# AN ONTOLOGY-BASED APROACH FOR EXCHANGING DATA BETWEEN HETEROGENEOUS DATABASE SYSTEMS

Mourad El-Hadj Mimoune, Guy PIERRA, Yamine AIT-AMEUR
LISI-ENSMA BP 40109 – Site du Futuroscope 86961 FUTUROSCOPE Cedex France
Tel: +33 5 49 49 80 60 Fax: +33 5 49 49 80 64
Email: {*mimoune, pierra,* yamine *}@ensma.fr*

Keywords: Heterogeneous databases integration, meta-modelling, database federation, ontology based database, event-based programming, PLIB, data exchange.

Abstract: This paper presents an approach which allows data exchange between heterogonous databases. It targets at simultaneously semantic and structural heterogeneity. From the semantic point of view, this approach proposes an ontology based approach. On the one hand this ontology can be referenced by universal identifiers and acceded by queries; on the other hand, it can be exchanged between heterogonous databases systems. From the structural point of view, this approach is based on the use of a generic meta-schema, formalised in the EXPRESS language, and allowing the exchange of any instance of any database schema. Exchanged instances reference, as much as needed, the global unique identifiers defined by the ontology. However, the conversion of exchange files to the various target systems can be achieved in a generic manner (e.g. independently of the particular exchanged model). The interest of the EXPRESS language to achieve directly such a program is presented as well.

## 1 INTRODUCTION

Any design of database goes through a conceptual or semantic modelling level. Several Object oriented languages and methods of analysis and design, such as UML (Rambaugh, 1999) OMT (Rumbaugh, 1991) or Merise-2 (Panet, 1994), may be used for that purpose. All those methods are based on the entity-relationship decomposition to which instances identification, and generalisation/specialisation mechanisms are added. The model, or conceptual schema, obtained by this modelling is intended to be implemented in various DBMS. Currently, several systems coexist. Some of them are purely relational (RDBMS), and the classical notion of object (like in O-O languages) doesn't exist, and some of them hold some notions of the object oriented paradigm (ex: Postgres or ORDBMS). There exists a third category which handles exclusively objects and all the material they rely on like in O2 or ObjectStore. Therefore, when the same design method is used to build the logical schema of a database or even when it is based on a common object model, the implementation is often largely different. This approach of design generates heterogeneous databases between which data exchange is difficult.

Difficulty for exchanging data between heterogeneous target databases results from:

1 difference between the application domains of the database,

2 (conceptual) difference between the conceptual model designed for the same model, and

3 structural difference between several implementation of the same conceptual model.

Our work tries to address the second and the third issues. We specially address domains for which ontologies can be described. Indeed, the ontology allows to define, in a single and consensual framework, categories of entities which can be represented and their fundamental attributes (Gruber, 1993). This situation often occurs when the universe of discourse consists of real world objects (like the

various types of electronic components for which the IEC 1360-4 standard defines both a reference categorisation and relevant attributes) or when it consists of entities occurring in a business processes (client, supplier, bill, order…). Moreover, we assume that the actors involved in the exchange process (source and target systems designers) have agreed on referencing an existing ontology. Note that referencing the ontology does not require to use exactly the same conceptual model nor to implement it in the same DBMS. The use of the same ontology suggests a technique for exchanging between two different systems.

Our proposed approach for the exchange of data between heterogeneous databases is based on the ontology sharing and it consists in:

- ? defining and agreeing on globally unique identifiers (GUIs) for essential concepts (entities, attributes and relations) corresponding to the universe of discourse,
- ? using a generic meta-schema, formalised in the EXPRESS language (ISO10303-11, 1994), to represent exchanged data instances in a neutral format,
- ? when some of the used concepts in the source database are not known by the target database, using a generic meta-schema, formalised in the EXPRESS language as well, to exchange the corresponding part of the ontology,
- ? using an EXPRESS physical file format (ISO10303-21, 1994) or the Standard Data Access Interface (SDAI) (ISO 10303-22, 1997) to achieve the effective exchange of data, and if necessary of ontology,
- ? exploiting one of the many mapping technologies available within the EXPRESS concept (ISO TC184/SC4/WG11, 1999) to achieve standard data conversions between receiving and sending systems.

The origin of our interest in exchanging data between heterogeneous databases is the need to represent and exchange in computer-sensible manner of catalogues of industrial components. Such a catalogues contain not only static aspects (properties of components) but also dynamic aspects (for example functions which describe components behaviour according to the particular environments where they are inserted).

The aim of our work in this application domain is to represent the whole information in a meta-model used like a central (federated) model to achieve data exchange regardless of the particular used conceptual schema, and of the particular used database systems for implementation.

In this domain, catalogue schema (i.e. its structure of classes) depends on the catalogue itself. So, exchange should involve not only components instances but also schema itself. The generic meta-schema developed, standardised in ISO (ISO 13584-24, 2002) (ISO 13584-25, 2002) and known as PLIB is used to represent the catalogue.

The goal of this paper is to present how this generic meta-schema can be used in other exchange domains, and to outline the approach we developed to achieve a bridge from the generic meta-schema to specific database schemas.

This paper is organised as follows. Section 2 analyses various sources of diversities encountered in databases targeted at the same domain. In section 3, we situate our approach among the work allowing data exchange and sharing. We will present briefly, in section 4, the EXPRESS modelling language used to define our models of data exchange. Techniques and tools usable to achieve models transformations in the EXPRESS environment are also discussed in this section. Section 5 gives the details of the modelling approach we developed to allow data exchange between heterogeneous databases using EXPRESS generic meta-schemas. We discus in section 6 the process of exchange itself, the stages it involves and the various techniques that can be implemented. Finally, section 7 gives the details of the implementation we proposed to restore exchanged data in receiving systems.

## 2 VARIOUS DIMENSIONS OF DIVERSITY

Even when targeted at the same domain, two database schemas may be very different. We analyse below the various dimensions of diversity.

*Denotation diversity:* in each database, entities (when represented), tables and attributes are identified by names. Various names can be assigned to a same concept, or inversely, various concepts can have a same name.

