Multi-Data Models Translations In Interoperable Information Systems

Christophe Nicolle, Djamal Benslimane, Kokou Yetongnon

Laboratoire LIESIB
Facultés de Sciences Mirande
Université de Bourgogne
21000 Dijon, FRANCE
Email : {cnicolle, benslima, kokou}@satie.u-bourgogne.fr

Abstract. Interoperation of heterogeneous and autonomous information systems has traditionally been hampered by semantic differences in their data models. In this paper, we address the problem by defining a methodology called TIME, which is based on an extensible meta model. Its key features are : a set of meta-types which can be used to represent the syntax and the semantics of data modeling concepts, a knowledge base of transformation rules that map a meta-type into other meta-types, and an inference engine which uses the transformation rules to translate schema from source to target models. The extensibility of the meta-model is achieved by organizing the meta-types into a generalization hierarchy that record similarities among modeling concepts. The hierarchy of meta-types allows the reuse of transformation rules during automatic generation of data model translators.

1 Introduction

To facilitate information exchanges between different databases, several systems that highlight the evolution of information sharing have been developed in the course of these last years. This evolution goes from a simple exchange of message (cooperative system) to a total unification of a set of databases as a unique database (federated and distributed system). These systems requires the definition of concepts to allow the interoperation of heterogeneous computers and local databases systems. The heterogeneity can take on several aspects such operating systems, semantics and internal physical organization models. In this article, we are interested only in heterogeneity semantic problems that result from differences in the structure, data manipulation languages and content of database information. These problems complicate the interoperation between the different information systems.

A classic solution is to integrate the local schemas into a global schema. Usually, the global schema is expressed in a common data model, in which the user expresses his queries [12]. The integration of a large number of data schema is nevertheless an expensive and intensive task [11]. An alternative is to place translation tools between the different data models within a federation. Thus, users of the federation access information from their local data model [14, 15].

The responsibility of translating from concepts of one model into another model is left to the user or to the administrator of the local system. But, this solution rapidly becomes too complex to implement when the number of heterogeneous databases wishing to communicate becomes large. This method is not therefore well adapted to a heterogeneous databases federation [8, 9].

Our objective is to define an environment that provides a tool to help solve the problem of translation in a heterogeneous database federation. In order to do so, we propose a methodology of data models translation based on an extensible meta-model called TIME (Traducteur Intelligent avec Métamodèle Extensible). TIME is composed of a minimum set of modelisation concepts which can be specialized to represent different data models. The main advantages of this methodology is that it enables the reuse of translation rules between different models.

The remainder of the article is organized as follows. In section 2, we present a discussion of different approaches of data model translation. Section 3 presents our meta-model (its structure, its meta-types and associated translation rules). In section 4, we describe translation rules that are associated with the meta-model, and we define the notion of meta-type correspondence. Section 5 discusses examples of specifications of the relational and object oriented data models. An example of scheme translation is presented in section 6. And finally, section 7 provides a conclusion to the article and a discussion of our future themes of research.

2 Discussion

Data model translations has been the subject of intensive research recently. The different solutions can be classified into 3 main approaches.

The first approach deals with one to one translations betweem two specific data models. This type of solution can be used for the migration of a database from a semantically poor model to a richer model or for conceptual design of database by first representing the system in a rich data model and translating the conceptual representation into a target model [3, 4]. This type of translation, where each concept of a source model finds its equivalent in the target model insures a great reliability of conversion. Nevertheless it is not well adapted for the interoperability of heterogeneous systems, where a considerable number of translators may be required (for N different models it is necessary N*(N - 1) translators). The addition of a new database with a new model necessitates the creation of as many translators as there are different models in the system.

The second approach, which is based on the utilization of a global model, is generally used in distributed systems. It allows for a one to many type of translation that is used to integrate the various databases into one system. The objective is to provide an interface between a user and the group of databases. The advantage of this solution is the reduction of the number of translators required. To add a database with a new data model and langage, it suffices to

create its equivalent in the global model. Users queries are posed on the global schema and mapped into queries on the local schemas.

To facilitate the integration of databases some solutions use a meta-model [1, 2, 10]. A meta-model is well suited for resolving semantic heterogeneity among data models. However, this solution requires users to express their queries with higher order logical predicates. An alternative is to map the meta-model back in the local user model. Thus allowing users to express queries in their native model.

The third approach is based on the utilization of a canonical or pivot model that allow the translation of N data models into M data models. This type of translation is very difficult to implement and generally corresponds to the specification of N+M one to one translations between the pivot model and the other models.

The above approaches have pointed out the need to reduce the number of translators requires to allow interoperability of information systems. Tools that are based on the reuse of pre-defined concepts and translations can be used to facilitate the generation of one to one data model translators [7, 13]. The semi-automatic or automatic generation of one to one translators thus facilitates the translation in both federated and distributed systems. To acheive this goal, we propose a data translation methodology called TIME (Traducteur Intelligent avec Méta-modèle Extensible) that uses an extensible meta-model, a knowledge base and an inference engine. The data model contains a reduced set of meta-types that are linked by generalization/specialization relationships. We use the specialization link as a mechanism for extending the meta-model by specializing existing meta-types to define new meta-types. As in object oriented models, an inheritance link between two meta-types allows the reuse of both structural properties and meta-type constraints. The inference engine uses a knowledge base containing a set of transformation rules to translate schema from source models to target models. The translation rules are expressed in first order logical predicate. The advantage of this methodology is to simplify the translation process by reusing translation paths between different models.

