THESIS / THÈSE

MASTER IN COMPUTER SCIENCE

Automatic generation of data converters

Lebailly, Christophe; Rosmant, Anne-Sandrine

Award date:
1999

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX, NAMUR

INSTITUT D'INFORMATIQUE

RUE GRANDGAGNAGE, 21, B-5000 NAMUR (BELGIUM)

Année Académique 1998-1999

Automatic generation of data converters

Christophe Lebailly
Anne-Sandrine Rosmant

Mémoire présenté en vue de l'obtention du grade de

Maître en Informatique

Abstract

In this paper, we attempt to propose a tool that eases the data migration among two databases or among a database and a data structure under textual format (XML). Our purpose is to implement the program which automatically generates the code for what we call the converter. The program that creates the converter bases itself on the semantic correspondences between the schemas representing the databases and the XML document. We explain the way we represent and analyse these correspondences (or mapping), in order to generate the conversion program.

Résumé

Dans ce document, nous tentons de proposer un outil qui facilite la migration de données entre deux base de données ou entre une base de données et une structure de donnée textuelle (XML). Notre but est d'implémenter le programme qui génère automatiquement le code de ce qu'on appelle un convertisseur. Le programme qui crée le convertisseur est basé sur les correspondances sémantiques qui existe entre les schémas représentant les bases de données et le document XML. Nous donnons une explication sur la manière dont l'analyse et la représentation de ces correspondances (mappings) sont effectués afin de pouvoir générer les programmes de conversion.

Acknowledgements

First of all, we want to thank Philippe Thiran for his kind attention and his priceless help. We also want to thank our supervisor Jean-Luc Hainaut who followed us all along the writing of this thesis. We add to our acknowledgements Majid for his everlasting smile and his good advises, Miguel, Droli, Michela and all the friends who gave us help on the completion of our work (sorry, we could not mention everyone).
We don't want to forget professor Arne Sølvberg, Terje Brasethvik, Hallvard Trætteberg, Tom Henriksen and all the people at NTNU Trondheim for their kindness and the welcome they gave us. We have to add a special thank to Mats and Brage, without them our stay in Norway would not have been so pleasant.
And finally, a very special thank to Mr Scott Adams who gave us inspiration.

"And how are these Java extractors?"

Terje

Table of Contents

Introduction

Introduction

Subject

The purpose of this thesis is the implementation of a program that automatically generates a data converter. The aim of this generated converter is to transfer the data from one data structure to another. Of course, the data to be transferred must be relevant, that is to say that there must be a corresponding structure to host them in the target structure. That is precisely one of the goal of our program that has to take a look at the schemas representing those data structures and find the correspondences between them.

We develop two cases of converter generation :

- Data conversion between two databases
- Data conversion between a database and a data structure under textual format (XML)

Context

Many organisations legacy databases are nowadays obsolescent and ill-structured. Hence, the need to migrate to newer devices. But this migration is not an benign operation, most of the time it looks like a high precision surgery that can only be done by human work. Moreover, sometimes most of the data are solely an artefact of the implementation and need not to be migrated, like a transplant surgery when you only need the vital organs to be taken.
It would be, thus, very effective to have tools able to extract the data from the legacy database, drop the waste, validate the relevant data against the new Database Management System (DBMS) and load them into it.

One of the biggest problem to handle is the semantic interoperability between the databases. Most of the time, these are heterogeneous (physically and/or semantically) and understanding the semantic at the schema or data levels is a huge work. Once again, it would be interesting to automate the process as much as possible, provide tools to assist the human trying to understand and map one or more schemas from the legacy database service to the target.

We are in a time where globalisation becomes a usual word, organisations are nowadays installed all over the world. These enterprises have thus to send their data from one place to another. One easy way to achieve this is by the way of Internet, that is why the transferred data may be stored in a textual format specially adapted for the Net, called XML.

Method

The method we use to implement the generator is the following :

- First, we analyse the schemas to retrieve the correspondences (mappings)
- Then, we store the information we gathered at the previous step
- Finally, we analyse both the target schema and the mappings to produce the generator

Thesis organisation

The division in chapter of our thesis follows this progression :

In chapter 1, we present a « state-of-the art » which defines some useful terms and gives a detailed view of the problem statement and the context of our work.

Chapter 2 is dedicated to a description of the models we use, that is to say the Entity-Relationship model, a limited one, the conceptual model and the XML model.

In the third chapter, we develop the generator architecture which is divided in two cases : DB-to-DB and DB-to-XML. We also detail the converters support tools.

The fourth chapter is devoted to the converter generation methodology. Here again, the methodology is divided in two cases, depending on the type of converter that has to be generated.

In chapter 5, we first detail the way we analyse the mappings between the source and the target schema. Then, we bring to light the manner we store the information given by the mapping analysis.

Chapter 6 is dedicated to the explanation of the way we generate the DB-to-DB converter, on the basis of the mappings analysis

The seventh chapter presents the description of the way the other case of the converter (DB-to-XML) is generated.

Finally, in the ninth chapter we develop a double case study for the generation of source code designed to transfer data, first from a DB to another and then from a DB to a XML document.

Chapter 1
State-of-the-art

Chapter 1 : State-of-the-art

This chapter deals with the current state of research in the data migration domain. We first define the notions of data migration, schematic heterogeneity and schema translation. Finally we give the position of this thesis in that domain.

1.1. Data Migration

Data migration[1] among heterogeneous databases has become a huge pole of interest as many organisations need to migrate their data from their old legacy databases (DB) to newer devices.

Legacy databases tend to be ill-structured and vast. Moreover, a considerable amount of data is solely an artefact of the implementation and need not to be migrated. It would be very effective to have tools that could extract data from the legacy DB, validate it against the migration DB, ignore it if it were not relevant, translate relevant data to the required formats, and load it into the target DB Management System (DBMS).

> *« Semantic interoperability is one of the least appreciated challenges in legacy database migration and also one of the most costly to resolve. Understanding the semantics of one database at the schema and data levels is a massive job. »*
> [Brodie,1995]

It is a job performed by human intervention for the most part. The more interesting enhancement you can bring here, is to automate the process as much as possible, provide tools to assist the human trying to understand and map one or more schemas from the legacy DB service to the target.

XML is a subset of Standard Generalised Markup Language (SGML) that is optimised for delivery data over the Web. It provides a universal method for describing data and because of its very structure, XML can be used as an intermediate format during a data migration. Indeed, a XML document contains both the data and their representation, it facilitates thus more precise declarations of content and more meaningful search results across multiple platforms.

[1] The process of translating data from one format to another

1.2. Database Heterogeneity

1.2.1. Heterogeneity

« A information system is called *homogeneous* if the software that creates and manipulates the data is the same at all sites. Furthermore, all data follows the same structure and format (data model) and is a part of a single universe of discourse. In contrast, a *heterogeneous* system is one that does not adhere to *all* the requirements for a homogeneous system. This means that any dissimilarity at any level in the information system design and implementation requires that the system be called heterogeneous. In this respect, heterogeneity can happen at all levels of the database system. For instance, different sites may use different languages to write applications, different query languages, different models, different DB Management Systems (DBMS), different file systems, etc. The more dissimilar the two systems are, the more difficult it is to bridge that heterogeneity. » [Sheth, 1999]

1.2.2. Problem statement

One of the biggest problem for most large corporations is that their data are stored in heterogeneous and independent databases that may have been developed on different times, different platforms, different DBMS and by different people who may have different « views of the real world ».

Large organisations' legacy IS must support their current business requirements, under penalty of seeing their competitiveness strongly decreasing. Therefore, it seems necessary to upgrade their systems and one way for achieving this is to integrate their different DBs. A first positive consequence to such an integration will be the important savings in maintenance. Another solution would have been to reengineered the whole but this is a really high financial cost. The replacement by a unique system, as for him, would be really expensive in terms of reorganisation costs. From this, it seems necessary to develop tools that allow users and application programs to access the multiple DBs as if they were a unique and homogeneous one.

« Accessing and managing data from such heterogeneous databases pose complex problems that can be classified into platform, Data Management System (DMS), location and semantic level. » [Thiran, 98]

Platform problems come, inter alia, from the fact that the different DBs reside on different machines, run under different operating systems and use different network protocols.

DMS problems are due to the diversity of existing DMS. Nevertheless, these can be easily solved for some families of DMS by the way of tools like JDBC[2] or ODBC[3].
The location problem resides in the difficulty for the user to know where the data are stored. And the semantic difficulties are due to the multiple way a unique object can be represented.
Some authors divide semantic conflicts into two kinds of difficulties [Kim, 1991], those that arise from data heterogeneity and those that come from schematic heterogeneity. The latter is the more interesting and will be developed more thoroughly in the next section. We can say briefly that schema conflicts result from the use of different schema definitions in different DBMS. Data conflicts are due to inconsistent data in the absence of schema conflicts.
The two types of data conflicts listed are : wrong data and different representations. The former is generally due to failures in maintaining a DB up to date (and failures to enforce integrity constraint) and to the fact that some attributes are expected to have the same value and actually, have not (this problem is also called incorrect-entry data). A query on a view comprising these two attributes will give a wrong answer. E.g. if the address of a person in a DB is different from that of the same person in another DB, a query for the person's address will return a wrong or corrupted information.
The different representations conflict can be easily understood by giving some meaningful examples : conflicts can occur when different words are used for the same data (E.g. : Texas, TX or Tx), or different codes (E.g. : ****, excellent or A) and even different types (E.g. : some DB store some particular data as string meanwhile other store them as integer). The difference can also come from different units used (E.g. : cm or inches, degrees or gradient).

We will not go further in developing data heterogeneity conflicts as we will concentrate on schematic heterogeneity conflicts.

1.3. Schematic heterogeneity

1.3.1. Definition

There are different definitions given in the DB heterogeneity literature. R.J. Miller in [Miller, 1998] defines schematic heterogeneity like :

---

[2] Short for ***Java Database Connectivity,*** a Java API that enables Java programs to execute SQL statements. This allows Java programs to interact with any SQL-compliant database. Since nearly all DBMSs support SQL, and because Java itself runs on most platforms, JDBC makes it possible to write a single database application that can run on different platforms and interact with different DBMSs.

