisticated search mechanism for templates in our libraries based on their declarative behavioral descriptions.

3.2 Semantic Optimizer

Database transformation is an extremely expensive process both in terms of time as well as system resources. A simple schema evolution operation such as *add_attribute* for a large number of objects (approx. 100,000 objects) can take on the order of hours for processing. Hence a complex operation as specified by a SERF transformation can take even longer. We thus have developed the CHOP optimizer that reduces the number of operations in a sequence of schema evolution operations and have shown it to have significant savings [CNR99]. However, since SERF templates may inter-leave OQL queries with schema evolution operations, our current CHOP techniques alone are not sufficient for their optimization. Thus, we are in the process of developing query rewriting techniques with emphasis on reducing the number of expensive method calls (in our case schema evolution primitives) in a SERF template by exploiting existing CHOP techniques. Beyond the evolution domain the time savings by these optimizations may potentially be of an even bigger advantage, for example, in doing transformations for data integration over several legacy systems,

3.3 Application of SERF to Other Problem Domains - Re-WEB

The SERF framework is directly applicable to many other domains that are volatile by nature and thus have an extensive need for re-structuring the underlying structure. In particular, the SERF system can be used for doing transformations above and beyond the *fixed* basic primitives that are provided by current systems. As an example, we have already applied the SERF framework as a tool for re-structuring web sites (this is part of the **Re-WEB tool** [CRCK98] which generates and re-structures web sites.). For this purpose, we have developed a web mapping for the direct generation of web-sites from the schema of an ODMG based database. The key of the Re-WEB tool is the application of the SERF technology to transform the underlying database to produce a diversity of views that match the desired web layout. A template library of frequent web-transformations is a distinct advantage of ReWEB to achieve for example personalized web pages in a fast and efficient manner.

This is but one of many possible applications of SERF. While some small extensions might be required for the SERF system, the key concepts are applicable to many key areas such as for creating data warehouses, for data cleansing, and for addressing schema integration problems.

4 Conclusion

The SERF framework brings to the user a general-purpose transformation framework with the advantages that have existed within some programming language environments, such as templates, libraries, consistency management, etc., but have been slow to propagate to the database arena. The SERF framework gives the users the *flexibility* to define the re-structuring semantics of their choice; the *extensibility* of defining new complex re-structuring transformations meeting specific requirements; the *generalization* of these transformations through the notion of templates; the *re-usability* of a template from within another template; the *ease* of template specification by programmers and non-programmers alike; the *soundness* of the transformations in terms of assuring schema consistency; and the *portability* of these transformations across OODBs as libraries.

An ODMG based implementation of the SERF framework, OQL-SERF [CJR98a], is currently underway at the Worcester Polytechnic Institute and the system is also being demonstrated at SIGMOD'99 [RCL+99].

Acknowledgments. The authors would like to thank students at the Database Systems Research Group (DSRG) at WPI for their interactions and feedback on this research. In particular, we would like to thank Jing Jin and Chandrakant Natarajan for their initial work on SERF. We would also like to thank Anuja Gokhale, Parag Mahalley, Swathi Subramanian, Jayesh Govindrajan, Stacia De Lima, Stacia Weiner, Xin Zhang and Ming Li for their help with the implementation of OQL-SERF.

References

References

[BKKK87] J. Banerjee, W. Kim, H. J. Kim, and H. F. Korth. Semantics and Implementation of Schema Evolution in Object-Oriented Databases. *SIGMOD*, pages 311–322, 1987.

[BMO+89] R. Bretl, D. Maier, A. Otis, J. Penney, B. Schuchardt, J. Stein, E. H. Williams, and M. Williams. The GemStone Data Management System. In *Object-Oriented Concepts, Databases and Applications*, pages 283–308. ACM Press, 1989.

[Bré96] P. Bréche. Advanced Primitives for Changing Schemas of Object Databases. In *Conference on Advanced Information Systems Engineering*, pages 476–495, 1996.

[Cea97] R.G.G. Cattell and et al. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann Publishers, Inc., 1997.

[CJR98a] K.T. Claypool, J. Jin, and E.A. Rundensteiner. OQL_SERF: An ODMG Implementation of the Template-Based Schema Evolution Framework. In *Centre for Advanced Studies Conference*, pages 108–122, November 1998.

[CJR98b] K.T. Claypool, J. Jin, and E.A. Rundensteiner. SERF: Schema Evolution through an Extensible, Re-usable and Flexible Framework. In *Int. Conf. on Information and Knowledge Management*, pages 314–321, November 1998.

[CNR99] K.T. Claypool, C. Natarajan, and E.A. Rundensteiner. CHOP: An Optimizer for Schema Evolution Sequences. Technical Report WPI-CS-TR-99-06, Worcester Polytechnic Institute, February 1999.

[CRCK98] K.T. Claypool, E. A. Rundensteiner, L. Chen, and B. Kothari. Re-usable ODMG-based Templates for Web View Generation and Restructuring. In *CIKM'98 Workshop on Web Information and Data Management (WIDM'98), Washington, D.C., Nov.6*, 1998.

[DK97] S.B. Davidson and A.S. Kosky. WOL: A Language for Database Transformations and Constraints. In *IEEE Int. Conf. on Data Engineering*, pages 55–65, 1997.

[Inc93] Itasca Systems Inc. Itasca Systems Technical Report. Technical Report TM-92-001, OODBMS Feature Checklist. Rev 1.1, Itasca Systems, Inc., December 1993.

[Ler96] B.S. Lerner. A Model for Compound Type Changes Encountered in Schema Evolution. Technical Report UM-CS-96-044, University of Massachusetts, Amherst, Computer Science Department, 1996.

[Mey92] B. Meyer. Applying "Design By Contract". *IEEE Computer*, 25(10):20–32, 1992.

[Obj93] Object Design Inc. *ObjectStore - User Guide: DML. ObjectStore Release 3.0 for UNIX Systems*. Object Design Inc., December 1993.

[RCL+99] E.A. Rundensteiner, K.T. Claypool, M. Li, L. Chen, X. Zhang, C. Natarajan, J. Jin, S. De Lima, and S. Weiner. SERF: ODMG-Based Generic Re-structuring Facility. In *Demo Session Proceedings of SIGMOD'99*, 1999.

[Tec94] O_2 Technology. *O_2 Reference Manual, Version 4.5, Release November 1994*. O_2 Technology, Versailles, France, November 1994.

[Zic92] R. Zicari. A Framework for O_2 Schema Updates. In F. Bancilhon, C. Delobel, and P. Kanellakis, editors, *Building an Object-Oriented Database System: The Story of O_2*. Morgan Kaufmann Pub., 1992.