3 Structure of TIME Meta-model

TIME defines two main components : a set of meta-types, and a knowledge base containing a set of basic transformation rules (Figure 1). The meta-types are used to capture the semantics of different categories of modeling concepts. There are two sorts of meta-types. Meta-object types represent concepts which can be used to model real entities while meta-link types denote data model concepts which characterize relationships among entities. To achieve extensibility, the meta-types of the meta-model are organized in a specialization (generalization) hierarchy. Thus, a new meta-type can be defined by specializing an existing meta-type. Initially, the meta-type hierarchy contains 5 basic meta-types: META, MComplex-Object, MSimple-Object, MN-Link and MBinary-link (figure 1). The knowledge base (KB) initially contains a set of basic transformation rules that

can be used to transform an instance of a basic meta-type into an other basic meta-type and a set of facts describing meta-types correspondence. These rules are expressed in a first order logical predicates that are utilized by the inference engine.

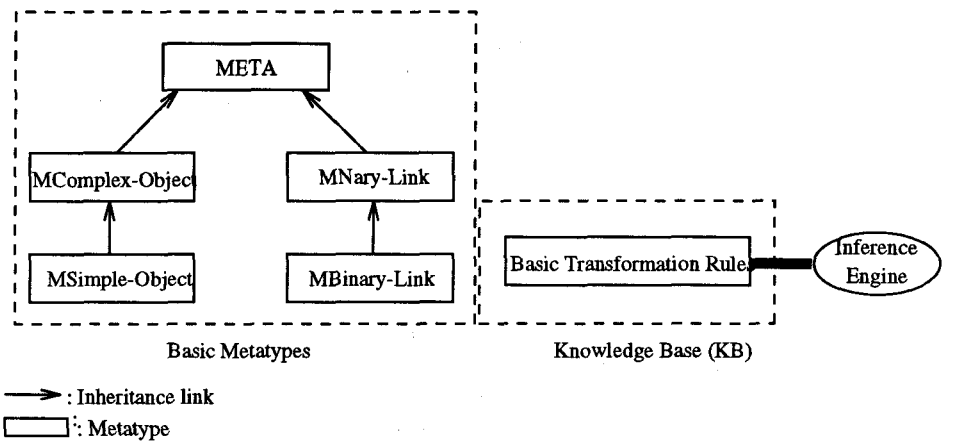

Fig. 1. Structure of the metamodel

A meta-type M is defined by a tuple M = (A_M,C_M,P_M), where A_M is a set of syntactic elements that describe the structure of M. C_M is a set of user defined constraints that are used to restrict the meta data constraints associated with the super meta-type of which M is a specialization. PM is a set operations or methods. It is used to model methods of the object oriented (or any similar) model. P_M is empty for data models (e.g. relational model) which do not allow the encapsulation of data and operations into a type. We give below an informal definition of the meta-types. A formal BNF [1] definition is given in appendix.

3.1 Definition of Meta-type META

The highest meta-type in the specialization (generalization) hierarchy is a generic meta-type, META, with an empty structure. Its purpose is to define meta-data constraints and operations that can be shared by all meta-types. It is defined by META=([ ], C_{Meta}, [ ]). C_{Meta} comprises a set of data modeling constraints. For example, C_{Meta} contains an ID function that uniquely identifies the instances of meta-types.

3.2 Definition of Meta-type MComplex-Object (CO)

Meta-type CO represents modeling concepts that are used to describe complex structure entities such as : OBJECT in object oriented model, ENTITY TYPE

[1] Backus Normal Form

in the Entity-Relationship model and RECORD in the Codasyl data model. Meta-type MComplex-Object is specified by CO = (A_{CO}, C_{CO}, P_{CO}), where A_{CO} is defined using the usual tuple and set constructions on meta-object types. The component C_{CO} and P_{CO} are not redefined at this level, but are inherited from the super meta-type META. An example showing an instance PERSON of meta-type MComplex-Object is given by:

PERSON =(A_{PERSON} := [Name : String ,
Address : [Street : String, Town : String],
Childs : [FirstName : String, BirthDate : String] (0,20),
Age : integer],
C_{PERSON} := [ ],
P_{PERSON} := [ ])

3.3 Definition of MSimple-Object (SO)

Meta-type SO, which is a specialization of meta-type CO, represents modeling constructs with flat structure. It is defined by SO = (A_{SO}, C_{SO}, P_{SO}) where the attribute structure A_{SO} inherits the tuple structure A_{CO} of meta-type CO, but restricts the type of its components to primitive domains. The components C_{SO} and P_{SO} are inherited from the meta-type MComplex-Object.

3.4 Definition of MNary-Link (NL)

Meta-type NL models types that represents connections between real world entities. Links can carried attributes. It is defined by NL = (A_{NL}, C_{NL}, P_{NL}). The component C_{NL} and P_{NL} are not redefined at this level, but are inherited from the super meta-type META. An example showing an instance OWN of an MNary-Link is given by :

OWN = (A_{OWN} := [PERSON : (1,n) , CAR : (1,1), DATE : (1,n)],
C_{OWN} := [ ],
P_{OWN} := [ ])

3.5 Definition of MBinary-link (BL)

Meta-type BL categorizes binary connections involving real world object types. It is a special case of NL and it is defined by BL = (A_{BL}, C_{BL}, P_{BL}).