[3] Abbreviation of ***Open DataBase Connectivity***, a standard database access method developed by Microsoft Corporation. The goal of ODBC is to make it possible to access any data from any application, regardless of which DBMS is handling the data. ODBC manages this by inserting a middle layer, called a database *driver*, between an application and the DBMS. The purpose of this layer is to translate the application's data queries into commands that the DBMS understands. For this to work, both the application and the DBMS must be *ODBC-compliant* -- that is, the application must be capable of issuing ODBC commands and the DBMS must be capable of responding to them.

« Two schemas are schematically heterogeneous if data under one schema corresponds to database or schema labels in the other. Schematic heterogeneity arises frequently since names for schema constructs (label within schemas) often capture some intuitive semantic information. »

E.g., a stock class may have subclasses, one for each company, where the name of the companies serve as labels for the subclasses.
We do not privilege this definition because it seems to us that it is too restricted. We rather use the interpretation given by Kim and Seo in [Kim, 1991] :

« Schema conflicts result from the use of different schema definitions in different DB's »

This definition seems to be more general and anyway less restrictive than Miller's view, but it needs some further explanation. There are two basic causes of schema conflicts. The first is the difference in the structures used to represent the same information (tables or attributes). E.g., some DB's may represent the address of a person as an attribute of this person while others may represent it in a separate table. The second main cause of schema conflicts is the use of different specifications for the same structure. We can enumerate some examples like the use of different names, data types, and constraints for semantically equivalent tables and/or attributes.

1.3.2. Schematic heterogeneity conflict classification [Kim, 1991]

The schematic heterogeneity conflict classification is developed for conflicts between relational DB's. A global view of the classification is shown in figure 1.1. Each conflict is explained below.

Schema Conflicts Classification
- A. Table-vs-table conflicts
  - 1. One-to-one table conflicts
    - a. Table name conflicts
      - 1) Different names for equivalent tables
      - 2) Same name for different tables
    - b. Table structure conflicts
      - 1) Missing attributes
      - 2) Missing but implicit attributes
    - c. Table constraint conflicts
  - 2. Many-to-many table conflicts
- B. Attribute-vs-attribute conflicts
  - 1. One-to-one attribute conflicts
    - a. Attribute name conflicts
      - 1) Different names for equivalent attributes

2) Same name for different attributes
b. Default value conflicts
c. Attribute constraint conflicts
1) Data type conflicts
2) Attribute integrity-constraint conflicts
2. Many-to-may attribute conflicts
C. Table-vs-attribute conflicts

Figure 1.1. Schema conflict classification

Table-vs-table conflicts

These conflicts occur when different DB's use different definitions to represent information in tables.

One-to-one table conflicts

- Table name conflicts
  These conflicts come from the use of different table names in DB's. There are two types of these conflicts : those due to the use of different names to represent semantically equivalent tables and those due to the use of similar names to represent semantically different tables.

- Table structure conflicts
  These conflicts occur when the number of attributes between corresponding tables is different. There are two interpretations for missing attributes : either the attribute is actually missing, or it is implicit and thus can be deduced. To illustrate the latter, let's take the example of a DB with a table *person*. This table contains several attributes, including the gender. Let's suppose another DB with a corresponding table *person* but with the gender attribute missing. If we assume that this DB represent the members of a Japanese sumo fighting club, the attribute gender is here implicit and has a default value *male*.

- Table constraint conflicts
  These conflicts arise from differences in the specification of table constraints. If an attribute is a primary key in a DB table and is a foreign key in another DB table, there can be some problems transferring data from a DB to another.

Many-to-many table conflicts

These conflicts occur when DB's use different number of table to represent the same information.

Attribute-vs-attribute conflicts

These conflicts are caused by different definitions for semantically equivalent attributes in different DB's.

One-to-one attribute conflicts

- Attribute name conflicts
  Attribute name conflicts are similar to the table name conflicts discussed earlier, but on the attribute scale.

- Default value conflicts
  These conflicts arise from the confusion that may be done if default value are not the same from a table to another.

- Attribute constraint conflicts
  These can be decomposed in data type conflicts and in attribute-integrity constraint conflicts. The formers are due to the use of different data types to represent semantically equivalent attributes in different DB's. The latter come from the possibly different definitions of attribute integrity constraints defined by a Check clause.

Many-to-many attribute conflicts

These conflicts are similar to the many-to-many tables conflicts, but on the attribute scale.

Table-vs-attribute conflicts

These conflicts occur if some DB's use a table and others use an attribute to represent the same information.

1.4. Schema translation

Schema translation is often combined with DB integration or view integration in a heterogeneous environment, so we first need to introduce these terms before developing the schema translation

1.4.1. Database Integration

DB integration or global schema design is a process that takes several, possibly heterogeneous, schemas (i.e. local schemas) and integrates them into a view (i.e. global schema) that provides a uniform interface for all the schemas [Batini, 1986].
If the local schemas are specified in different data models, they may be *translated* into a common model before integration is performed. [Ioannidis, 1993]. (Figure 1.2)

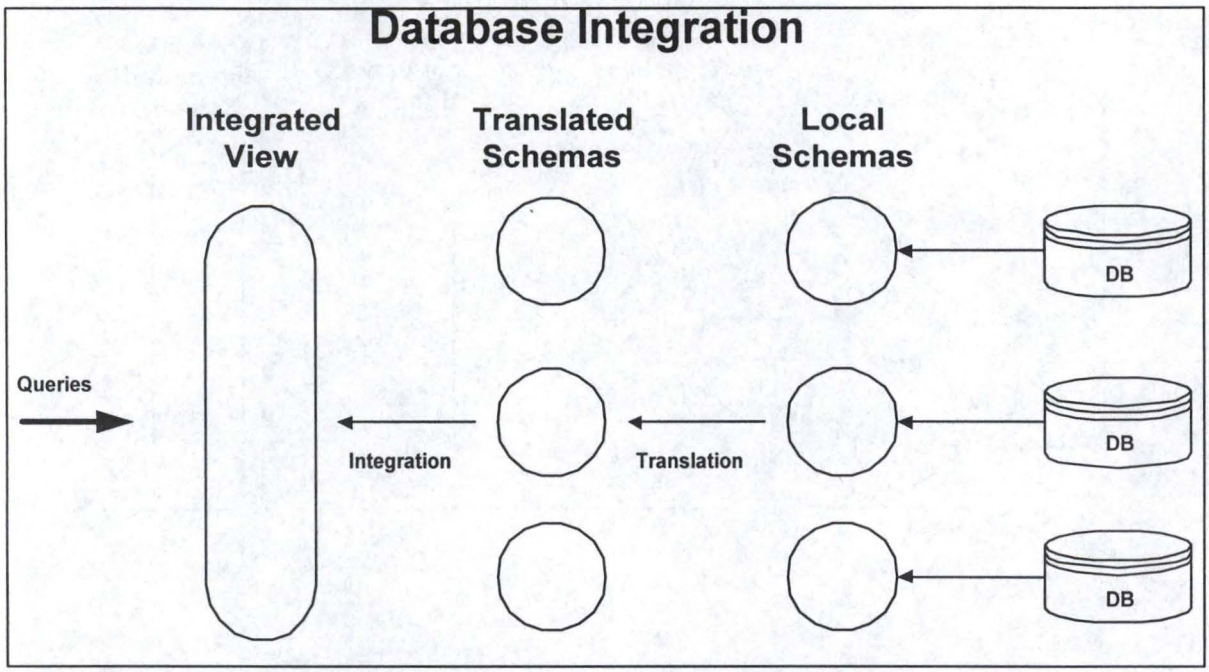


Figure 1.2. : DB integration

1.4.2. View Integration

View integration or logical DB design is a process that takes a set of user views and logically integrates them into a single conceptual schema [Batini, 1986].
These views contain requirements for the portion of a DB that interests different users. The result of their integration is the schema for an actual DB. If the views are specified in different data models, they may be translated into a common model before the integration is done. Depending on the model used for integration, the resulting conceptual schema may also be translated into a target schema in a final step. For example, the Entity-Relationship model is commonly used for schema integration and the relational model used for the target schemas. (Figure 1.3.)

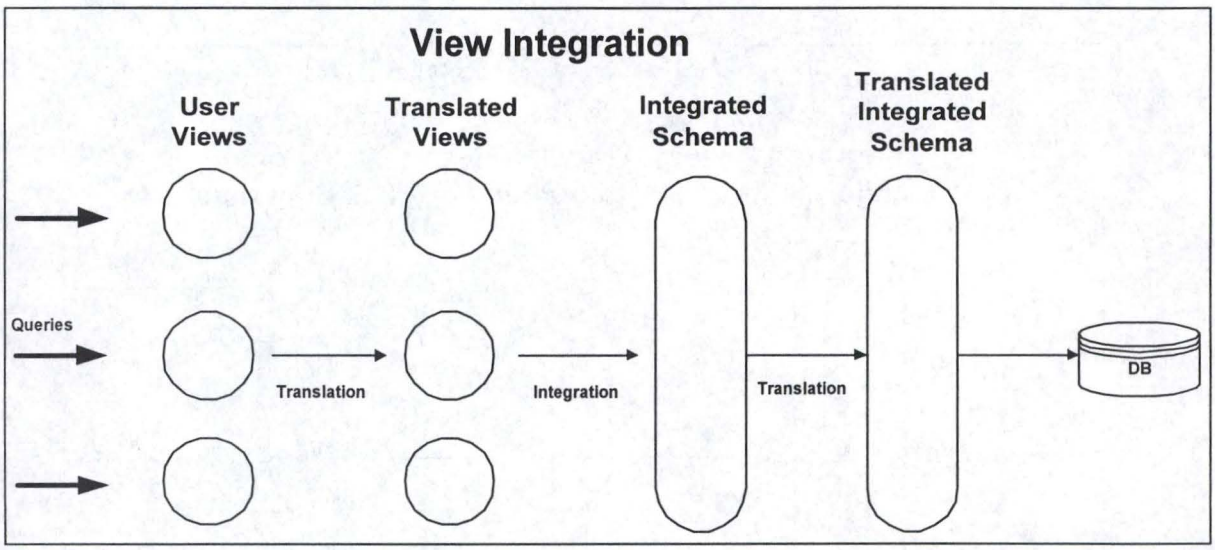


Figure 1.3. : View integration

1.4.3. Schema Translation

As explained above, schema translation is often combined with DB integration or view integration as a part of the integration process. But schema translation can also be required in situations not involving integration [Ioannidis, 1993]. In a unidirectional system (for example, a data transfer between two DB's), where we have only two schemas S1 and S2, the mapping[4] from S2 to S1 involves only a translation. If the aim, in this case, is to populate the target DB from the source DB, then it is compulsory that S2 be at least a semantic subset of S1. If the system has to be bi-directional (transfer in both ways) the schemas must be semantically equivalent (which is not the case in an unidirectional system).

