Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1996 Proceedings

Americas Conference on Information Systems (AMCIS)

8-16-1996

A Meta-model Based Multi-Data Models Translation Methodology In Interoperable Information Systems

Kokou Yétongnon
Laboratoire Ingenierie Informatique, BP 138 - Universite de Bourgogne, kokou@khali.u-bourgogne.fr

Christophe Nicolle
Laboratoire Ingenierie Informatique, BP 138 - Universite de Bourgogne, cnicolle@khali.u-bourgogne.fr

Nadine Cullot
Laboratoire Ingenierie Informatique, BP 138 - Universite de Bourgogne, ncullot@khali.u-bourgogne.fr

A Meta-model Based Multi-Data Models Translation Methodology In Interoperable Information Systems

Kokou Yétongnon, Christophe Nicolle, Nadine Cullot
Laboratoire Ingénierie Informatique
BP 138 - Université de Bourgogne
21000 Dijon - France.
Email : {kokou,cnicolle,ncullot}@khali.u-bourgogne.fr

1. Introduction

Interoperable database systems are increasingly used for sharing and accessing information across heterogeneous databases. These systems may include a variety of data models ranging from traditional data models such as the relational, hierarchical and network to newer semantic and object oriented data models. Semantic heterogeneity, which may result from differences in the structures or data manipulation languages of databases, hinders cooperation among information.

Different approaches have been advocated for addressing the problem. A classic solution is to unify the component information systems by a global schema, which resolves semantic differences between the local schemas. User queries are posed on the global schema and mapped into queries on the local schemas. This solution is appropriate for tightly coupled federated systems [Sheth90]. In this way, several approaches for allowing interoperability of information systems use a higher-order language or a meta-model to design global schemas. Barsalou et al [Barsalou92] define an extensible meta-model that provides a set of meta-types to represent concepts of other data models. In the same way, Atzeni et al [Atzeni93] provide a meta-model composed of a limited set of meta-constructs to modelize other data models. Hong et al [Hong90] present a meta-model to resolve semantic differences of various object oriented data models. Another way is to use a common multi-database language approach which allows query variables to range over databases, attributes and values (Litwin et al [Litwin93]).

In contrast to this, the loosely coupled federated approach does not require all schemas to be integrated into one or more global schemas. Instead, it is based on import schemas or views that can be combined on demand at various levels (end user, local data base administrator, ...) [Hsiao92a, Hsiao92b]. In this way, interoperability can be achieved by a mapping approach which defines a pairwise mappings between existing databases. However, it may require a set of translators whose number become quadratic in terms of the number of data models.

Data models translation plays a key role in the loosely coupled federated approach. In this paper we focus on addressing multi-datamodel translation problems involved in the mapping approach to reduce the translation complexity. We present an ongoing research that is based on an extensible meta-model called TIME (Traducteur Intelligent avec Méta-modèle Extensible). The meta-model defines a set of meta-types which allow the representation of various modelling concepts. It includes a knowledge base and inference engine that uses transformation rules, expressed in first order logical predicate, to convert a schema or query from a source model to one or more target model(s). A key feature of our solution is to allow reuse of both meta-types definition and associated translation rules [Nicol95, Nicol96].

Our methodology of datamodel translation is composed in two steps. First, specification of datamodel concepts in the meta-model; Second, translation of schemas from a source model to one or more target model(s). This is done by translating the source schema into an equivalent meta-model schema, then by applying equivalence preserving transformation rules to the intermediate schemas and finally by translating the result into the target model.

2. Meta-model Definition

The meta-model TIME defines two main components: a set of meta level concepts called meta-types, and a knowledge base that contains a set transformation rules. To achieve extensibility, the meta-types are organized in a specialization (generalization) hierarchy that allows the definition of a new meta-types by specialization of existing meta-types. A meta-type M is defined by a tuple (AM,CM,PM) where AM is a set of syntactic elements that describe the structure of M. CM is a set of user defined constraints that are used to restrict the meta data constraints associated with the super meta-type of M. PM is a set of operations or methods used primarily to model methods of the object oriented (or any similar) model. PM is empty for data models which do not allow the encapsulation of data and operation into a type (e.g. relational model).

Figure 1 depicts the meta-model extended by the definition of two data models. Level 1 contains the set of basic meta-types. The highest meta-type, META, is a generic meta-type that define an identity function. MComplex-Object (CO) and MSimple-Object (SO) meta-types correspond to the complex structures and flat entities respectively. MNary-Link (NL) and MBinary-Link (BL) meta-types are used to categorize n-ary and binary relationships respectively.