4 Translation of meta-types

To allow the translation of schema with the meta-model, we define two types of transformation rules. The first, basic transformation rule allow the transformation between instances of two basic meta-types directly linked in the inheritance lattice. The second, non-basic transformation rules allow the transformation between one or more instances of non-basic meta-types (4.2).

4.1 Notion of correspondence

To allow the specification of a new meta-type in the meta-model we define the notion of meta-type correspodences to detect which existing meta-types we link with the new one. A good definition of these correspondences allows for a better definition of transformation rules between these meta-types and avoid semantic losses. In function of these correspondences, the system defined the type of the inheritance link between the meta-types (specialization link, may-be a link).

We define three meta-types correspondences : strong correspondence, simple correspondence and weak correspondence. the correspondence between meta-types M_1 and M_2 can be different from the one between meta-types M_2 and M_1.

Strong Correspondence : M_1 and M_2 are in strong correspondence relationship, that is to say that an instance of one can be transformed into an instance of the other without semantic loss if and only if :

$$M_1 = M_2$$

In this case M_1 and M_2 are equivalents.

The existing meta-type is used to represent the new one. The system establishes a virtual link between the concept of the model and the existing meta-type.

Simple Correspondence : M_1 is in relationship of simple correspondence with M_2, that is to say that during the transformation of an instance of M_2 in an instance of M_1, programs, constraints or structures can be lost, if and only if:

$$(M_1 \subseteq M_2)$$

In this case, a specialization link is introduced between M_1 and M_2. For example, MSimple-Object is in relationship of simple correspondence with MComplex-Objet (Figure 1) .

Weak Correspondence : M_1 is in relationship of weak correspondence with M_2, that is to say that during the transformation of an instance of M_2 in an instance of M_1, constraints and structures can be lost. In this case, a specialization link is introduced between M_1 and M_2. Moreover, to avoid semantic losses, the system compares others meta-types with M_1 in order to link the new meta-type with another type by a Maybe a link. For example, in figure 3, MRelation is in relationship of weak correspondence with MSimple-Object. This is due to inclusion dependencies. The system links MRelation with MBinary-link to represent these dependencies by links during the transformation process.

If more of one existing meta-types are in weak equivalence with the new meta-type, the system choose the existing meta-type where differences are smaller.

4.2 Basic transformation rules

A transformation rule converts an instance of a basic meta-type to one or more instance(s) of the basic meta-type which are directly connected in the specialization (generalization) hierarchy. Transformation rules are expressed in first order logical predicate of general form $R_b(I_1,M_1,I_2,M_2)$, where I_1 is the source meta-scheme, I_2 is the target meta-scheme, M_1 and M_2 are basic meta-types. This rule produces the target meta-scheme I_2 from I_1 by converting all instances of meta-type M_1 in the source meta-scheme I_1 into one ore more instance(s) of meta-type M_2. There are four transformation rules:

- $R_b(I_1,nl,I_2,bl)$ transforms an n-ary meta-type NL into a set of binary Link. Note that in some cases an n-ary link (NL) can be converted into both a meta object MO and a set of binary links where the participant meta-types of the initial n-ary link are linked to MO by the binary links. This is necessary to preserve the semantic correspondence between I_1 and I_2 when the participant object meta-types have cardinalities of (1,N) or (0,N). This rule is formally described in appendix 8.2.
- $R_b(I_1,bl,I_2,nl)$: this transformation rule noted only nas a trivial case since by definition a binary link is a an Nary-Link.
- $R_b(I_1,$ so,$I_2,$ co) by definition this transformation is trivial since simple objects (SO) are complex objects with flat structures. In some cases, however, it may be necessary to restructure two or more simple objects linked by binary links into a complex object.
- $R_b(I_1,co,I_2,so)$ flattens all complex structures into a set of objects with simple structure, and generates a set of binary link meta-types (BL) to connect the newly created simple objects to the original object.

A formal description of the first transformation rule, $R_b(I_1,nl,I_2,bl)$, is given in appendix.

Figure 2 shows the use of the transformation rule $R_b(I_1,co,I_2,so)$. This rule translate the source meta-scheme I_1 into a target meta-scheme I_2 by replacing the complex-object PERSON with three instances of meta-type MSimple-Object PERSON, CHILDS, ADDRESS linked by two instances of meta-type MBinary-Link $PERSON-CHILDS$, $PERSON-ADDRESS$.

5 Data models specification

The specification of a data model in TIME is accomplished in two steps :

- Step 1 defines meta-types corresponding to the concepts of the data model by specializing existing meta-types or by adding new meta-types.
- Step 2 defines and associates transformation rules between the meta-type created and their respective super meta-types in the generalization hierarchy. In this step meta-type correspondence is defined, and the knowledge base is completed by adding new rules to the set of transformation rules and new meta-types correspondences to the fact base.