1.5. Position of our work

Our thesis is mainly dedicated to problems of data migration among heterogeneous DB's and of schema translation.

1.5.1. Migration of data among heterogeneous database

The aim of our work is to develop a tool that automatically generates the conversion program that performs the data migration either between DB's, or between a DB and a XML document.

[4] « Mappings are functions that correlate the schema objects in one schema to the schema objects in another schema » A. Sheth

In the DB-to-DB conversion case, we face a problem of schematic heterogeneity, that is to say that the schemas that represent the source and the target DB's are both E-R schemas, have some correspondences, but the information is stored in the DB's under different data structures. The DB-to-XML case is fairly similar, excepted that the underlying models are heterogeneous (a physical DB and an XML document).

1.5.2. Schema translation

In both cases, the schemas that represent the target DB or XML document have been translated from the schema attached to the source DB. Up to us to retrieve how it has been transformed in order to find out the correspondences between the two schemas.
In the DB-to-DB part, the InterDB support case helps us to work only on the DB conceptual layer, so we don't have to worry about the conceptual to physical conversion/translation.
In the DB-to-XML part, we have to manage another type of translation, that is to say, the conversion of a E-R model based schema to an XML tree-structured document. Here again, the InterDB tool gives us the ease to work on a source conceptual schema.

Chapter 2
Models

Chapter 2 : Models

A DB is generally described by three schemas : the conceptual, the logical and the physical schemas. The conceptual schema is a data independent, with high semantic, data representation. The logical schema structures the data like the user and the programmer are used to work with. Finally, the physical schema is the physical data structure.

In the data transfer process, we work with conceptual schemas, whether it is a source or a target schema. The DB schemas respect the Conceptual Model (CM), while the XML model has its particular constructs. All these schemas are based on the Entity Relationship (ER) formalism. A query language exists in order to make queries on CM schemas. In this chapter, we first give an introduction to the ER formalism, then we define the CM valid constructs. Next, we introduce XML and define the XML model. Finally, we show how to make queries on such schemas.

2.1. Entity-Relationship Model

Conceptual schemas are often based on the Entity-Relationship (ER) model. Indeed, this model offers constructs with high semantic that can easily be understood and manipulated by the user that wants to represent reality.

In the ER model, the application domain is seen as an *entity* set. Entities are *related* one another and are characterized by *attributes* that describe their inner properties. In this section, we give a definition and the graphical representation of the *entity type*, *attribute* and *relationship type* concepts. [Hainaut, 94]

2.1.1. Entity type

Reality is perceived as a set of entity. Each entity has the same characteristics and belongs to an entity type (ET). An ET can correspond to real objects (E.g. : Movie) or to abstract concepts (E.g. : show).

ET are represented by a rectangle with the name inside.

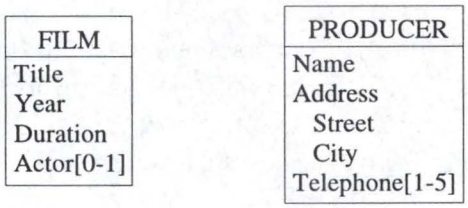
Example :

2.1.2. Attribute

Each ET has some characteristics, its attributes. Each attribute has a type such as numeric or character. Attributes are optional or mandatory. Optional attributes may not have a value for some entities, the other, no. Usually, attributes have only one value, but sometimes, they can have many, these are multivalued attributes. Some attributes may be composed of other attributes, they are called compound attributes, the other, atomic.

Attributes are written in the rectangle of their corresponding ET. The attribute cardinality is written in square brackets [i-j] where i is the minimum cardinality and j the maximum cardinality. E.g, if the cardinality is [0-1] that means that the attributes is optional. The [1-1] cardinality is implicit, it is not represented on the schema. That is the reason why nothing is written next to single-valued and mandatory attributes.

Example :



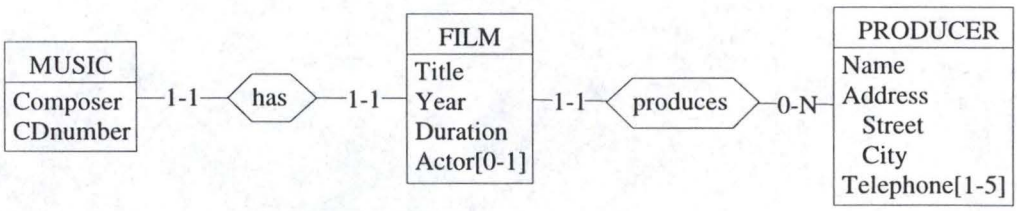
A film has a title, a release year, a duration and may have one actor (or actress). A producer is characterized by its name, address and telephone number. The address is composed of the street and the city name. The producer has between one and five different phone number.

2.1.3. Relationship type

ET's are linked via relationship types (RT). RT may have some attributes and they all have cardinalities. There is two kinds of RT, the one-to-many and the one-to-one. Let A and B be two ET linked with a RT R. If R is one-to-many, there must be zero or one A for each B and each B has many A via R. On the contrary, if R is many-to-many, each A has many B and each B has many A via R. A RT that links two ET's is called binary while RTs that join more than two ET's are called n-ary.

The graphical representation of the RT is an hexagon with the name of the RT inside.

Example :



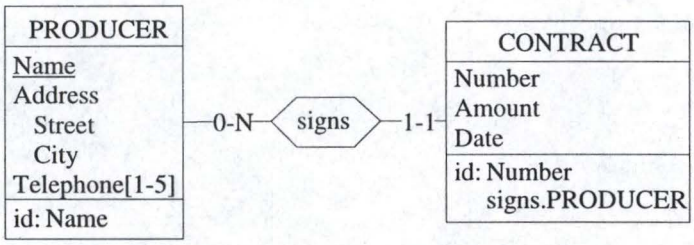
A film has one music and the music is for that film only. Each film has only one producer but the producer may produce several films.

2.1.4. Identifier

An ET can have an identifier. The identifier value is always different for each entity which means that, given an identifier value, there is never more than one entity with that value. Usually, this identifier is one attribute but it can be composed of several attributes and/or a role in a RT. If the ET has several identifier, one is called the primary identifier and the others are the secondary ones.

If the identifier is composed of attributes, the attributes are underlined. On the other hand, if it is made up of at least a role, it is written in the bottom of the ET.

Example :



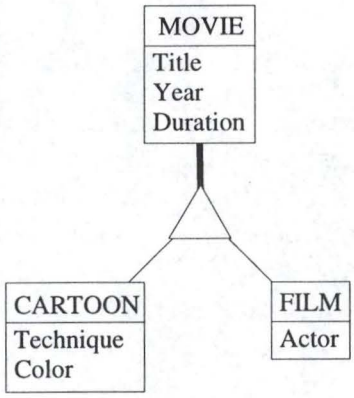
Each producer has a different name. Several contracts may have the same number but it is not possible to find ones signed by the same producer with the same number.

2.1.5. Is-a relationship

An ET may be a subtype of another ET. The subtype inherits all the characteristics of the supertype, i.e. its attributes, RT and identifier. We say that there is an is-a relationship between the ET's. An ET may have several subtype.
The are three particular is-a relationships : total, disjoint and partition. If it is total, each supertype is at least one of the subtype. In a disjunction, the supertype can be only one of the subtype. The partition is total and disjoint, i.e. each supertype has one and only one subtype.

The graphical notation is a triangle that links both ET. The is-a relationship type is written into the triangle. P stands for the partition, T, for total and D for disjoint. If nothing is specified in the triangle, that means that the is-a relationship has no special characteristic.

Example :



The cartoon and film are a specialization of a movie. They inherits its attributes title, year and duration. A cartoon is characterized by the technique and the color whereas the film has one actor (actress).

2.2. Conceptual model

One negative aspect of the ER model is the ambiguity that it creates. Indeed, this high semantic model provides a profusion of constructs that allow the same information to be represented in different data models. This ambiguity may create some schema conflicts just as it is exposed in chapter 2. In order to restraint the risk of having such problems, in the transfer process, the source and target schemas we work with are not full ER schemas but rather simplified ones. Indeed, some constructs are not authorized. Let's name this limited ER model the Conceptual Model (CM). However, this restriction has not too many consequences for the user considering that an ER schema can be transformed into an equivalent CM schema (A transformation plan is given in appendix A).

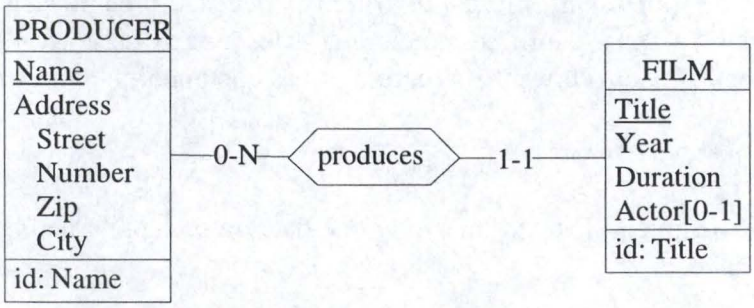
In CM, there are ET's, RT's and attributes. Table 2.1. shows the constructs with their constraints.