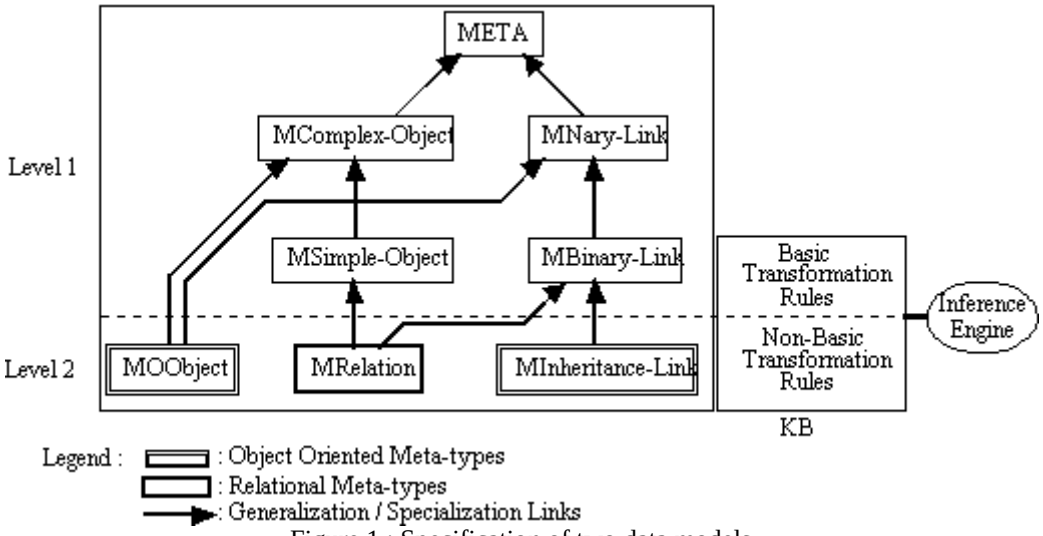

Figure 1 : Specification of two data models

Interoperability between two or more data models requires the meta-model extension to include their respective modelling concepts. For example, level 2 in figure 1 displays the meta-types that correspond to the modelling concepts of the Relational model (concept of Relation) and the concepts of the Object Oriented model (concepts of object and inheritance link).

To model the concept of relation, the basic meta-type MSimple-Object is specialized to define structures of the relational model. The notion of primary keys of the relational data model is defined from the identity function inherited from META. Inclusion dependencies can not be defined from MSimple-Object. But they can be derived from structures and constraints associated with MBinary-Link. Thus, we obtain a multi-inheritance link from MSimple-Object and MBinary-Link to the new type MRelation (REL).

To introduce the object oriented data model in the meta-model we define two new metatypes. First, to represent the object concept, second to represent inheritance link concept. We specialize the basic meta-type MComplex-Object to represent the structure of an object. Nevertheless, reference attributes which modelize composition links are represented by specializing the meta-type MNary-Link. We obtain a multi-inheritance link from MComplex-Object and MNary-Link to the new meta-type MOObject. Inheritance links of the object oriented data model are represented as a specialization of the basic meta-type MBinary-

Link because it is the nearest concept in the structure. We redefine the constraints of MBinary-Link to correspond of the notion of inheritance, and add cardinalitities constraints on the structure.

3. Transformation rules definition

Transformation rules, that are coupled with the specialization hierarchy, transform a meta-schema into a target meta-schema by converting one instance of a meta-type into one or more instance(s) of another meta-type directly connected in the specialization hierarchy. It may in some case generate new instances of another meta-type. The knowledge base initially contains the set of basic transformation rules which work on basic meta-types. New transformation rules are added to this base to handle instances of new meta-types. In the case of multi-inheritance link, more than two rules are defined. One from the new meta-type to its super-meta-types and one from each super-meta-type to the new one.

These rules are expressed in a first order logical predicates and are used by the inference engine. A rule is of general form R(I1, M1, I2, M2), where I1 is the source meta-schema, I2 is the target meta-schema, M1 and M2 are directly linked meta-types. The application of this rule produces the target meta-schema I2 from I1 by converting all instances IM1 of meta-type M1 in the source meta-schema I1 into one or more instance(s) IM2 of meta-type M2. For each transformation rules we associate constraints by defining preconditions. These constraints allow to chose which instances to translate during the translation process.