*Semantic diversity:* in a same domain, assuming that same names are used, the modelled concepts may be different. One model may present particular specialisations regarding to concepts existing in another model. Moreover, if some specialisation occurs, it may be based on different specialisation criteria. For example for a *car* one can specialise according to origin (*French, foreign*), type (*racing car, van, convertible…*). Concerning choice of attributes, each category of user and thus each schema may use a particular subset of the possible attributes of one entity.

*Structural diversity:* in the same domain and assuming that the same concepts (entities and

attributes) are used, final database schema may be different from each other. This diversity results from three causes: need to avoid redundancy (normalisation rules), order of table's attributes, and finally DBMS particularities.

Normalisation is intended to remove data redundancy by splitting tables that contain functional dependencies in separate tables. This process may generate structural diversity because tables and relations may be different. Indeed, we can represent *address* of *organisation* in the same table: *organisation (name, number, street, town, zip_code, country, phone, fax)* or in separated tables: *organisation (name, add_id)* and *address (add_id, number, street, town, zip_code, country, phone, fax)*. Removing redundancy has also been considered in the field of industrial components modelling where two schemas have been defined (El-Hadj Mimoune, 2001). Thus leading to structural diversity.

When the same tables are chosen there is no reason why attributes should be arranged in the same order, or all attributes should be used in all databases (example: one can use *fax* attribute other not).

Finally, implementation diversity is due to existence of various DBMS, each one having their own particularities. It is obvious that implementation of a conceptual model described in an Oriented-Object paradigm will not be the same one in a RDBMS as in an OODBMS. For example, for representing a set of first names of one person one would use a relation in a RDBMS and an aggregate in O-O databases. The same applies for inheritance. It will be implemented by relations in RDBMSs, and the ORDBMS which cannot support this mechanism (e.g. Oracle), and by inheritance of classes and tables in O-O systems and O-R supporting the inheritance mechanism (example: Postgres).

# 3 RELATED WORKS

Several work on database addressed ease of exchange, sharing, integration and interoperability between heterogeneous databases

The difficulties that the various contributions aim at addressing relate, on the one hand, on structural heterogeneity which results from the various languages, systems and models that can be used to represent similar information, and, on the other hand, on semantic heterogeneity which results from different denomination and definitions of the concepts taken into account.

For handling semantic heterogeneity, there exist two main approaches (Lakshmanan, 1993):

1- the approach based on a common data model consists in defining a federated data meta-model covering all the concepts involved in one databases at least and then to establish a mapping between each model and the federated model. This approach, which also solves the structural heterogeneity problem, allows a user to access to a set of databases, known as federated database, in a unique language and according to a common schema described in the federated meta-model (Arens, 1993)(Ahmed, 1991)(Landers, 1982). One disadvantage of this approach is the considerable cost of centralized integration of all models, and of the maintenance of the federated data model (Lakshmanan, 1993) (Arens, 1993) (Ling Ling, 2001).

2- the approach based on higher order logic (Krishnamurthy , 1988) (Lakshmanan, 1993) allows simultaneous interrogation of several databases using different ontologies, schemas, and possibly different formalisms. The query languages allow to write requests not only on the data but also on the schema and on databases themselves. The disadvantage of this approach is the difficulty for a user to master of each concept of each database and to be able to formulate its request.

The approach proposed in this paper uses:

- ? for semantic integration, an ontology-based approach where each database may either reference a priori the common ontology or may be mapped, a posteriori, on the common ontology,
- ? for denotation integration, each concept in the common ontology is associated with Globally Unique Identifier (GUI),
- ? for structural integration, translation through a meta-model is used.

Moreover, we have proposed a meta-model of ontology, which can be referenced. It allows databases designers to refer a priori, in a non-ambiguous way to the existing ontology. This model also provides possibility for exchanging ontology specific to each database thus, making them easily accessible to a user at the federated level.

The following section briefly presents the EXPRESS language and the way we propose to use it for exchanging data between heterogeneous databases.

# 4 THE EXPRESS LANGUAGE

## 4.1 EXPRESS concepts

EXPRESS is a data specification standard language (ISO10303-11, 1994). It was developed

initially to define data models of industrial products. It is now used for the modelling data in various domains (Plantec, 1999) (Ait-Ameur, 2000). EXPRESS build on previous work on data models, such as the entity/relationship approach (Peter, 1976), OMT (Rumbaugh, 1991), NIAM (Habrias, 1988) etc. It was defined to make such models, more precise and computer sensible, and to offer a powerful representation of constraints on data. EXPRESS was developed within the framework of the STEP project (Standard for Exchange of Products dated) (Bouazza, 1995). It is not only a conceptual modelling language defining the information tokens to be exchanged, but it is also a data specification language (DDL) (Herbst, 1994) that specify the data to be generated and validated by computers. EXPRESS allows a two levels of representation of information:

- an intentional description. This description, called schema, corresponds to the conceptual schema of one database. It is defined in term of a set of entities, modelled according to an object oriented approach (inheritance, oid, attributes, derivation…). Entities are associated to a set of typed attributes. Lastly, constraints, functions and attribute derivation allow to associate a set-based semantics to the intentional description. Moreover, derivations of attributes can be expressed,
- an extensional description. This description corresponds to the population of one database, for which a specific of representation format has been defined (ISO10303-21, 1994). A population represented in such a format is named physical file. It constitutes of a set of instances in the structure of which conforms to the schema. This description constitutes a particular interpretation, in the logical meaning, of the intentional description (Ait-Ameur, 2000).

In an EXPRESS schema, an entity represents a set of objects having common properties. These properties are modelled by attributes and constraints. The whole entities are gathered in a schema which can be referenced by other schemas. The attributes fields can also be modelled by types.