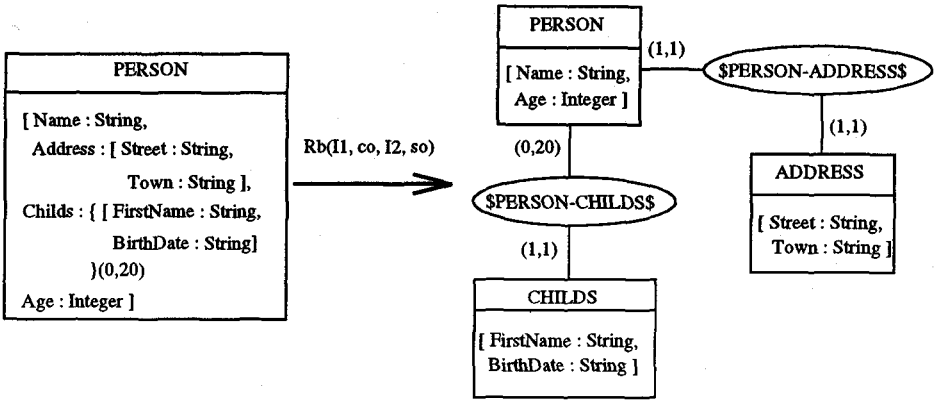

Fig. 2. An example of transformation of MComplex-Object into MSimple-Object and MBinary-Link

In this section we present an example to illustrate the specification of concepts of the relational and object oriented models. The specification of other models in the meta-model TIME is given in [6]. Figure 3 depicts two levels. The top level shows the basic meta-types, basic transformation rules and basic meta-type correspondence. The second level displays the meta-types that correspond to the modeling concepts of the Relational and Object oriented data models, and the transformation rules between these new meta-types, their super-meta-types and the new meta-types correspondences.

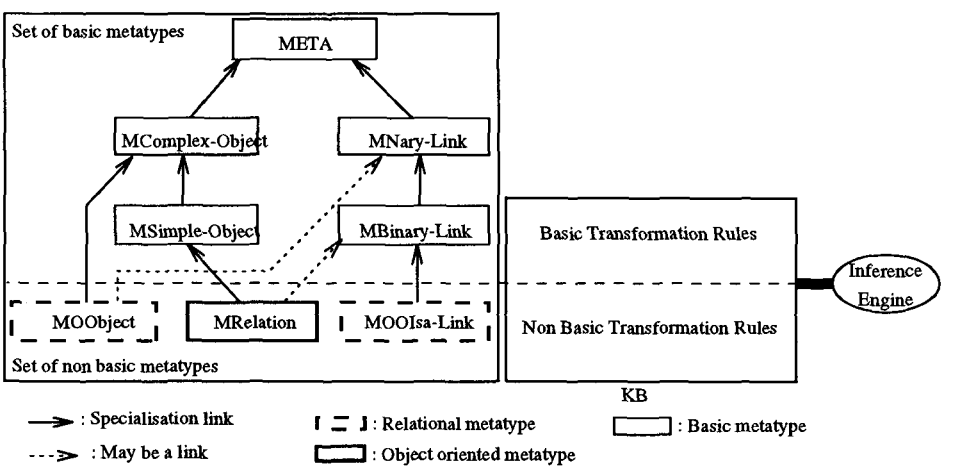

Fig. 3. Specification of two data models

5.1 Specifying the Relational Model in the meta-model TIME

In this section we specify the concepts of the relational model in the meta-model TIME. We follow this with a description of the transformation rules of each concept.

The specification of the relational data model in TIME consists to define a meta-type MRelation (REL) which represents the concept RELATION (relation name, monovalued attributes on simple domains, functional and inclusion dependencies). It is defined by REL=(A_{REL}, C_{REL}, P_{REL}). As shown in figure 3, MRelation specializes the meta-type MSimple-Object. Thus, the structure A_{REL} is identical to the structure of A_{SO}. MRelation, in C_{REL}, refines the ID function defined in META to correspond to the precise definition of the relational primary key; namely, a sub-sequence of attribute labels that uniquely identify the tuples of a relation. P_{REL} is empty.

Two non-basic transformation rules expressed in the first order logical predicate are associated with meta-type MRelation. They are used to convert instances between MSimple-Object and MRelation meta-types. They are informally defined as follows :

- $R_t(I_1,so,I_2,rel)$ transforms directly an instance of a meta-type SO into an instance of a meta-type R. To complete the definition of this new instance, the system analyzes links and cardinalities associated with the initial instance of SO to define primary key and foreign key constraints.
- $R_t(I_1,bl,I_2,rel)$ transforms an instance of a meta-type BL into attributes defining foreign key constraints on the instance of meta-type REL.
- $R_t(I_1,rel,I_2,so)$ transforms an instance of a meta-type REL into an instance of a meta-type SO. The constraint on keys defined in REL is translated into instance of BL to conserve the same semantic during the transformation.

5.2 Specifying the object oriented model in the meta-model TIME

This section presents the specification of the concepts of the object oriented data model, and gives a description of the transformation rules.