There are two groups of transformation rules. The first group is composed of four basic transformation rules. The rule Rb(I1, CO, I2, SO) flattens all complex structures into a set of objects with simple structure, and generates the appropriate set of binary links to connect the newly created simple objects to the original object. The rule Rb(I1, SO, I2, CO) may be used to restructure two or more simple objects linked by binary links into a complex object. The rule Rb(I1, NL, I2, BL) transforms an instance of MNary-Link into a set of instances of MBinary-Link. Note that in some cases the transformation of an n-ary link can eventually generate an instance of MComplex-Object. This new instance of object is linked by the new binary links to the participant objects of the initial n-ary link. The rule

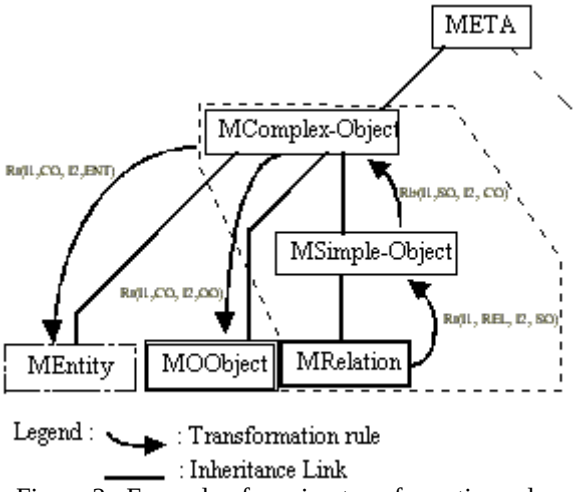

Figure 2 : Example of reusing transformation rules

Rb(I1, BL, I2, NL) does the reverse mapping from instances of MBinary-Link to obtain one instance of MNary-Link.

The second group contains all the non-basic transformation rules. Due to lack of space we present only three rules associated with the meta-type MRelation and its super-meta-types. The rule Rt(I1, REL, I2, SO) transforms an instance of a meta-type MRelation into an instance of a meta-type MSimple-Object .

Constraints on inclusion dependencies defined in MRelation are translated into instances of MBinary-Link to conserve the same semantic during the transformation. The rule Rt(I1, SO, I2, REL) transforms an instance of a meta-type MSimple-Object into an instance of a meta-type MRelation. The rule Rt(I1, BL, I2, REL) transforms an instance of a meta-type MBinary-Link into instances and/or inclusion constraints of MRelation. Links and cardinalities associated to the initial instance of MSimple-Object are used to define foreign key constraints.

4. Translation path and rule reusability.

Once the meta-type hierarchy is created, interoperation can be done by translating schemas, queries and application from one model to another model. The meta-type hierarchy and the corresponding translation rules are recorded in the knowledge base. Our methodology re-uses existing transformation rules and existing meta-types to define new transformation rules and meta-types. After building a new meta-type, the definition of translation between this new meta-type and its super-meta-type(s) is made easier by the fact that the two meta-types has close syntactics. The step by step transformation of instances from a source meta-type to target meta-types allows a best control of the translation and avoids loss of semantics. A translation path represents a complete transformation from a source schema to a target model. It is composed of a set of basic and non basic transformation rules that can be reused for others transformations (Figure 2, dotted line).

Transformation can be shared and reused by several translation paths when we translate the same source schema into one or more target models. Figure 2 shows an example of translation paths from the meta-type MRelation to two meta-types : MOObject and MEntity. This last meta-type modelize the concept of Entity of ERC+ data models. Note that the path leading from MRelation to MComplex-Object is reused when the relational schema is translated to Object-Oriented (MOObject) and ERC+ (MEntity) data models.

5. Conclusion.

We have presented a methodology for multi-datamodel translation in cooperative databases systems. We addressed the semantic heterogeneous problem of data models by defining and using a meta-model called TIME. The meta-model can be used as an intermediate data model translator design tool to allow interoperability. The meta-model provides a minimum set of meta-types that capture the semantics of different modelling concepts. It achieves extensibility by defining new meta-types as specialization of existing meta-types. Finally, it allows the reusability of translation rules by defining and coupling schema transformation rules to the meta-type specialization/generalization lattice. Our future objectives are to expand the above results to define a formal methodology and algorithms for cooperative query processing. This will allow us to define query interface for the interoperation or migration of existing systems. Moreover, the meta-types generalization hierarchy, the transformation rule and the reusability property of the model can be used to semi automatically generate data model compilers.

References available upon request from second author.