In the example below, *A* and *B* are entities. They have typed attributes *ai, bi*. The data types are either simple types (*real, integer*), collections of a given types (*a4* attribute), named types (user defined type *mytype*) or entity (*bn* attribute). The *a6* attribute is a derived attribute computed from the values of the two attributes *a1* and *a3*. This derivation function allows expression of data invariants in a functional form. Such a derivation function expression can be replaced by a more complex function written using the imperative part of the EXPRESS language, close to the PASCAL language. This can be used to describe either derivation functions or logical constraints. Logical constraints represent another class of data invariants. They are introduced by WHERE and RULE clauses which respectively describe local entity invariants and global schema invariants.

```
ENTITY A;
         a1: REAL;
         a2: OPTIONAL NUMBER;
         a3: INTEGER;
         a4: SET OF mytype1;
         a5: mytype2;
    DERIVE
         a6: REAL: = a1*a3;
    INVERSE
         a6: B FOR bn;
    UNIQUE
         a3;
    WHERE
         Wr1: EXISTS (a2) OR (a1 * a2 >
    0);
    END_ENTITY;

    ENTITY B;
         b1: mytype;…
         bn: A;
    END_ENTITY;
    TYPE mytype = INTEGER;
    WHERE
         Wr: SELF >0;
    END_TYPE;
```

Inheritance links are expressed by the *SUPERTYPE* and *SUPTYPE* keywords.

```
    Entity E1
         SUPERTYPE OF (E11 ANDOR E12)
         SUBTYPE of (E);
         ….
    END_ENTITY;
```

In the previous example, *E1* is the mother class of *E11* and *E12* and it is a daughter of *E*. Only the *SUBTYPE* clause is mandatory. It allows to specify that one instance can belong simultaneously to one or several super-classes.

The *SUPERTYPE* clause is less usual. It allows to specify, e.g., that an instance of *E1* may be, at the same time, instance of *E1* and of *E12* (ANDOR).

In the physical file entities instances are described by the values of their explicit attributes.

```
#1 = A(Va1, ……, Vai) ;
#2 = B (Vb1,……, #1) ;
```

*Vai* and *Vbi* represent respectively explicit attribute values of *ai* and *bi*. Each instance is associated with an identifier which allows to reference it (*#1* and *#2*). The derived and inverse attributes are not represented in the physical files because they can be directly computed from the EXPRESS schema by the receiving system.

## 4.2 EXPRESS-based tools and technology

Since EXPRESS is associated with a formal syntax and a precise semantics, it has been possible to develop a set of tools to handle EXPRESS models and data. With any EXPRESS schema, are associated:

- ? a standardised access interface to data conforming with this EXPRESS schema: the SDAI (Standard Data Access Interface), available in several language binding,
- ? and a file format for exchanging data conforming with this schema.

Moreover, tools are available that allow the following:

- ? to generate an SDAI confirming to any APIs access,
- ? to create a physical file from the content of any database associated with an SDAI,
- ? and to populate a database associated with a SDAI from any exchanged physical file.

EXPRESS is a set-oriented specification language; it allows to handle collections and sets.

Moreover, the EXPRESS technology also includes the following:

-the (non standard) EXPRESS-C language which allows to program data manipulation by using directly EXPRESS notations,

- the standard EXPRESS-X language which allows to express, in a declarative way, the correspondences between two schemas. It generates automatically a program capable of transforming all the data conforming to one schema into data conforming to another schema.

Finally, the capability to express derived attributes using functions allows event-based programming within the data model (Plantec, 1999)(El-Hadj Mimoune, 2001). We use precisely this approach for implementing data conversions in our proposed approach for heterogeneous databases.

## 4.3 EXPRESS-G

EXPRESS-G is the graphical representation of EXPRESS. It allows a synthetic representation of an EXPRESS schema. This formalism can be used, e. g., in the preliminary phases of data model designs. EXPRESS-G represents the structural and descriptive constructs of the EXPRESS language (classes and attributes) but the procedural constructs (derivation and rules) are not represented. The following example illustrates an EXPRESS-G representation of a simple data model relating to geometrical entities (see figure 1).

In this example *geometric_entity* can be either a *circle* or a *point*. A *circle* has a *centre* and a *radius*,

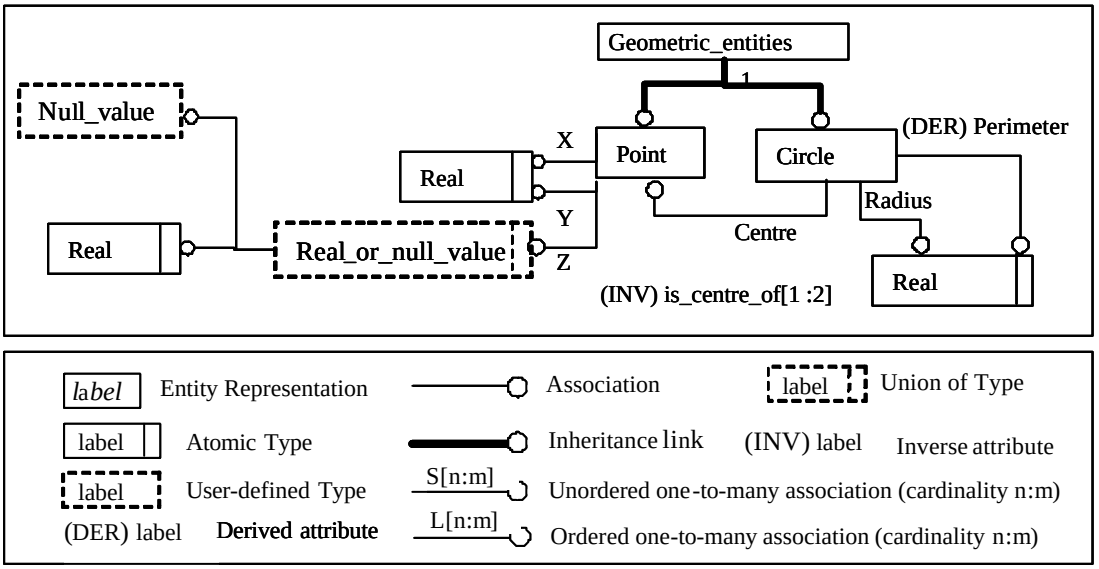


Figure 1: EXPRESS-G representation of geometric entities

and a derived attribute *perimeter*. A *point* has coordinates *X, Y, Z* but *Z* may have either a real value or a null value introduced by the *SELECT* type *real_or_null_value*. A *SELECT* type represents a union of types. Finally in this model, a *point* can be the centre of two circles at the maximum This is specified by the inverse attribute *is_centre_of*.

We described above the concepts necessary for understanding the remainder of this paper. For more information on the EXPRESS language, the reader may consult (Schenk, 1994) (ISO 10303-11, 1994).