Meta-types MOObject (OO) and MOOIsa-Link (OIL) represent the object oriented concepts, object and IS-A link, respectively. As illustrated in figure 3, the meta-type MOObject specializes the meta-type MComplex-Object. It is defined by OO = (A_{OO}, C_{OO}, P_{OO}). The structure A_{OO} is identical to the structure of A_{CO}. Conversely, The meta-type MOObject refines the component P_{CO} to introduce the behavioral aspect of the objects. A detailed and formal description of methods is beyond the scope of this paper. Thus the component P_{OO} is not presented. C_{OO} is empty. The meta-type MOOIsa-link is a binary link. In addition to the constraints inherited from meta-type MBinary-Link, it maintains a subset constraints between the population of the specialized and generalized meta-types which generalizes the concept of inheritance of the object oriented data model. Moreover, MOOIsa-Link specialises the structure of MBinary-Link to represent connection without attribute. The meta-type MOOIsa-Link is defined by OIL = (A_{OIL}, C_{OIL}, [ ]).

- R_t(I_1,co,I_2,oo) transforms directly an instance of a meta-type MComplex-Object into an instance of a meta-type MOObject. The composition links are transformed into instance of MNary-Link meta-type.
- R_t(I_1,oo,I_2,co) transforms an instance of a meta-type MOObject into an instance of a meta-type MComplex-Object. The transformation of the behavioral aspect is not presented here to simplify our discussion. The composition links are transformed into instance of MNary-Link meta-type.
- R_t(I_1,nl,I_2,oo) transforms an instance of a meta-type MNary-Link into reference attribute in the structure of the instance of MOObject meta-type MOObject.
- R_t(I_1,bl,I_2,oil) directly transforms an instance of a meta-type MBinary-Link into an instance of a meta-type MOOIsa-Link. The identical attributes in instances O_1 and O_2 in the are absorbed into the sub-type OO_2.
- R_t(I_1,oil,I_2,bl) transforms an instance of a meta-type MOOIsa-Link into an instance of a meta-type MBinary-Link. The attributes of the surper type instance are created into the sub-type instance.

6 Translation From Relational to Object Oriented Model

In this section we first describe the principles of the translation process give an example to illustrate the translation between a relational data model and an object oriented data model.

6.1 Principles

Let RS be a relational scheme. Our goal is to use the meta-model TIME to obtain an equivalent scheme in an object oriented data model. This requires three steps:

- **Step 1** : Translate the initial relational scheme into an intermediate meta-scheme I1 that includes instances of meta-type MRelation and key constraints expressions.

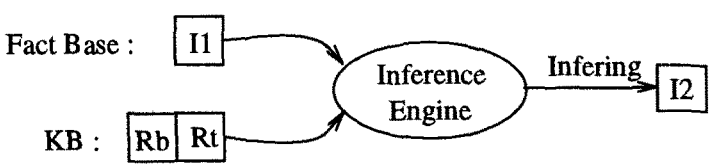

Fig. 4. Transformation of I_1 into I_2

- **Step 2** : Transform I_1 in to I_2 by replacing all instances of meta-type MRelation by instances of meta-types MOObject and MOOIsa-Link.This is done with the inference engine by applying basic and non-basic transformation rules (R_b and R_t) on the instances of meta-scheme (I_1) (cf. Figure 4).
- **Step 3** : Translate the meta-scheme I_2 into a final object oriented scheme.

6.2 Example

Let RS be the following relational scheme :

PERSON (PersonN#, Name)
CHILDREN (PersonN#, BirthDate)
ADDRESS (AddressN#, Number, Street, Town)
LIVE (People, House, Since)
CAR (CarN#, Color, Type, Owner, Date)

and let the constraints which define the foreign keys be given by :

CHILDREN.PersonN# $\subset$ PERSON.PersonN#
CAR.Owner $\subset$ PERSON.PersonN#
LIVE.People $\subset$ PERSON.PersonN#
LIVE.House $\subset$ ADDRESS.AddressN#

The first step of the translation process is to create a meta-model instance for each concept in the scheme. In this case, we must create instances of the meta-type MRelation for each relation. The application of the translation rules to the relational scheme example yields the following intermediate meta scheme I_1 that includes instances of meta-type MRelation and key constraints expressions.

I1 = (PERSON = (A_{PERSON}=[PersonN#:String, Name:String],
C_{PERSON}=[PERSON(PersonN#) $\rightarrow$ PERSON(Name)],
P_{PERSON}=[ ]),

CHILDREN = ($A_{CHILDREN}$=[PersonN#:String, BirthDate:String],
$C_{CHILDREN}$=[CHILDREN(PersonN#) $\rightarrow$ CHILDREN(BirthDate),
CHILDREN.Person N# $\subset$ PERSON.PersonN#],
$P_{CHILDREN}$=[ ]),

ADDR = (A_{ADDR}=[AddrN#:String, Num:String, Street: String, Town:String],
C_{ADDR}= [ADDR(AddrN#) $\rightarrow$ ADDR(Num,Street,Town)],
P_{ADDR}=[ ]),

CAR = (A_{CAR}=[CarN#:String, Type:String, Owner:String, Date: String],
C_{CAR}=[CAR(CarN#) $\rightarrow$ CAR(Type), CAR(CarN#,Owner) $\rightarrow$ CAR(Date),
CAR.Owner $\subset$ PERSON.PersonN#],
P_{CAR}=[ ]),

LIVE = (A_{LIVE}=[People:String, House:String, Since:String],
C_{LIVE}=[LIVE(People, House) $\rightarrow$ LIVE (Since),
LIVE.People $\subset$ PERSON.PersonN#,
LIVE.House $\subset$ ADDRESS.AddressN#] ,
P_{LIVE}=[ ]))

Next, we map the above instances of relational meta-types MRelation into a scheme containing instances of basic meta-types. The goal is to obtain a meta-scheme that contains object oriented meta-constructs. We achieve this by following the translation paths form MRelation to the meta-type MOOBject and by

applying transformation rules among the basic meta-types. We use the foreign key constraints to classify the relations into MSimple-Object or MBinary-Link types. Labels of the form 'XXX' are intermediate instances generated by the translation process. The results of this step are given by following the intermediate meta scheme (I_1).