Constructs	Constraints
Entity type	Any number of attributes and identifiers
Attribute	Atomic or compound, mandatory or optional Domain : Char(n), Num(n), Num(n,m)
Identifier	n level-1 attributes or roles
Relationship type	Binary, one-to-many or one-to-one
Role	Cardinality : [1-1], [0,1] [0-j], [0-N]
Name	Host language compliant

Table 2.1. Conceptual model constructs

ET's have any number of attributes which can be atomic or compound, mandatory or optional (cardinality [1-1] or [0-1]). There are no multivalued attributes in the CM. Identifiers are composed of one or more attributes or roles. Relationship types are binary one-to-many ([1-N], [0-N], [1-j] or [0-j][1]) or one-to-one ([1-1] or [0-1]).

Example :



A producer is identified by his name. He has an address composed of the number, street, zip code and city. He may produce several films that are identified by their title. A film is characterized by its release year and its duration. One can give the name of one actor (actress). A film always have a producer.

2.3. XML model

In the data transfer process, a XML document is represented by a conceptual schema. This target schema is not a full ER schema, some constructs are not valid. This XML special schema format is called the XML model. In this section, we first give an introduction to XML and then we define the XML conceptual model.

2.3.1. XML

The eXtensible Markup Language (XML) is a markup language for documents containing structured information. XML is a flexible way to create information formats and is therefore used to share information over the Web in a consistent format.
A XML document may be validated through a Document Type Definition (DTD) which defines the grammar of the document. It may be also validated through XML schemas which are specially adapted for DB schemas.
It is interesting to extract data from the XML document. This can be done through the use of special queries provided by the XML query language (XML QL).

[1] N stands for any number, while j represents a fixed maximum cardinality (except 1).

Definition

XML is a markup language, i.e. a system for marking or tagging a document that indicates its logical structure and gives instructions for its layout on the page for electronic transmission and display. XML is therefore a language for documents containing structured information. The XML specification defines a standard way to add markup to documents [Walsh,98].

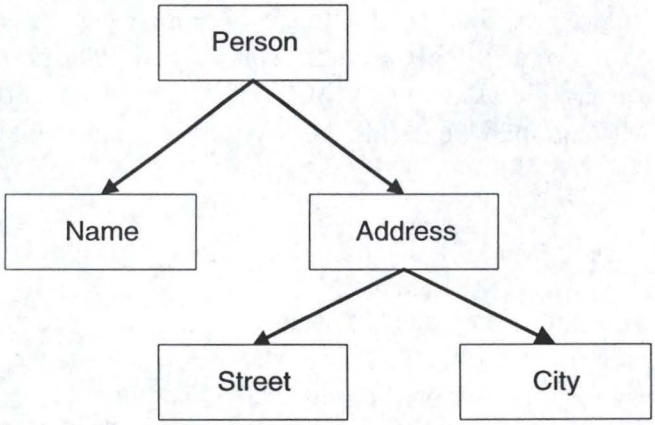
In a XML document, the data, written between the begin and end tags (E.g. : <Person> and </Person>), are tree structured. Indeed, each tag is composed of a series of other tags and if one follows the structure of the document, he will find a tree.

Example :

This XML document contains only one data : a person named Smith that lives in San Francisco :

```
<person>
        <name> Smith</name>
        <address>
                <Street> Powell street </Street>
                <City> San Francisco</City>
        </address>
</person>
```

The format of that data is the following tree structure :



XML is a subset of the Standard Generalized Markup Language (SGML), this means that XML is easier to learn and to use than the complete SGML.
XML differs from the Hyper Text Markup Language (HTML). HTML has predefined tag set and semantics while XML has not. XML provides a facility to define tags and the structural relationship between them (a node may be linked to another one). It describes the content of the tags in terms of what data is being described. E.g : between the tags <movie> and </movie> one can imagine that there is a movie. The consequence of this

characteristic is that a XML document contains the data and their structure. Furthermore, XML tags can be nested at any level of complexity in order to represent complex document.
We use XML for structured documents because HTML, with its predefined tags and semantics, doesn't allow arbitrary structures, and SGML is too complex.

A XML document consists of a sequence of elements. The boundaries of these elements are delimited by a start tag and an end tag. Each element has a type, is identified by a name and may have a set of attributes. Each attribute has a name and a value.

Example :

```
<film id="1">
        <title>Trois couleurs: Bleu</title>
        <actor>Juliette Binoche</actor>
</film>
```

This is a very simple XML document with three elements : film, title and actor. The film element has one attribute : id that identifies it and is composed of the two entities title and actor which are free text and have no attribute.

There is two categories of document : well formed and valid. A document is well formed if the element tags match and nest properly and if the attribute appear only once in the start tag. See [Walsh,98] for a complete list of the conditions.

Example :

```
<film>
        <title>Trois couleurs : Bleu
        <actor>Juliette Binoche</title>
</film id ="1">
```

This document is not well formed while the previous example is a well formed XML document

A document is valid only if it is well formed and if it contains a proper Document Type Definition (DTD) and respects the constraints of that DTD, E.g., element matching and nesting is valid, required attributes are provided and their values are of the correct type,... An example of valid XML is given in page 21.

DTD

A Document Type Definition (DTD) is the grammar of the XML document, it specifies which elements it could contain, with which attributes, and how they could nest. In the DB environment, we can say that the DTD is the schema while the XML document contains the data.

A DTD is composed of elements, attributes, notations and entities. Here is a brief definition of the grammar for the elements and the attributes as one can find in [Bourret].
An element is defined, in the ELEMENT statement, as a group of one or more subelements/subgroups (non-terminal elements), character data (PCDATA), EMPTY, or ANY (terminal elements). Groups can be either a sequence or a choice of subgroups and/or subelements. On groups, subgroups and subelements, we can apply the optional (?), one-or-more (+) or zero-ore-more (*) operators. (Table 2.2.)

Element's declaration	Description
<!ELEMENT cinema (name, address)>	The element film is defined as a sequence of one subelement name and one subgroup address
<!ELEMENT name (#PCDATA)>	The element name is composed of character data
<!ELEMENT address (street, city, phonenbr?)>	The element address is composed of one street, one city and can contain one phone number

Table 2.2. : Element declaration

An element can have one or more attributes which are defined into the ATTLIST statement. An attribute can be optional, required or fixed, and if it is optional or fixed, it may have a default value (Table 2.3.). It also has a type. Table 2.4. contains a non exhaustive type list.

Attribute's declaration	Description
<!ATTLIST film year CDATA #REQUIRED duration CDATA #IMPLIED>	The element film has two attributes : one is required (year) and one is optional (duration). Both are character data and have no default value.
<!ATTLIST person language CDATA "Norsk"	The element person has an attribute language which is optional and have the default value "Norsk"

Table 2.3. : Attribute declaration

Attribute's type	Description
CDATA	Character data
ID	An attribute with this type identifies the element among the other elements.
IDREF	The attribute links its element to another element. Its value is therefore the value of the ID attribute of the pointed element.
IDREFS	The attribute points to many elements

Table 2.4. : Attribute type

Example :

Here is a valid XML document given the DTD which is inside the document. The DTD is written a the beginning of the document.

```
<?XML version="1.0."?>
<!DOCTYPE cine [
<!ELEMENT (cinema,movie)>
<!ELEMENT cinema (name,location)>
<!ATTLIST cinema presents IDREFS #REQUIRED>
<!ELEMENT location (street, city)>
<!ELEMENT movie (title, producer, actor*)>
<!ATTLIST movie id ID #REQUIRED>
<!ELEMENT name #PCDATA)>
<!ELEMENT street #PCDATA)>
<!ELEMENT city #PCDATA)>
<!ELEMENT title #PCDATA)>
<!ELEMENT producer #PCDATA)>
<!ELEMENT actor #PCDATA)>
]>

<cine>
<cinema presents = "1", "2">
        <name>Cameo</name>
        <location>
                <street>rue des carmes</street>
                <city>Namur</city>
        </location>
</cinema>
<movie id="1">
        <title>Trois couleurs : Bleu </title>
        <producer>Karmitz</producer>
        <actor>Juliette Binoche</actor>
</movie>
<movie id="2">
        <title>Les schtroumpfs</title>
        <actor> Le Grand Schtroumpf</actor>
        <actor>Le Schtroumpf à lunettes</actor>
</movie>
</cine>
```

Cine is composed of a cinema that presents two movies. The cinema is the *Cameo* and is located in *Namur*. The movies are *Trois couleurs : Bleu* and *Les schtroumpfs*. Each of them is identified by a number (respectively, 1 and 2). This id is referenced by the idrefs attribute presents of the cinema element.

XML schema

"XML schemas are an attempt to replace DTD's with something "better""[Walsh,99].

Indeed, the main default of DTD's in the DB domain is that they have extremely limited datatyping. They can only express the datatype for the attributes (CDATA, ID, IDREF). For the elements, there is only one datatype : PCDATA, which means all possible characters.

A XML schema defines a model for documents in terms of constraints. There are two kinds of constraints :

- **content model** constraints which describe the order and sequence of the element tags
- **datatype** constraints which describe valid units of data.

A XML document is valid if it is well formed and if a schema is associated to it and the document does not violate any of the constraints of that schema. Therefore, there is two validity, one for the content model and one for the data type. The content model validity tests whether the order and nesting of tags is correct, while the datatype validity tests whether the information are of the correct type and fall within the specified legal values. Indeed, the datatype is characterized by the value space and the lexical space. The value space is the set of permitted value for the datatype while the lexical space consists of a set of valid literals. Each value in the datatype value space maps to one or more valid literals in its lexical space.
The ability to express datatype validity is the real advantage of the schemas compared to DTD's.

The XML schemas are written in an XML syntax, unlike the DTD's. They are XML documents where elements and attributes are used to express the semantics of the schema.

Example :

```
<elementType name="Title">
<mixed/>
</elementType>
```

The elementType element is used to declare an element. The name of the element is given in the name attribute. The tag <mixed/>[2] means that the Title element can contain a mixture of character data and elements.

To define a datatype, we use the datatype element.