# 5 OUR APPROACH FOR INTEGRATION

The approach we propose is summarized by three (simplified) EXPRESS models. We present initially these models, and then we discuss how the various aspects of diversity are taken there into account.

## 5.1 Data models to support exchange

### 5.1.1 Identification of concepts: definition of globally unique identifier (GUI)

In order to avoid the problems of denotation diversity (see 2), a schema defining GUIs has been defined. This Schema allows to identify, through a particular BSU (Basic Semantic Unit) three categories of concepts: sources of concept definition (*supplier_BSU*), entities (*class_BSU*) and properties (*property_BSU*).

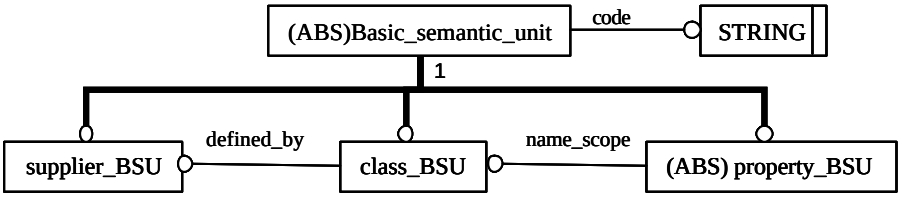


Figure 2: universal identification of concepts

The identifier of each concept inherits *code* attribute (it also embeds a version number not discussed here). Identification of the information source is a simple code, but the manner to assign it, defined in the ISO13584-26 standard, ensures its unicity. The complete identifier of one entity constitutes of a code and of a reference to its information source (by the *defined_by* attribute). Thus, each information source shall ensure unicity of its entity codes to ensure a global unicity of *class_BSU* instances.

Finally, property identification constitutes of a code and a reference to a *class_BSU*. To ensure unicity of this identifier, its information source shall assign unique codes for the attributes of each class it defines.

In the remainder of this paper, it is assumed that shared ontologies are susceptible to exist (or to be developed) for the entities and the attributes of the universe of discourse. It is also assumed that entities and attributes defined in the ontology are identified according to the above schema. In the particular case of the industrial components libraries, such ontologies effectively exists (example: the IEC 1360-4 standard), or are under development for various application domain. The structure of our identification model allows to gather in the same database concepts coming from various ontologies.

### 5.1.2 Exchange of instances: a generic meta-schema

The model used for exchanging instances is an other generic schema allowing to represent any instance of any database, whose entities and attributes are identified by a GUI defined according to the previously presented identification model.

In addition to a reference to the GUI of its class, an instance is represented, by a set of couples (attributes, values).

An attribute is identified by its GUI, and a value is represented like in an EXPRESS physical file.

In order to increase the readability of this paper, the schema below (figure 3) simplifies this model by supposing that the basic types are only *string*, *integer*, *null_value* and entity data types.

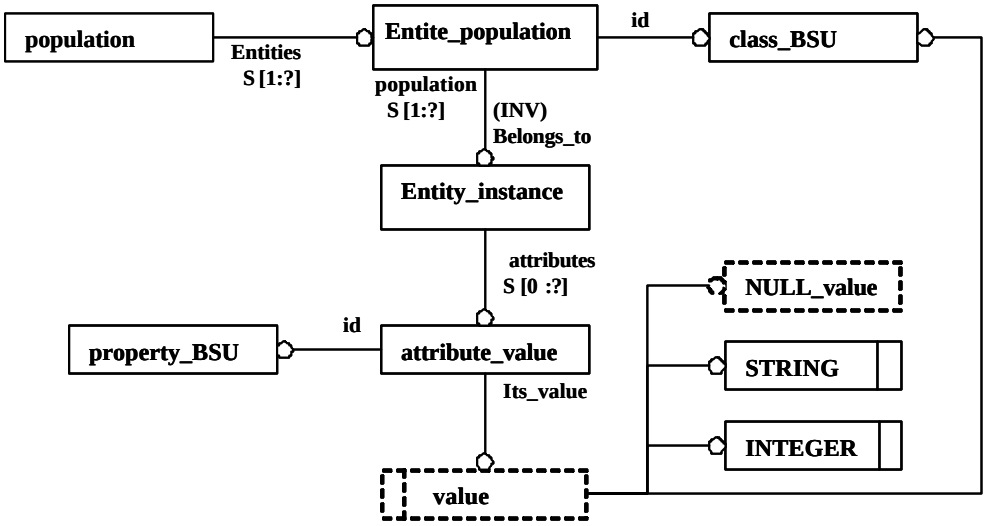


Figure 3: generic meta-schema for instances exchange

This schema is illustrated by an implementation example in the section 6.

### 5.1.3 Exchange of models: a generic meta-schema

Two exchange scenarios may be considered.

First, if all the actors involved in an exchange agree on ontology and on the GUI of the ontology concepts; then using the generic meta-scheme defined in figure 3, it is not necessary to exchange the source system model to be able to interpret exchanged data on the target system.

For example, assume that entity *A*, belonging to the shared ontology, has been subtyped as *A1* in the source system. In this system, each instance of *A1* is described by the attributes *p1, p2., pn* inherited from *A* and belonging to the shared ontology and by attributes *q1, q2, qn* defined by the source system. In this case, the receiving system can decide:

? either to project the *A1* instances on the definition of *A* and to store them in the population of *A*,

? or to create a new entity *A1* subtype of *A* and conforming with the exchanged definition. The definition of *A1* itself may (and should) also be stored in the destination database.