I1' = (PERSON = (A_{PERSON}=[PersonN#:String, Name:String],
C_{PERSON}=[ ],
P_{PERSON}=[ ]),
CHILDREN = ($A_{CHILDREN}$=[BirthDate:String],
$C_{CHILDREN}$=[ ],
$P_{CHILDREN}$=[ ]),
ADDR = (A_{ADDR}=[AddrN#:String, Num:String, Street: String, Town:String],
C_{ADDR}= [ ],
P_{ADDR}=[ ]),
CAR = (A_{CAR}=[CarN#:String, Type:String, Date:String],
C_{CAR}=[ ],
P_{CAR}=[ ]),
LIVE = (A_{LIVE}=[Since:String],
C_{LIVE}=[ ],
P_{LIVE}=[ ]) ,
$PEOPLE-LIVE$ = ($A_{\$PEOPLE-LIVE\$}$ = [PERSON:(1,n), LIVE: (1,1)],
$C_{\$PEOPLE-LIVE\$}$ = [ ],
$P_{\$PEOPLE-LIVE\$}$ = [ ]),
$HOUSE-LIVE$ = ($A_{\$HOUSE-LIVE\$}$ = [ADDR:(1,n), LIVE: (1,1)],
$C_{\$HOUSE-LIVE\$}$=[ ],
$P_{\$HOUSE-LIVE\$}$=[ ]),
OWN = (A_{OWN} = [PERSON:(1,n),CAR:(1,1), [Date:String]],
C_{OWN} = [ ],
P_{OWN} = [ ]),
$PERSON-CHILDREN$ =
($A_{\$PERSON-CHILDREN\$}$ = [PERSON:(0,1), CHILDREN:(1,1)],
$C_{\$PERSON-CHILDREN\$}$ = [ ],
$P_{\$PERSON-CHILDREN\$}$ = [ ])]

To obtain inheritance link, we transform instances of MBinary-Link (Only instances on the general structure : [$Object_1$: (0,1), $Object_2$: (1,n)]) into instances of MInheritance Link.

To obtain object types, the simple object meta-types are first changed into complex object meta-types which generalize the object oriented type. This transformation is straightforward. It follows from the generalization link between MSimple-Object and MComplex-Object. Next, we use rules $R_t(I_1$",nl,I_2,co) and $R_t(I_1$",co,I_2,oo) to translate object and relationship meta-types into object oriented meta-type as defined below :

I2 = (PERSON = (A_{PERSON}=[PersonN#:String, Name:String],
C_{PERSON}=[ ],
P_{PERSON}=[ ]),

CHILDREN = ($A_{CHILDREN}$=[BirthDate:String],
$C_{CHILDREN}$=[ ],
$P_{CHILDREN}$=[ ]),

ADDR = (A_{ADDR}=[AddrN#:String, Num:String, Street:String,Town:String],
C_{ADDR}= [ ],
P_{ADDR}=[ ]),

CAR = (A_{CAR}=[CarN#:String, Type:String, Own:PERSON, Date: String],
C_{CAR}=[ ],
P_{CAR}=[ ]),

LIVE = (A_{LIVE}=[Since:String, People:PERSON, House:ADDR],
C_{LIVE}=[ ],
P_{LIVE}=[ ]) ,

$PERSON-CHILDREN$ =
($A_{\$PERSON-CHILDREN\$}$ = [PERSON:(0,1), CHILDREN:(1,1)],
$C_{\$PERSON-CHILDREN\$}$ = [ ],
$P_{\$PERSON-CHILDREN\$}$ = [ ])]

After the step 3 we obtain the following target object oriented scheme :

Class PERSON
Type Tuple (PersonN# : string, Name : String)

Class CHILDREN inherit PERSON
Type Tuple (BirthDate : String)

Class ADDR
Type Tuple (AddrN# : String, Num : String, Street : String, Town : String)

Class CAR
Type Tuple (CarN#:String, Type: String, Own : PERSON, Date : String)

Class LIVE
Type Tuple (Since : String, People : PERSON, House: ADDR)

7 Conclusion

In this paper we have presented a methodology for translating multiple data models. We addressed the problem by defining an extensible meta-model called TIME and used it as a data model translator design tool. The main characteristics of the meta-model are :

- It provides a minimum set of meta-types that capture the semantics of different categories of concepts found in most data models.
- It achieves extensibility by organizing the meta-types in an specialization/generalization hierarchy. Thus, a new meta-type is defined by specializing an existing meta-type.
- It achieves translation by defining a knowledge base composed of a set of transformation rules.

- It allows the reuse of transformation rules and sharing of translation step between multi- models translation.