Example :

```
<datatype name="TelephoneNbr">
        <basetype name="string"/>
```

[2] <mixed/> is an empty element. This notation is equivalent to <mixed></mixed>

```
        <lexicalRepresentation>
                <lexical>999/99-99-99</lexical>
                <lexical>999/99/99/99</lexical>
        /lexicalRepresentation>
</datatype>
```

The *telephonenbr* type is defined here as a string that has two possible formats. The lexical space is defined in the LexicalRepresentation element. E.g. the telephone number 081/23-13-48 has the correct format.

The complete definition of a XML schema is given in [W3C,99a] and [W3C,99b].

XML schemas seems a great improvement of the DTD's. But, today, the DTD's have the virtue of being well understood and of offering a good way to describe the document structure. Unfortunately, it will take some time before XML schemas are well understood and used, that's why we are going to use a DTD to define the DB schemas we are working with in the transfer process.

XML QL

There is a special query language for XML document called XML-QL and which has been defined by [W3C,98]. XML-QL is SQL-like and allows XML document to be queried like a DB, the corresponding DTD being the DB schema. It can extract data and perform some data transformations via operations as join and aggregate. It can also support the data construction that is required by transformations. XML-QL syntax can be found in [W3C,98].
An introduction to the language through examples is given in appendix B.

2.3.2. XML model constructs

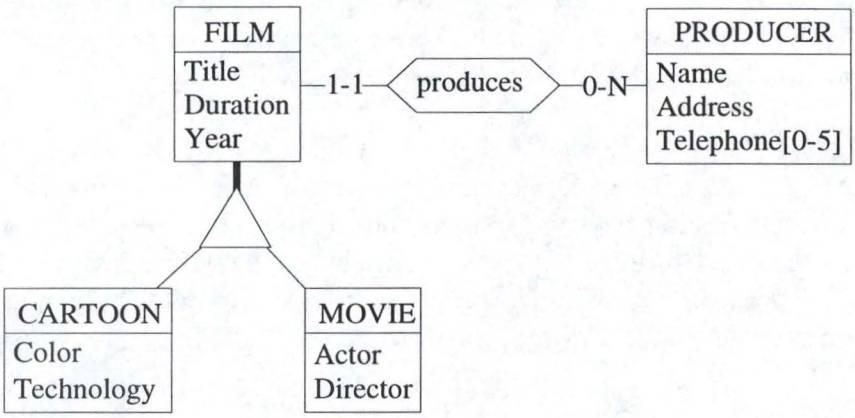
The target XML conceptual schema represents the XML document using ER objects. Not all constructs are valid, just like in the CM. All CM constructs are accepted but there are less constraints on the ET, RT and attributes. Table 2.5. shows the constructs and their constraints.

Constructs	Constraints
Entity type	Any number of attributes and identifiers
Attribute	Atomic or compound, mandatory or optional, single valued or multivalued. Domain : Char(n), Num(n), Num(n,m)
Identifier	n level-1 attributes or roles
Relationship type	Binary, one-to-many or one-to-one
Role	Cardinality : [1-1], [0,1] [0-j], [0-N]
IS-A	no constraint

Table 2.5. : XML model constructs

The difference with the CM is that multivalued attributes and isa-constructs are authorized.

Example :



A film is characterized by its title, duration and the release year. It has one producer which can produce several films. A cartoon and a movie are a film, that means that they have all the film attributes. The cartoon has particular attributes : technology and color which is used to express whether it is colored or not. A Movie is characterized by an actor and a director.

2.4. Conceptual Query Language

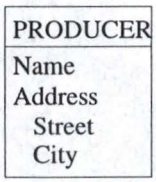
Data in a DB can be extracted, altered or deleted using queries. The Structured Query Language (SQL) is a language that offers such queries. SQL queries are based on the physical schema. Because, in the data transfer process, we work with conceptual schemas, we would like to have queries based on them. Such queries are provided by the Conceptual Query Language (CQL) defined by [Thiran,1999a].

CQL is based on SQL and specially adapted to support the CM. If we want to execute queries on a conceptual schema, we have therefore to work on schemas that respect the CM. CQL includes selection and update queries.

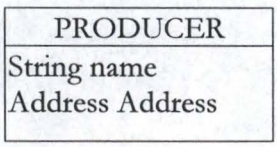
CQL results objects are called entity objects. Their properties correspond to the attributes of the entity types. E.g. to the ET producer, correspond the entity objects producer and address. The last is an implicit entity object.

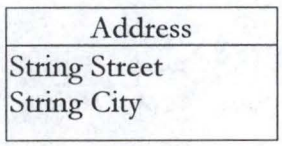
Example :

ET :



Entity type objects:





CQL **selection statements** uses the *select-from-where* clause, their format is given is table 2.6. We can access attributes of compound attributes via the name of the compound attributes.

Example :

Select P.name
From Producer P
Where P.address.city ="Namur"

This query selects the name of all the producers that live in Namur.

CQL also offers the possibility to follows links in the conceptual schema by giving the relationship name in the *where clause*. The order of entity names in the *where clause* is irrelevant.

Example :

Select M.title
from Movie M, producer P
where P produces F

This query gives the title of all the movies that have a producer.

CQL updates queries allow to make some data modifications. The **delete, update** and **insert statements** are defined on ETs while the **link** and **unlink statements** are defined on the two ETs that are connected by the RT. Table 2.6. gives the format of these statements.

Statement	Format
Select	*select* select-clause *from* ent-list *where* where-clause
Delete	*delete object* object-name *from* ent-name
Update	*update object* object-name1 *into* object-name2 *from ent-name*
Insert	*insert object* object-name1 *into* ent-name [*linked with* object-name2 *via* rel-name {*and* object-name-i *via* rel-name-i}]
Link	*link* object1 *to* object2 *via* rel-name
Unlink	*unlink* object1 *to* object2 *via* rel-name

Table 2.6. : statements format

The **delete** statement removes one instance of the ET. This query also deletes all the entity relationships with other entities.

Example :

Delete object obj_producer from Producer

The object obj_producer is removed from the ET Producer

The **update** statement replaces one instance of an ET by another.

Example :

Update object obj_producer into new_obj_producer from Producer

The object obj_producer is replaced by the object new_obj_producer in the ET Producer.

The **insert** statement is used to add a new entity in an ET. The *linked with* clause in the insert statement is optional. It is used to indicate the entity object(s) that is (are) in relation with the new entity.

Example :

Insert object new_obj_producer into Producer linked with obj_movie via produces

A new producer is inserted into the Producer ET. This instance is linked with a movie entity represented by the obj_movie object.

The **link** statement is used to create a new relation between two entities.

Example :

Link obj_producer to obj_movie via produces

A relation produces is created between the objects obj_producer and obj_movie

The **unlink** statement removes a relationship between entities.

Example :

Unlink obj_producer to obj_movie via produces

The link between the objects obj_producer and obj_movie is removed.

Chapter 3
Architecture

Chapter 3 : Architecture

The converter architecture differs regarding to the converter type : DB-to-DB or DB-to-XML. In this chapter, we define the two converter architectures and then we bring to light the converters support tools.

3.1. DB-to-DB architecture

The DB-to-DB converter architecture, as shown in figure 3.1. is fairly simple. There are two **conceptual schemas**, the second being derived from the first one via symmetrically reversible transformations. Let's call the first the source schema and the latter the target schema. The target schema is a subset of the source schema. A **local server** is attached to each schema, its role being to manage the conceptual/physical conversion of each local **DB's** and to give the availability to directly address the conceptual schemas. The **converter** is built on the basis of the transformations performed on the source schema. It comprises two parts : the extraction module and the insertion module. The aim of the first module is to extract the relevant data from the first DB, using the local server. By relevant, we mean every data supposed to be present in both the source and target DBs. The insertion module is designed (obviously) to insert the extracted data in the corresponding location of the second DB.

Each component is described below.

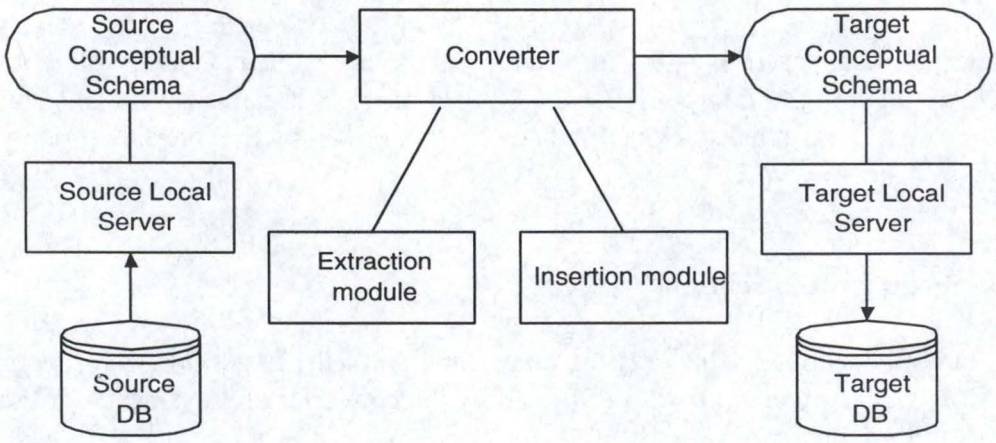


Figure 3.1. : DB-to-DB architecture

Source DB

The source DB is the local DB from which the converter collects the data. A local server is linked to that DB.

Source local server

The source local server is attached to the local source DB. It offers a CQL interface based on the source conceptual schema. That is to say that it is possible to address queries at the DB conceptual-layer via the source local server, the latter managing the conceptual/logical and logical/physical translations. This device is provided by the InterDB support case, as explain in section 3.3.2.

Source conceptual schema

The source conceptual schema is the source DB representation. This schema is easily read and understood by the user. It respects the CM constructs. Some symmetrically reversible transformations are performed on this schema in order to produce the target schema.
The converter CQL selection queries addressed to the local server are based on the source conceptual schema.

Target DB

The target DB is the local DB to which the converter transfers the data picked up in the source DB. It is attached to the target local server

Target local server

The target local server is attached to the local target DB. It is used by the converter to insert the data into the target DB using update queries. It offers a CQL interface based on the target conceptual schema. This device is provided by the InterDB support case, as explain in section 3.3.2.