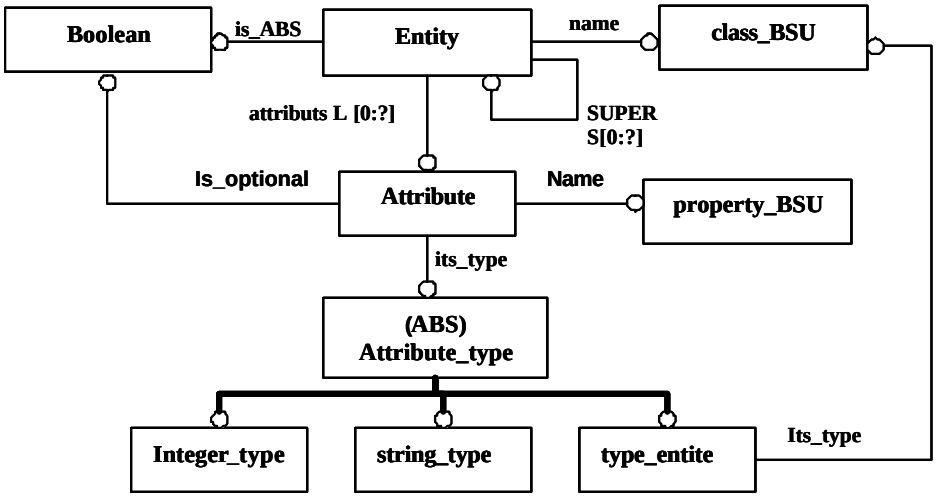


Figure 4: generic meta-schema for models exchange

Second, if the source system contains additional attributes, and/or additional entities, it is necessary to exchange their identifications and their definitions to allow their interpretation and/or storage on the receiving system.

The above schema (figure 4) presents a simplified version of the generic meta-schema allowing to exchange models. For simplicity and

readability, we assume that only integer, and string, and entity data types exist.

In this schema each entity data type is associated with a GUI (i.e. *class_BSU* defined in figure 2) and with a list (ordered) of attributes. The *super* attribute allows representation of inheritance and an entity can be defined as abstract (attribute *is_ABS*). Each attribute is associated with (*property_BSU* defined in figure 2 as well) and with a type *its_type*. An attribute can be mandatory or optional (*is_optinal* attribute).

## 5.2 Taking into account the diversities

This section outlines how the various sources of diversities are taken into account in our proposed approach.

*Diversity of names:* our GUI schema allows to reach the following objectives:

? the same concept, entity or attribute, is always identified the same way,

? a new concept cannot be confused with another concept, and the source of its definition is clearly identified, and

? for a new concept, the definition can be exchanged at the same time as the data which reference it.

*Semantic diversity:*

? any instance belonging to a shared entity data type (i.e. defined in the same ontology) and associated with attributes whose definitions are shared, can be interpreted without any ambiguity on the receiving system whatever be the represented attributes and whatever be the order in which they appear,

? if an entity has been specialised from a shared entity, and if its model is exchanged, it is possible to represent its exchanged instances either like instances of the shared entity which was specialized, or instances of the specialized entity.

*Structural* diversity:

? each entity being represented explicitly and independently of table structure, the particular structure of tables of the source database schema does not appear in the exchange file,

? attribute values being identified by a GUI and not by their position, attributes order has no importance,

? lastly, if the source and the target systems do not use exactly the same sub set of shared attributes for some entity instances, the strategy programmed on the receiving system may, for instance, neglect the additional attributes and associate a null value the missing attributes.

# 6 THE EXCHANGE PROCESS

The exchange process always includes entity instances and possibly entity models, if models are also exchanged.

The exchange process includes five different stages at the maximum, presented on figure 5. Three of them are always present (1, 3 and 5). The most complete process supposes that three EXPRESS models are available:

? the generic meta-schema discussed in section 5.1,

? an EXPRESS model, called "EXPRESS-source", representing the internal schema of the source database in the EXPRESS language (whatever be the DBMS and internal schema)

? an EXPRESS model, called "EXPRESS-target", representing the internal schema of the target database also in the EXPRESS language.

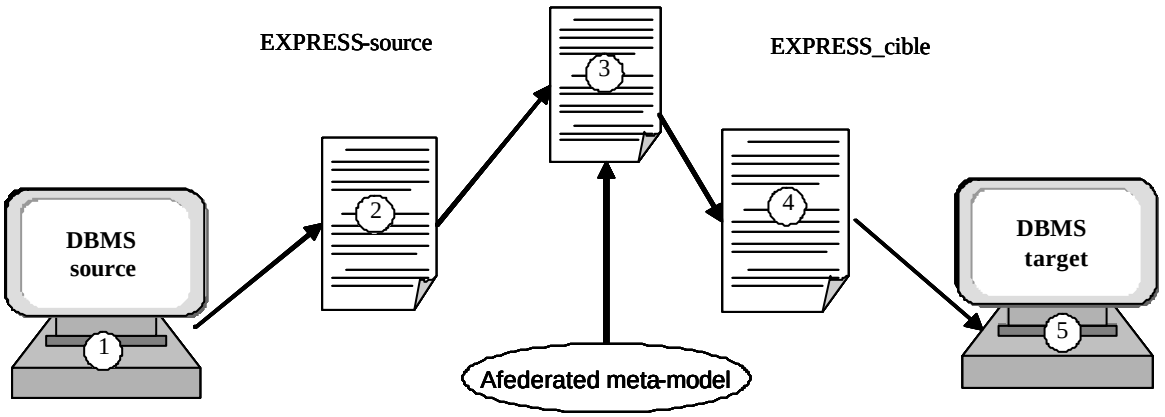


Figure 5: Data exchange process between heterogeneous systems

Then, data pass though five steps taking (at the maximum) five forms (figure 5 above):

1. source data of the source system in the source format (e.g. tuples of RDB, instances of OODB, …),
2. EXPRESS instances of the EXPRESS-source model representing the same content as 1,
3. EXPRESS instances of the federated data meta-model,
4. EXPRESS instances of the EXPRESS-target model,
5. target data of the target system.

The translation from the source database to the EXPRESS-source model is implemented in a programming language supported by the source database. The needed program, generic in nature, exploits the system catalogue of the database (also named metabase). The result is a physical file of the EXPRESS-source model. This file is then converted thanks to one of the conversion techniques available in EXPRESS technology (e.g. EXPRESS-X mapping). Once the instances of the federated meta-model are available, generation of the target data may be achieved either directly, or through a tow step process, using a physical file of the EXPRESS-target model.

In both cases, the last translation from an EXPRESS model instances to data in the target system requires the generation of the suited instructions of data manipulation (INSERT) and possibly, in case of model exchange, of data definition (CREATE).

In order to implement these translation processes, three techniques may be applied:

1. for generating EXPRESS-driven data from source data base native data, using a database-specific language to generate EXPRESS instances confirming to EXPRESS-source or to federated meta-model,
2. for generating DML and DDL instructions in the target database from either EXPRESS-target model or EXPRESS federated meta-mode, using an event-based programming within an EXPRESS data management system, or
3. using the EXPRESS-C language to program the browsing of the EXPRESS population and then generate instructions in the associated DML and DDL (the same statements are generated as in 2).