Our future objectives are first to use the notion of correspondence to place new meta-types in the inheritance lattice of our meta-model. We will work to associate the notion of correspondence with the notion of distance to measure the level of semantic losses during the translation process. Next, we need to extend the above results, and to define a formal methodology and algorithm for heterogeneous query processing. This will allow us to define a query interface for the interoperation or migration of existing systems. Last, to use the meta-types generalization hierarchy, the knowledge base of transformation rules and the reusability property of the model to automatically or semi-automatically generate data model translators.

8 Appendix

8.1 BNF description of the meta-types

This section presents the description for each basic meta-types and for the meta-types MRelation, MOObject and MOOIsa-Link.

MComplex-Object : CO = (A_{CO}, [], []) where A_{CO} is described in BNF by :

Let A a set of labels,
A_{CO} := [structure_CO] | { structure_CO } (m,n) | {structure_CO }
structure_CO := a : T | a : T, structure_CO
where a is an attribute name (a ∈ A),
T := D | A_{CO}
D := String | ... | Integer
m := 0 | n
n := 1 | 2 | ...

MSimple-Object : SO = (A_{SO}, [], []) where A_{SO} is described in BNF by :
A_{SO} := [structure_SO]
structure_SO := a : T | a : T, structure_SO
where a is an attribute name (a ∈ A),
T := D

MNary-Link : NL = (A_{NL}, [], []) where A_{NL} is described in BNF by :
A_{NL} := [structure_NL]
structure_NL := M : (m,n), M : (m,n) | M : (m,n), M : (m,n), A_{CO} |
M : (m,n), structure_NL
where a is an attribute name (a ∈ A),
where the type of M is a CO or SO and M ∈ A

MBinary-Link : BL = (A_{BL}, [], []) where A_{BL} is described in BNF by :
A_{BL} := [structure_BL]
structure_BL := M : (m,n), M : (m,n) | M : (m,n), N : (m,n), A_{CO}

where a is an attribute name (a ∈ A),
where the type of M is a CO or SO and M ∈ A

MRelation : REL = ([], C_{REL}, []), the constraints C_{REL} are defined by three axioms as follows : Let R_1, R_2 be instances of REL, A_i a sub-sequence of attribute labels and F_k the ID function defined in META,

Axiom 1 defines a key is not null is given by :
$\forall R_1 \in REL, \forall A_i \in R_1, \forall F_k \in ID(R_1), R_1.F_k(A_i) \neq NULL$

Axiom 2 defines a primary key, whose attribute(s) define(s) uniquely all non key attribute(s) is given by :
$\forall R_1 \in REL, \forall A_1, A_2 \in R_1, \forall F_k \in ID(R_1)$,
where $R_1.F_k(A_1), R_1(A_1) \rightarrow R_2(A_2)$

Axiom 3 defines a foreign key, who are key attribute(s) of an instance R_1 of REL define(s) uniquely a sub-sequence of non key attribute(s) of an instance R_2 of REL.
$\forall R_1, R_2 \in REL, \forall A_1, A_2 \in R_1, \forall A_2 \in R_2, \forall F_{k1} \in ID(R_1)$
with $R_1.F_{k1}(A_1), \forall F_{k2} \in ID(R_2)$ where $R_2.F_{k2}(A_2), R_1(A_2) \subset R_2(A_2)$.

MOObject : OO = (A_{OO}, [], P_{OO}) where A_{OO} is described in BNF by :
A_{OO} := [structure_OO] | { structure_OO } (m,n) | {structure_OO }
structure_*OO* := a : T | a : T, structure_OO
where a is an attribute name (a ∈ A),
T := D | A_{CO} | t' where the type of t' is a OO and t' ∈ A
D := String | ... | Integer
m := 0 | n
n := 1 | 2 | ...
The term P_{OO} is not developed in this paper.

MOOIsa-Link : OIL = (A_{OIL}, C_{OIL}, []). The structure A_{BL} and the constraints C_{BL} are refined in the meta-type MOOIsa-Link as follows :
A_{OIL} := [structure_OIL]
structure_OIL := M : (0,1), M : (1,1)
where a is an attribute name (a ∈ A),
where the type of M is a OO and M ∈ A

C_{OIL} is defined as follows,
Let OO_1 and OO_2 instances of the meta-type MOObject,

Axiom 4 : ∀, OO_1, OO_2,
OIL = [OO_1 : (0,1), OO_2 : (1,1)] ⇒ (∀ e : OO_1(e) ⇒ OO_2(e))

This constraint defined that OO_2 is a subtype of OO_1.