Target conceptual schema

The target conceptual schema is the representation of the target DB. It respects the CM constructs. The update queries performed by the converter on the target local server are based on this schema.

Converter

The converter acts like a mediator between the two DB's, its aim being to transfer data from one to the other. It is composed of two parts, the extraction and insertion modules. The extraction module is in charge of extracting the data from the source DB, while the insertion module inserts them into the target DB.

The converter generation process is described in chapter 6.

3.2. DB-to-XML architecture

The DB-to-XML converter architecture is shown in figure 3.2. Just like in the DB-to-DB case, there are the **source** and the **target conceptual schemas**. The first represents the **source DB** and the second one, the **XML document**. The target schema is derived from the source schema via symmetrically reversible transformations. A local server is attached to the source DB. The **converter**, between the source and the target conceptual schemas, is composed of two modules : the extraction and the write modules. The extraction module is used to extract the data from the source DB via the **local server**. The write module creates the XML document based on the target schema. The XML document contains all the data in a textual format and is validated by its DTD.

The source DB, the source local server and the source conceptual schemas have all the same definition and use than those already defined in the DB-to-DB architecture. The other components are described below.

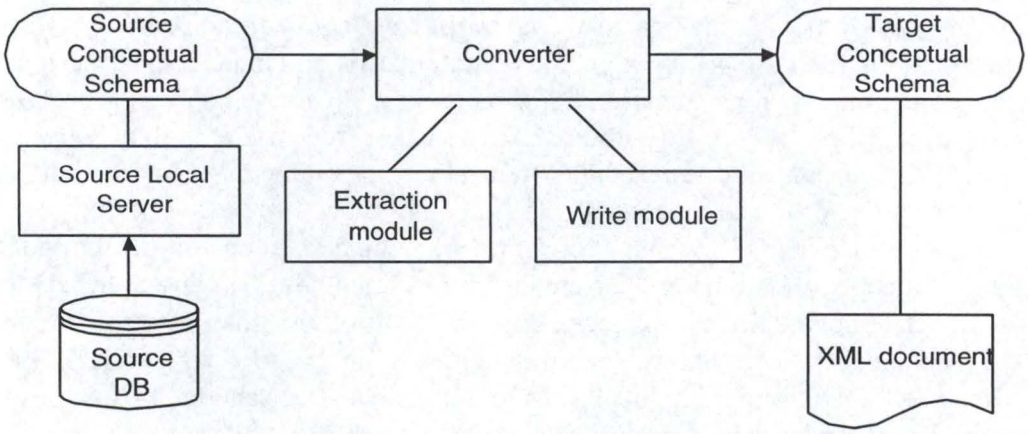


Figure 3.2. : DB-to-XML architecture

XML Document

The XML document is used to store the data in a tree-structure format given by the DTD. This document is produced by the converter.

Target conceptual schema

The target conceptual schema is the representation of the XML document. It is derived from the source conceptual schema. The DTD associated to the XML document is based on that schema. The strategy to translate the target schema into a DTD is explained in chapter 7.
The target schema respects the XML model structures defined in chapter 3.

Converter

In this architecture, the converter resides between the source and target conceptual schemas. Its two modules, the extraction and the write modules, transfer the data from the local DB to the XML document. The extraction module, just as its DB-to-DB counterpart, retrieves the data from the source DB via the source local server. The write module, is designed to create the XML document and to fill in it with the extracted data.

The process of the DB-to-XML converter generation is described in section 7.2.

3.3 Support Case

3.3.1 DB-Main case tool

« The DB-Main CASE environment is a complete set of tools dedicated to database applications engineering. This graphical, repository-based, software engineering environment is dedicated to *database applications engineering*. Besides standard functions such as specification entry, examination and management, it includes advanced processors such as transformation toolboxes, reverse engineering processors and schema analyses tools. In particular, DB-Main offers a rich set of *semantics-preserving transformational operators*[1] that allow developers to carry out in a systematic way the physical/conceptual mapping. Another interesting feature of DB-Main is the *Meta-Case layer*, which allows method engineers to customize the tool and to add new concepts, functions, models and even new methods. In particular, DB-Main offers a complete development language, Voyager 2[2], through which new functions and processors can be developed and seamlessly integrated into the tool. » [Thiran, 1999]

We use the DB-Main graphical tools to draw the source schema, then we use the semantics-preserving transformational operators to transform the source schema and translate the target. DB-Main also offers the (more than useful) possibility to register all the transformations made on the source schema.

[1] The set of transformations is defined in chapter 4.
[2] The Voyager2 programming language is defined in chapter 4.

3.3.2 InterDB

InterDB is dedicated to the integration and the interoperability of heterogeneous and distributed information systems. A part of the InterDB architecture is shown in figure 3.3. and described below.

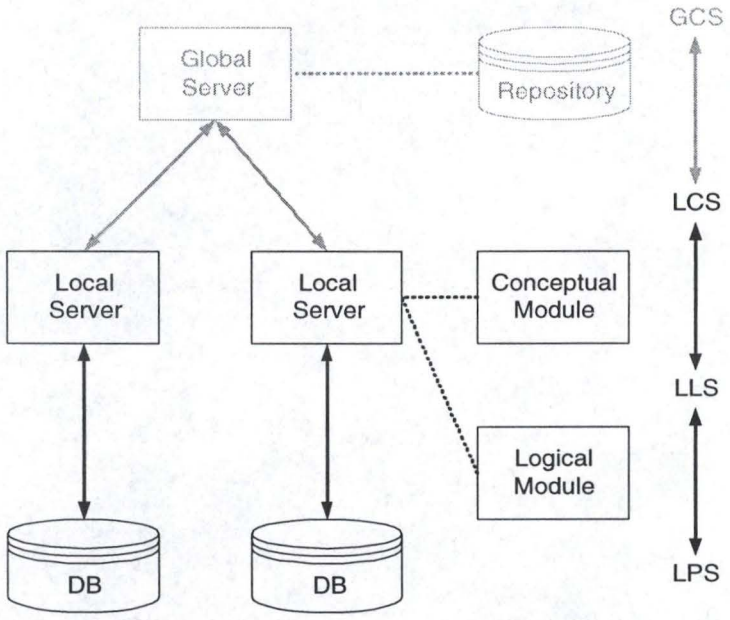


Figure 3.3.: InterDB architecture

In the InterDB architecture, there are local servers attached to each DB. Each of these mediators is based on the DB Local Conceptual Schema (LCS). A local server comprises the logical and the conceptual modules. The *logical module* hides the physical constructs of the DB and dynamically transforms queries and data from the logical model to the physical one (Local Physical Schema). All logical modules offer a common interface. The *conceptual module* provides a conceptual view of the Local Logical Schema (LLS) and hides all the technical and optimization constructs. Therefore, each local server appears as a conceptual BD with a unique interface for application programs.

The global server[3] offers a conceptual interface based on the GCS (Global Conceptual Schema). It manages the global queries (queries addressed to the DB, independently of their distributed locations).

The generator uses the local server provided by InterDB. Indeed, the generated converter is linked to two or one local server, the DB-to-DB converter being attached to the source and target local servers whereas the DB-to-XML converter being only linked to the source local server. The local server allows the converter to operate, i.e. to address queries, on conceptual schemas because it performs the conceptual/logical and logical/physical translation of the queries.

[3] We don't use the InterDB global server, this is why the description is just a brief one

Chapter 4
Methodology

Chapter 4 : Methodology

In this chapter, we present the converter generation methodology which is quite different according to the converter type we have to generate : DB-to-DB or DB-to-XML. Then, we expose the set of theoretical transformation from which the mappings are derived. Finally we show up some useful tools like the DB-Main repository.

4.1. The DB-to-DB converter generation methodology

The DB-to-DB converter generation methodology can be decomposed in two steps, as shown in figure 4.1 :

- mapping analyse
- DB-to-DB converter generation

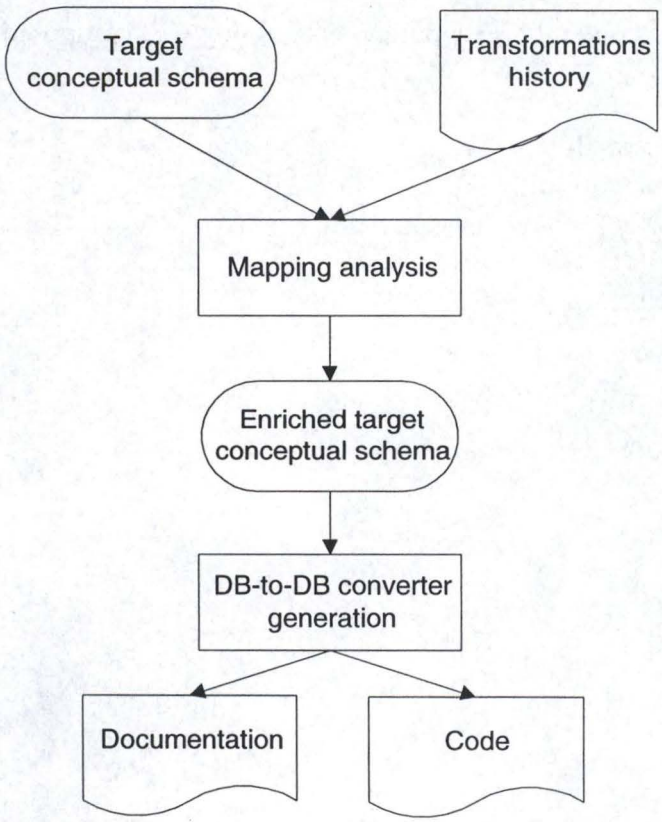


Figure 4.1. : DB-to-DB converter generation methodology

4.1.1. Mapping analysis

The target conceptual schema representing the conceptual design of the target DB can be found in the DB-Main repository. The latter also offers the transformation historic (stored in a log file). This file contains the historic of all the transformations performed on the source conceptual schema in order to produce the target conceptual schema. All these transformations are described in section 4.3. By analysing the information given by the repository and the log file, the generator is able to discover the mappings and to produce an enriched target conceptual schema. The enriched target schema is the same than the target schema except that the mappings are inserted, in a defined format, into the repository. The correspondences between the source and target conceptual schemas can now be found by the generator into the target schema.