This last method will be illustrated in section 7.

Notice that this approach addresses structural and descriptive aspects of the databases integration. There is no description of how procedural aspects (queries) are integrated. This work has been addressed in (Ait-Ameur, 2000).

# 7 IMPLEMENTATION EXAMPLE

If the exchanged data correspond only to shared entities and attributes, then, according to the previous sections, the exchange between heterogeneous databases only consists of instances conversion.

In the following, we discus a simplified example of instances conversion. The used instances correspond to the car rental example defined in figure 6.

This schema describes a service of car rental where each service is managed by an organization and has a set of cars that can be rented. Entities *organization* and *car* are described by a set of attributes.

The approach we propose for converting the EXPRESS federated meta-model instances to target database statement consists in adding derived attributes to the federated meta-model defined on figure 3 These derived attributes do not appear in the exchanged physical files (since they are computed by the system at each exchange phase). Their role is to generate SQL statement following an event-based programming approach. Thus the same exchanged file represents licit instances either for the initial schema, defined in figure 3, or for the modified schema that includes derived attributes. If the exchanged file is regarded as a set of instances of the modified schema, it is possible, to compute for each instance the value of the associated derived attribute using an EXPRESS data management system.

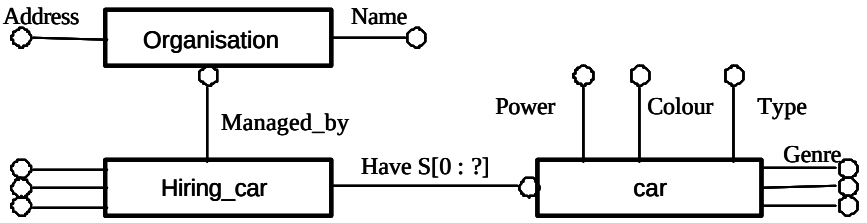


Figure 6: car rental data model

In this approach, the whole set of DDL and DML SQL statement needed to convert of the whole set of instances can be generated (Ait-Ameur, 2000). The next figure presents an example of this approach.

```
ENTITY entity_population;
     Id : class_BSU;
     Population : set [1:?] of
entity_instance;
DERIVE
     SQL_table_name : STRING :=
SELF.Id ;
     SQL_schema : STRING := 'CREATE
TABLE ' + SELF.SQL_table_name +
'(' +
compute_attributes(SELF.population
) + ');' ;
SQL_population : STRING :=
concatenate_SQL_population
(SELF.population);
END_ENTITY ;

ENTITY entity_instance;
     Att_value : LIST [1:?]of
atribute_value;
DERIVE
     SQL_population : STRING :=
'INSERT INTO ' +
SELF.belongs_to.SQL_table_name +
'values ' + assemble_att_value
(SELF.att_value) + ');' ;
INVERSE
     Belongs_to : population_entity
FOR population ; ……
END_ENTIY ;

ENTITY population;
     Entities : set[0:?] of
population_entite;
END_ENTITY;
```

The name of the representation (i.e. the table) is the same as the name of the concept that it represents (i.e. the entity *car*). The attribute *SQL_schema* of the entity *entiy_population* allows to generate the CREATE TABLE statements which correspond to the entity *car*. The compute_*attributes* function computes the attributes and their types, necessary to create SQL tables, by browsing attributes *att_value* (a couple attribute; value) of the entities. It is supposed here that each instance is defined by the same attributes. The attribute *SQL_population*, of the entity *entity_population*, allows to gather all the insertion orders of the various instances of the population by performing the concatenation of the content of the *SQL_population* attribute of each *instance_entity* referenced by the attribute *population*.

The population to be exchanged is represented by the following EXPRESS physical file:

```
#1 = entity_population (#2, (#3)) ;
#2 = class_bsu ('car') ;
#3 = entity_instance ((#6, #7)) ;
#4 = property_bsu ('genre', #2) ;
#5 = property_bsu ('color', #2) ;
#6 = attribute_value (#4,
'Peugeot') ;
#7 = attribute_value (#5, 'red') ;
#8 = entity_population (#10, (#9)
#9 = entity_instance ((#13, #14)) ;
#10 = class_bsu ('car_rental') ;
#11 = property_bsu
('managed_by',#10) ;
#12 = property_bsu ('have', #10) ;
#13 = attribute_value (#11, #15) ;
…….. ;
#14 = attribute_value (#12, (#2)) ;
#15 = entity_instance (#16, (#19
,#20)) ;
#16 = class_bsu ('organisation')
#17 = property_bsu ('name', #16) ;
#18 = property_bsu ('address',
#16) ;
#19 = attribute_value (#17, 'ADA') ;
#20 = attribute_value (#18, '9 wool
street 86000') ;
```

In this case, the representations of entities names (*car*, *organisation*…) and of the attribute names (*name*, *address*…) are simplified to increase readability. In practice, they should be replaced by their GUIs. These identifiers would be defined by the ontologies. Let us note that such ontology already exists in a number of domains such that electronic components (IEC 61360-4 standard) and for products and services classification (UNSPSC for Universal Standard Products and Services Classification). Figure 7 shows a part of the IEC 61360-4 ontological dictionary which identifies at the same time entities and attributes.

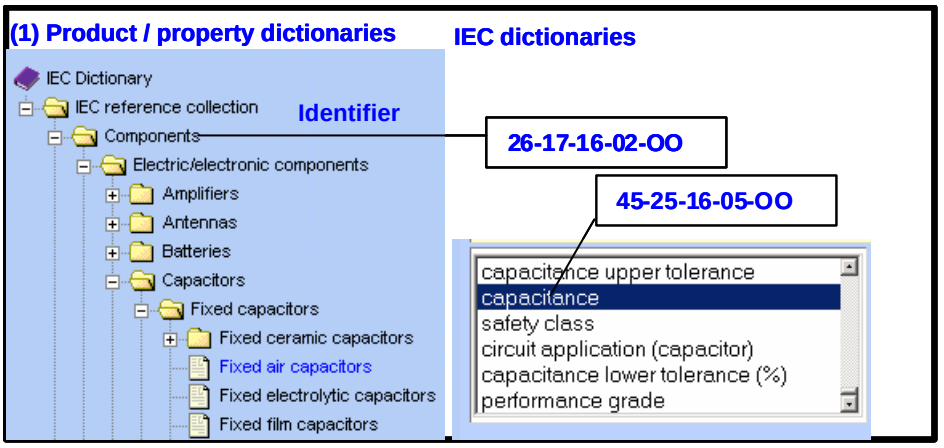


Figure 7: Concepts definition (entities and attribute) In IEC dictionary and their GUIs.