8.2 Formal description of $R_b(I_1, nl, I_2, bl)$

$\forall$ M_{nl} = (A_{nl}, C_{nl}, P_{nl}) instance of MNary-Link
$r_b(M_{nl}, nl, M_{bl}, bl) \rightarrow M_{bl}$ instance of MBinary-Link,
N_j instances of MBinary-Link (j = 1, 2, ..., m),
M_{oc} instance of MComplex-Object,

- Two objects are linked by the instance of meta-type MNary-Link.
  $\forall$ A_{Mnl} := [M_1 : ($Cmin_1$, $Cmax_1$), M_2 : ($Cmin_2$, $Cmax_2$)] [2]
  $\Rightarrow$ Creation of one instance M_{bl} with the same name of M_{nl} with type of MBinary-Link were

$$A_{Mbl} := [M_1 : (Cmin_1, Cmax_1), M_2 : (Cmin_2, Cmax_2)]$$

- More of two object are linked by the instance of MNary-Link, maximal cardinalities of these object are sup of 1.
  $\forall$ A_{Mnl} := [M_1 : ($Cmin_1$, $Cmax_1$),..., M_n : ($Cmin_2$, $Cmax_2$), [r_i : t_i]], n > 2
  $\Rightarrow$ Creation of one instance of M_{oc} where A_{Moc} := [r_i : t_i] [3]
  Creation of a set of instances of MBinary-Link N_j = (A_j, C_j, P_j), linking the new object and each participant object in M_{nl} |

$$A_{Nj} := [M_{oc} : (1,1), M_i : (Cmin_i, Cmax_i)], j = 1, 2, ..., m; i = 1, 2, ..., n$$

- More of two objects are linked by the instance of MNary-Link, at least one object has a maximal cardinalities equal to one [4].
  $\forall$ A_{Mnl} := [M_1 : ($Cmin_1$, $Cmax_1$),..., M_n : ($Cmin_2$, $Cmax_2$)], n > 2 and
  $\exists$ $Cmax_k$ = 1, k = 1, 2, ..., n.
  $\Rightarrow$ Creation of a set of instances of MBinary-Link N_j = (A_j, C_j, P_j), j = 1, k, linking objects M_k with others participant objects of M_{nl} |
  $A_{Nj} := [M_k : (Cmin_1, 1), M_i : (Cmin_i, Cmax_i)], k \neq i$

References

1. Paolo Atzeni, Riccardo Torlone : "A Metamodel Approach for the Management of Multiple Models in CASE Tools", Proceedings of the DEXA International Conference in Berlin, Federal Republic of Germany, 1991
2. Thierry Barsalou, Dipayan Gangopadhyay : "M(DM) : An Open Framework for Interoperation of Multimodel Multidatabase Systems",Proceedings of the 8th International Conference of Data Engineering, Tempe, Arizona, pp 218 - 227, Feb 3-7, 1992.

[2] This rule is done if A_{Mnl} has attributes
[3] or A_{Moc} := [ ], if M_{nl} has not attribute
[4] if M_{nl} has attributes, independent of the cardinalities, we apply the last rule.

3. M.Bouzeghoub, G. Gardarin, E. Metais : "Database Design Tools : An Expert System Approach", Proceedings of VLDB 85, Stockholm August 1985.
4. M.Bouzeghoub, E. Metais : "Semantic Modelling of Object Oriented Databases", Proceedings of the 17th International Conference on Very Large Databases (VLDB), Barcelona, pp3-14, September 91.
5. S.A. Demurjian, D.K. Hsiao : "Towards a Better Understanding of Data Models Through the Multilingual Data system", IEEE Trans on Software Engineering, pp 946-958, Vol 14, N7, July 1993
6. Christophe Nicolle, "TIME, un Traducteur Intelligent avec Méta-Modèle Extensible", Technical Report N94/01, November 94, University of Burgundy
7. Christophe Nicolle, Djamal Benslimane, Nadine Cullot, Kokou Yetongnon, "A Metamodel Based Methodology fot Translating Data Models in Interoperable Information System", Procceding of DATASEM, October 8-10, 1995.
8. David K. Hsiao, "Federated Database and Systems : Part I. A Tutorial on their Data Sharing" VLDB journal 1, pp127-179, Dennis Mc Leod Editor, 1992.
9. David K. Hsiao, "Federated Database and Systems : Part II. A Tutorial on their Resource Consolidation" VLDB journal 1, pp285-310, Dennis Mc Leod Editor, 1992.
10. Manfred A. Jeusfeld, Uwe A. Johnen : "An Executable MetaModel for Re-Engineering of Database Schemas", Proceedings in the International Conference on Entity-Relationship Approach, Manchester, December, 1994.
11. Laks V.S. Lakshmanan, Fereidoon Sadri, Iyer N. Subramian. : "On The Logical Foundations Of Schema Integration And Evolution In Heterogeneous Database Systems.", DOOD 93, Phénix, USA, in Computer Sciences 760, Spring Valley, Dec 93.
12. Amit P. Sheth, J.A. Larson : "Federated Database Systems, for Managing Distributed, Heterogeneous, and Autonomous Databases", pp183-235, ACM Computing Surveys, V ol 22, N3, September 90.
13. Stefano Spaccapietra, Martin Andersson, Yann Dupont, Kokou Yetongnon, Christine Parent, Christophe Nicolle, " Integrating Schemas of Heterogeneous Database Systems", in the Second Meeting on the Interconnection of Molecular Biology Databases (MIMBD95), July 20-22, 1995 in Cambridge, United Kingdom.
14. Susan D. Urban : "A Semantic Framework for Heterogeneous Database Environments", First International Workshop on interoperability in Multidatabase Systems, Kyoto, Japan, pp 156-165, April 1991.
15. Susan D. Urban : "Resolving Semantic Heterogeneity Through the Explicit Representation of Data Model Semantics". Sigmod Record, Vol 20, N4, December 1991.