4.1.2. DB-to-DB converter generation

The enriched target conceptual schema contains the mappings between the source and the target schemas. Each mapping is analysed by the generator in order to create the DB-to-DB converter.
The converter source code is therefore produced and, in the same time, some useful documentation linked to the code is also generated.

4.2. DB-to-XML converter generation methodology

The DB-to-XML converter generation methodology is composed of three steps, as shown in figure 4.2. :

- mapping analyse
- DTD generation
- DB-to-XML converter generation

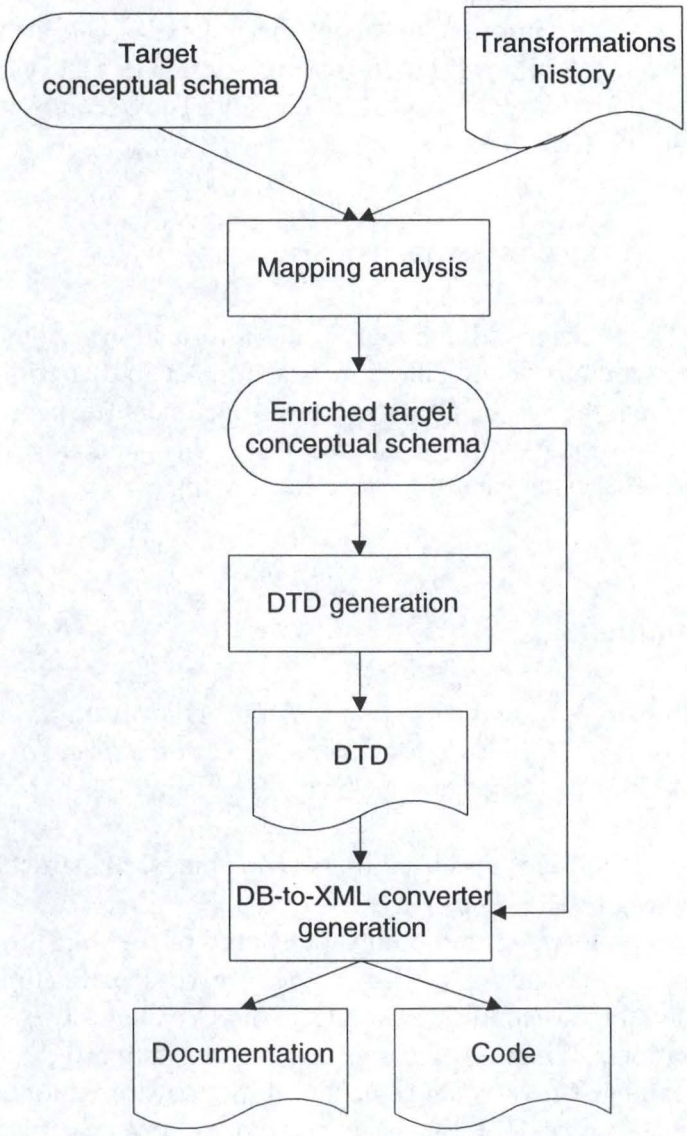


Figure 4.2. : DB-to-XML converter generation methodology

4.2.1. Mapping analyse

This process is similar to its DB-to-DB counterpart, the reader can refer to the explanation in section 4.1.1.

4.2.2. DTD generation

As we are working here not with a target DB, but with a XML document, we have to define the format in which we want to insert the data in the output file. The format of a XML document is given by the DTD that can be internal or external to the document. In our case, we produce an extern DTD on the basis of the enriched target conceptual schema. The generated DTD is the translation of a conceptual view into a tree-

structured view. All the information about the mappings can also be inserted in the DTD. It is important to build a DTD for two reasons. The first one is that the DTD is used by the generator to build the XML document. The second use of the DTD is to validate the produced XML file.

4.2.3. DB-to-XML Converter generation

Now that the DTD is generated and that all the information needed lie in the enriched target conceptual schema, we are able to produce the code that will effectively perform the data transfer from the source DB to the XML document (respecting the format given by the DTD). Alongside with the source code, the generator also produces some documentation that explains what the converter is doing.

4.3. Transformations

"A schema transformation is an operator T that replaces a source construct C in schema S with construct C', leading to schema S'. C' is the target of source construct C through T : C'=(C). [Hainaut,95]

A transformation Σ is specified by two mappings, the structural mapping T and the instance mapping t : $\Sigma = <T,t>$.
The structural mapping, which represents the syntax of the transformation, is such that C'=T(C). In order to define T, we use a predicative specification, i.e. we give the precondition P and the postcondition Q on C. P and Q includes two kinds of statements : structural declaration and naming and applicability constraints.
The instance mapping, which is the semantic of the transformation is such that c' = t(c) where c is an instance of C. t can be expressed by any data manipulation language expressions.
The complete specification of Σ is therefore : $\Sigma =<P,Q,t>$.

There are three classes of transformations : non reversible, simply reversible and symmetrically reversible transformations.
Non reversible transformations are transformations that have no inverse. It is impossible to recover the source C from target C'. These transformations are not semantic preserving.
A transformation $\Sigma 1 = <T1,t1> = <P1,Q1,t1>$ is **simply reversible** iff, $\exists\ \Sigma 2 = <T2,t2> = <P2,Q2,t2>$, an inverse transformation, such that $\forall$ c of C : $P1(C) \Rightarrow T2(T1(C)) = C$ and $t2(t1(c)) = c$. $\Sigma 2$ is the inverse of $\Sigma 1$, but not conversely.
A transformation $\Sigma 1 = <T1,t1> = <P1,Q1,t1>$ is **symmetrically reversible** iff, $\Sigma 1$ is reversible and its inverse $\Sigma 2$ is reversible. In other words :
$\forall$ c of C : $P1(C) \Rightarrow [T2(T1(C)) = C \text{ and } t2(t1(c)) = c]$
$P2(C') \Rightarrow [T1(T2(C')) = C' \text{ and } t1(t2(c')) = c']$
Since $\Sigma 2 = <Q1,P1,t2>$, the concise notation for $\Sigma 1 + \Sigma 2$ is $\Sigma = <P,Q,t1,t2>$

The symmetrically reversible transformations on DB conceptual schemas are shown in table 4.1.

Transformation	Description
ET-to-RT	An entity type is transformed into a relationship type
ET-to-ATT	An entity type is transformed into an attribute
Split-merge	Components of an entity type are extracted into a new entity type (split) or can be added to an existing entity type (merge)
Make-subtype	A supertype is given to two entity types
RT-to-ET	A relationship type is transformed into an entity type
RT-to-ISA	The common entity type of two relationship types is transformed into a supertype of the other entity types
Att-to-ET/value	An attribute is transformed into an independent entity type. Each new entity represents a distinct value of the attribute
Att-to-ET/instance	An attribute is transformed into an independent entity type. Each new entity represents an instance of the attribute.
MultAtt-to-SingleAtt	A multivalued attribute is transformed into an atomic attribute which is the concatenation of its values
MultAtt-to-SerialAtt	A multivalued attribute is transformed into a list of atomic attributes
Disaggregate	A compound attribute is replaced by its components
Aggregate	A list of attributes is replaced by a compound attribute
rename	A TE, RT or attribute name is changed.

Table 4.1. : Symmetrically reversible transformations

Rem : The MultAtt-to-SingleAtt and MultAtt-to-SerialAtt are not fully reversible transformations, they are reversible from left to right.

4.4. Support Tools

The generator uses some tools in order to be supported during its generation process. The main tool is DB-main with its repository and history file. DB-Main offers also a language : Voyager2.

4.4.1. DB-Main repository

The repository contains the project current methodology and history. It is also used by the DB-main tool to record the product specifications, mainly schemas and texts.

The repository is composed of **objects**. All the instances of an object type must respect the object definition. The aim of the repository is to store the definition of any ER schema. It is thus normal that all the concepts like *entity type, relationship type, attribut* have a corresponding object type. E.g., the entity_type object type corresponds to the *entity type* concept.

Two kinds of relation may exist between object types : **is_a** and **link relations**. The is_a relation expresses the generalization concept. When an object A derives from another object B, we say that A is a specialization of B or that B is the generalization of A. All the properties of the object A also hold for the object B.

Example :

co_attribute —o—→attribute

The co_attribute is-a attribute. The co_attribute and attribute object types correspond to the concepts of *compound attribute* and *attribute*.

The link relation represents a one-to-many relation between two objects. If there is such a link, to each instance of an object A corresponds a collection of instances of the object B. Each side of the relation r plays a role r or @r.

Example :

owner_of_att —owner_att— attribute

the object owner_of_att plays the role owner_att with a cardinality 0-∞ while the object attribute plays the role @owner_att with cardinality 1-1. The object type owner_of_att is a generalization of object of ent_rel_type (object that groups the entity_type and rel_type object)and co_attribute types.

There is a special object in the repository : the meta_property. The meta-property is used to dynamically extend the repository by adding new fields to one object-type in the repository. Adding one new instance in the meta_property entity type and linking it with one instance of a meta_object enables to add one property to the entity type described by the meta-object. Once this modification is done, the new property is available in the DB-Main Case tool and in Voyager 2.
The complete repository's schema is given is appendix C.

4.4.2. History manager

Another specificity in the repository is the history manager. It records all the activities carried out by the user. Though the historic is hidden inside the repository, a log file is created when the analyst requests it either to examine its content or to replay some actions. We use this file to recover all the transformations performed on the source schema in order to build the target schema.

4.4.3. Voyager2

Voyager 2 is a complete, fourth-generation, semi-structural language. It is similar to traditional languages like C and Pascal. Voyager 2 offers predicative access to the repository, the analysis and the generation of external texts and the definition of recursive functions and procedures. The very difference between Voyager 2 and the other languages is the ability to make queries on the predefined repository. New object

types and properties can be dynamically added and managed through Voyager 2 procedures.

Example :

```
attribute: att;
entity_type:ent;
begin
for att in ATTRIBUTE[att]{@OWNER_ATT : [ent]} do
        {
        print(att.name);
        }
end
```

This program uses a predicative query that gives all the attributes of a given ET, and prints their names on the screen.