In our example one could reference the UNSPSC ontology to define identifiers for the entities *car* and *car_rental*. Products and services classification in the UNSPSC is illustrated in figure 8. It allows to universally associate the code /0111/4/UNSPC-1.25-10-15-03 to concept "car" and the code /0111/4/UNSPC-1.78-11-18-06 to the concept "car rental" where "/0111/4/UNSPC-1" is the identifier of the UNSPSC classification itself.

Data exchange between heterogeneous databases encounters difficulties mainly due to heterogeneity: structural heterogeneity and semantic heterogeneity. Structural heterogeneity results from the diversity of the used databases management systems and normalisation processes. Semantic heterogeneity results from differences of denomination and of the conceptual models being able to be defined for the same field.

| Segment | Family | Class | Commodity | BTI | Title |
|---|---|---|---|---|---|
| 25 | 0 | 0 | 0 | 0 | Commercial and Military and Private Vehicles and their Accessories and Components |
| 26 | 0 | 0 | 0 | 0 | Power Generation and Distribution Machinery and Accessories |
| 25 | 10 | 0 | 0 | 0 | Motor vehicles |
| 25 | 11 | 0 | 0 | 0 | Marine transport |
| 25 | 12 | 0 | 0 | 0 | Railway and tramway machinery and equipment |
| 25 | 10 | 15 | 0 | 0 | Passenger motor vehicles |
| 25 | 10 | 15 | 1 | 0 | Minibuses |
| 25 | 10 | 15 | 2 | 0 | Busses |
| 25 | 10 | 15 | 3 | 0 | Automobiles or cars |
| 25 | 10 | 15 | 4 | 0 | Station wagons |
| 25 | 10 | 15 | 5 | 0 | Minivans or vans |
| 25 | 10 | 15 | 6 | 0 | Limousines |
| 25 | 10 | 15 | 7 | 0 | Light trucks or sport utility vehicles |
| 25 | 10 | 15 | 8 | 0 | Sports car |
| 25 | 10 | 16 | 0 | 0 | Product and material transport vehicles |
| 78 | 11 | 18 | 6 | 0 | Vehicle rental or leasing |

Figure 8: Cars and hiring car classification in UNSPSC

Moreover, in the domain where such ontology does not exist, a similar description of concepts is accomplished. In this manner the use of the generic schemas presented in this paper would be possible.

# 8 CONCLUSION

In this paper we proposed an approach allowing the take into account the two difficulties outlined previously (i.e. structural and semantic heterogeneities). Data exchanged between databases are related to application domains. Concerning semantic heterogeneity, we proposed an ontology-based approach. A shared ontology allows both to reference it in existing database (a priori integration), and, if needed, to exchange it together with their instance if it is not already shared. To

solve denomination diversity, we proposed to associate each concept of the ontology with a GUI (basic semantic unit) (if models are exchanged).

For structural aspects, we proposed a generic meta-schema susceptible to be used for any exchange between any databases. To identify the exchanged entities and attributes, this meta-schema references the shared ontology. If the ontology is not already shared data and its ontology are exchanged simultaneously. Our three meta-schemas are expressed in the EXPRESS language. This allows to use the various programming techniques available in the EXPRESS technology to perform all the necessary translations. Our approach allows both a simple data exchange (instances level) and exchange of ontologies (if these ontologies need also to be exchanged).

If the meaning and GUI of all the entities and all the attributes on the sending system are known by the receiving system, an instances exchange would be enough. In the opposite case (sending system contains additional entities and/or additional attributes); this proposed approach allows to exchange the ontologies themselves to allow their interpretation and their storage in the receiving system.

The translation from the source database to the federated meta-model can be carried out directly, or by using an intermediate model that is an EXPRESS representation of the source database internal schema. The translation from the federated meta-model to a target database can be carried out either directly, or by using an EXPRESS representation of the target database content. In both cases, this would require generation of the instructions of data manipulation (INSERT) and possibly, of data definition, (CREATE…). In EXPRESS technology various techniques may be used to generate the statements of data definition and data manipulation statements necessary to restore the exchanged content on the target system. In this paper, following (Plantec, 1999) (Ait-Ameur, 2000), we proposed to use the EXPRESS derived attributes technique to carry out this generation. This technique allows to reduce the complexity of the generation programme by splitting it up into elementary fragments where each one is in charge of converting one or a reduced number of entities type.

The approach we proposed in this paper was validated in the field of the industrial components libraries exchange. For this application field we designed a specific DB schema for POSTGRES RODBMS (Stonebraker, 1990). We then showed how an EXPRESS physical file could be automatically converted into data definition and data manipulation statements of this DBMS thanks to use of derived attributes added to the EXPRESS data model.

Future extensions, we plan to study the integration of the procedural aspects already studded in (Ait-Ameur, 1995) (Ait-Ameur, 2000) in order to exchange also constraints.

## REFERENCES


Ahmed, R., Smedt, P. D., Du, W., Kent, W., Ketabchi, M. A., Litwin, W. A., Rafii, A., and Shan, M. C., 1991. The Pegasus Heterogenous Multidatabase System. IEEE Computer.

Ait-Ameur, Y., 2000. Développements Controlés de Programmes par Modélisation et Vérifications de Propriétés, Habilitation à diriger les recherches, Université de Poitiers.

Ait-Ameur, Y., Besnard, F., Girard, P., Pierra, G., Potier, J.C., 1995. Specification and Metaprogramming in the EXPRESS Language. In SEKE'95, Conference on Software Engineering and Knowledge Engineering, IEEE-ACM Sigsoft.

Ait-Ameur, Y., Pierra, G., Sardet, E., 2000. An object oriented approach to represent behavioural knowledge in heterogeneous information systems. In OOIS 2000, Proc. of the 6th International Conference on Object Oriented Information Systems, Springer.

Arens,Y., Chee, C., Hsu, C.-N., and Knoblock, C.A., 1993. Retrieving and Integrating Data from Multiple Information Sources. International Journal on Intelligent and Cooperative Information Systems, Vol. 2.

Atzeni, P. Torlone, R., 1997. MDM: a Multiple-Data-Tool for the Management of Heterogeneous Database Schemes. Proceeding of International Conference on Management Data and Symposium on Principales of Database Systèmes. ACM SIGMOD, Tucson.

Bouazza, M., 1995. la Norme STEP. Hermes.

Castano, S., De Antonellis, V., , 1997. Semantic Dictionary Design for Database Interoperability. Proceeding of the 13th International Conferce on Data Engeneering, IEEE Computer Society Press.

Dittrich, K. R. and Geppert, A, 1997. Object-Oriented DBMS and Beyond. In proceeding of the Conference on Current Trends in Theory and Practice of Informatics, .

El-Hadj Mimoune, M. Ait-Ameur, Y. Pierra G. and Potier, J.C, 2000. Integration of component description in product data management systems. In CE 2000, 7th ISPE International Conference on Advance in Concurrent Engineering, , Technomic Publ. Co..

El-Hadj Mimoune, M., Ait-Ameur, Y., Pierra, G., 2001. Modélisation du contenu des catalogues de composants industriels : de la représentation implicite à la représentation explicite. IN ISPS'2001, 5th

International Symposium on programming and Systems, SERIST.

Gruber, T.~R., 1993. Towards Principles for the Design of Ontologies Used for knowledge sharing. In Formal Ontology in Conceptual Analysis and Knowledge Representation, N.~Guarino and R.~Poli, Eds., Kluwer Academic Publisher's.

Habrias, H., 1988. Le modèle relationnel binaire. La méthode NIAM. Eyrolles.

Herbst, A., 1994. Long-Term Database Support for EXPRESS Data. In 7th Int'l. Working Conf. on Scientific and Statistical Database Management, IEEE Computer Society Press.

ISO 10303-11, 1994. Industrial Automation Systems and Integration -- Product Data Representation and Exchange -- Part 11: Description methods: The EXPRESS language reference manual.

ISO 10303-21, 1994. Industrial automation systems and integration - Product data representation and exchange - Part 21: Implementation methods: Clear text encoding of the exchange structure (Physical file). ISO document.

ISO 10303-22, 1997. Industrial automation systems and integration — Product data representation and exchange – Part 22: Implementation methods: Standard Data Access Interface.

ISO 13584-25, 2002. Industrial automation systems and integration -- Parts library -- Part 25: Logical resource: Logical model of supplier library with aggregate values and explicit content.

ISO TC184/SC4/WG11, 1999, Industrial Automation Systems and Integration -- Product Data Representation and Exchange, The EXPRESS-X Language Reference Manual.

Krishnamurthy R. and Naqvi, S., 1988. Towards a Real Horn Clause Language. In Proceedings of the International Conference on Very Large Data Bases, Morgan Kaufmann Publishers.

Lakshmanan, L.V.S., Sadri, F., and Subramanian, I.N., 1993. On the logical foundation of schema integration and evolution in heterogeneous database systems. Proc. Int. Conf. on Deductive and Object-Oriented Databases, Springer-Verlag, Phoenix.

Landers, T., and Rosenberg, R., 1982. An overview of multibase. Distributed Databases. Proc. 2nd Int'l Conf. on Distributed Databases, H-J Schneider.

Ling Ling, Y., Renee M., and Laura H., 2001. Data-driven understanding and refinement of schema mappings. Proc. ACM SIGMOD Conference, ACM Press.

Panet, G., Letouche, R., 1994. MERISE/2, MODELES ET TECHNIQUES MERISE AVANCES. Les éditions d'organisation.

Peter Pin-Shan Chen, 1976. The Entity-Relationship Model. Toward Unified View of Data. ACM Transaction on Database System.

Pierra, G. Ait-Ameur Y. and Sardet, E, 2002. Industrial Automation Systems and Integration, Parts Library, Logical Model of Supplier Library. ISO 13584-24.

Pierra, G. Wiedmer, H. U., 1998. Industrial Automation Systems and Integration, Parts Library, Methodology for Structuring Parts Families. ISO 13584-42.

Pierra, G., 1994. Modelling classes of prexisting components in a CIM perspective: The ISO 13584/ENV 400014 approach. Internationale revue of CAD/CAM and Infographie.

Pierra, G., 2000. Représentation et Echange de données techniques, Mécanique et Industrie, Mec Ind, 1.

Plantec, A, 1999. Utilisation de la norme STEP pour la spécification et la mise en œuvre de générateurs de code, thèse, Université de Rennes 1.

Rambaugh, J., Jacobson, I., and Booch, G. , 1999. The Unified Modeling Language Reference Manual. Addison-Wesley.

Rumbaugh, J. Blaha, M. Premerlani, W. Eddy,F. and Lorrensen, W, 1991. Object-Oriented Modelling and Design. Prentice Hall, International Edition.

Sardet, E., Pierra, G., Murayama, H., Oodake, Y. and Ait-Ameur, Y., 2001. Simplified Representation of Parts Library : Model, Practice and Implementation.IN proceeding of PDT Days, QMS edition.

Schenk, D. A. and Wilson, P. R, 1994. Information Modeling: The EXPRESS Way. Oxford University Press.

Schreuber, T., Wielinga, B. and Breuker, J., 1992. KADS: A Principled Approach to Knowledge-based System Development. Academic Press.

Stonebraker, M., Rowe, M., and Hirohama, L., 1990. The implementation of Postgres. IEEE Trans. on Knowledge and Data Engineering.

Sudarshan Chawathe et al., 1994. The TSIMMIS Project: Integration of Heterogeneous Information Sources. In Proceedings of IPSJ Conference.

Vermeer, M. W. W. and Apers, P. M. G.,1996. On the Applicability of Schema Integration Techniques to Database Interoperation. In ER'96, Proceedings of Fifteenth International Conference on Conceptual Modelling, Cottbus.