The lexical analyzers offered by Voyager 2 are very useful in order to parse and analyze an input file like the log file.
Once a program is written and compiled with the Voyager 2 compiler, it can be executed in the DB-Main Case tool.

Chapter 5
Mapping Analysis

Chapter 5 : Mapping analysis

The first step in the converter generation methodology is the analysis of the mappings between the source and target schemas. Once extracted from the log file, these mappings are inserted into the target conceptual schema. Thus we have an enriched schema.
Given that the source conceptual schema is attached to a local server, it has to comply with the Conceptual Model (CM). In the DB-to-DB transfer process, the target schema is also attached to a local server, it has thus to respect the CM structures. The XML schema has its own structures as defined in chapter 2.

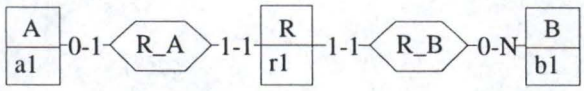
The first section of this chapter describes the symmetrically reversible transformations and defines the mapping format for each of them that can be applied on the source schema. Then, we explain how to insert the mappings in the target conceptual schema. Finally, we give a brief explanation of the AnalyseMap program.

5.1. Mapping representation

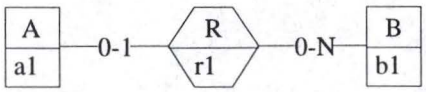
In table 4.1., one can find the list of the main symmetrically reversible transformations that can be applied on a conceptual schema. Transformation on multivalued attributes are not analyzed here because these attributes are no valid constructs in the CM. Some other of these transformations must be more constrained in order to be performed on the source conceptual schema, this is due to the very structure of that schema. These transformations are defined and illustrated below. A comment on their potential use on the source schema is given too.
For each transformation, we define the mapping format and give an example. A meta property named "*corresp*" is created for each ET, RT and attributes of the target schema. It contains the mapping, i.e. the correspondence with the source schema.

5.1.1. ET-to-RT

With ET-to-RT, an ET is transformed into a RT. This transformation can be applied only under some conditions. Indeed, the ET must have an identifier and be linked to at least two other ET with one-to-many RT.



⇔



This is the RT-to-ET inverse transformation.

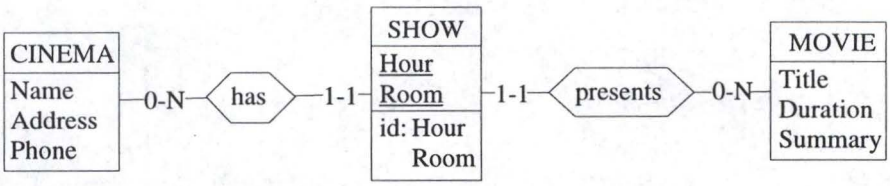
The ET-to-RT transformation is used on the source conceptual schema with an ET linked to only two other ETs and with no attributes. Indeed, a n-ary RT with attributes is not a valid construct of the CM.

Although it is rare to find a schema that contains an ET without attributes, the mapping format for each RT, and attributes obtained by ET-to-RT is :

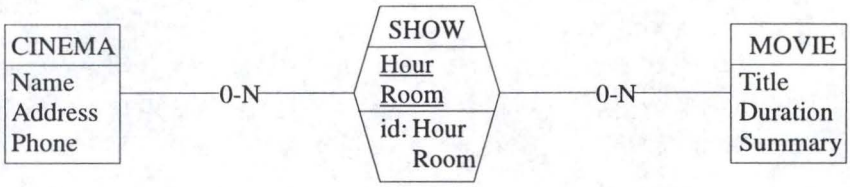
- For each RT : ET-to-RT (*ent*) where *ent* is the transformed TE
- For each attribute : ET-to-RT (*ent,att*) where *att* is the corresponding *ent* attribute

Example :

Source schema :



Target schema :

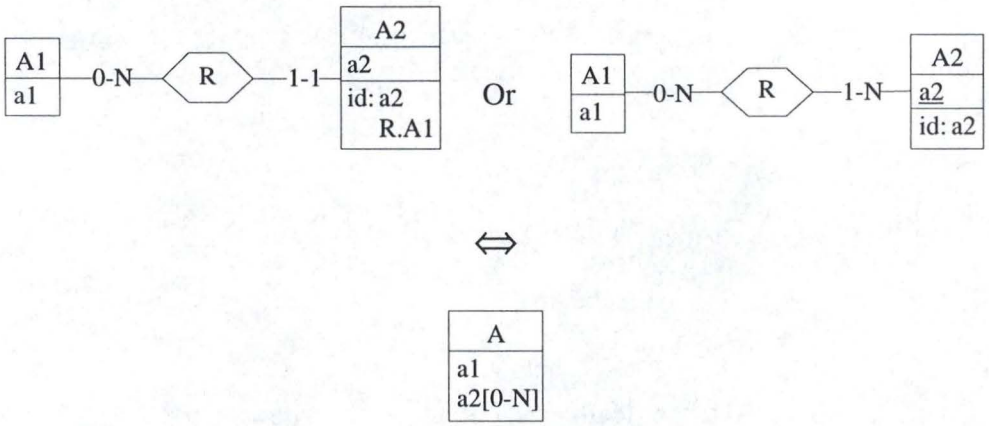


Mapping :

Show : corresp =ET-to-RT(show)
Hour : corresp = ET-to-RT(show,hour)
Room : correspt= ET-to-RT(show,room)

5.1.2. ET-to-Att

ET-to-Att transforms an ET into an attribute. Some conditions are required for this transformation : the ET must be linked to only one other ET, have only one identifier and all its attributes must belong to that identifier.



This is the Att-to-ET/value or Att-to-ET/instance inverse transformation.

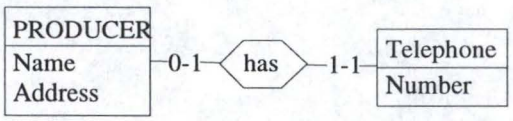
Given that the RT are binary, the ET-to-Att always transforms the ET into an ET attribute and not into a RT attribute, which respects the CM. However, we should give another constraint on that transformation : RT must be one-to-one because multivalued attributes are no valid constructs. This additional condition is obviously not necessary in the XML transfer process.

The mapping format for each attribute obtained by ET-to-Att is:

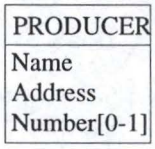
- ET-to-Att (*ent,att,rel*) where *ent* is the former TE, *att* its attribute and *rel* the RT that linked *ent* with the ET.

Example :

Source schema :



Target schema :



Mapping :

Number : corresp = ET-to-Att (telephone,number,has)

5.1.3. Split-merge

The split-merge transformation is used to extract (split) some components from an ET and to insert (merge) them next into another ET.

This transformation is symmetrically reversible and its inverse is itself.

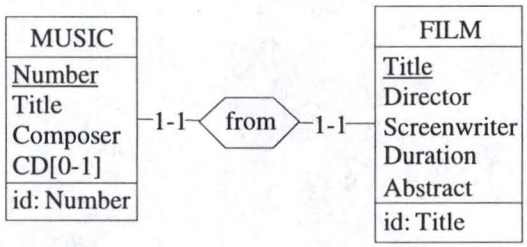
Split-merge can be applied on the source schema in order to create the DB or the XML conceptual schema.

The mapping format for each attribute transferred by the split-merge transformation is:

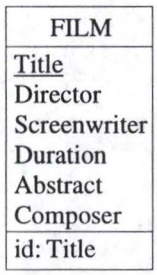
- split-merge (*ent,att,rel*) where *ent* is the name of the ET from which the attribute *att* is extracted. *rel* is the name of the RT that links *ent* to the ET where *att* is inserted.

Example :

Source schema :



Target schema :

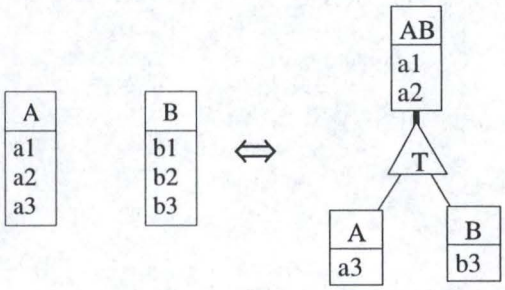


Mapping :

composer : corresp = split-merge (music,composer,from)

5.1.4. Make-Subtype

A super type is given to some ET using the Make-subtype transformation. The common attributes are moved into the super type. There are three special is-a relationships : partition, total, disjoint. If it is total, each supertype is at least one of the subtype. If it is disjoint, the supertype can be only one of the subtype. The partition is total and disjoint, i.e. each supertype has one and only one subtype. With the make-subtype transformation, we create a total relationship.



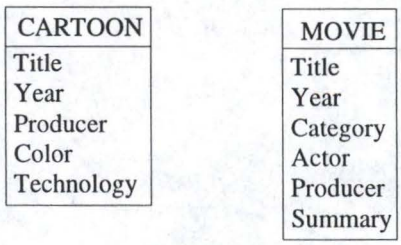
As the is-a construct is not allowed in the CM, it is impossible to realise this transformation. However, in the XML case, there is not this restriction and we can imagine that a supertype is created.

The mapping is inserted in the meta-properties of the supertype and its attributes. The mapping is not necessary for each subtype attributes because they are not changed The mapping formats are :

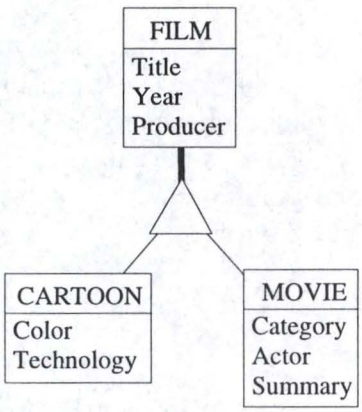
- For each supertype : Make-subtype (*ent1, ent2*) where *ent1* and *ent2* are the former ET.
- For each supertype attribute : Make-subtype (*ent1.att, ent2.att*) where *att* is the attribute. In the transfer process, it is not necessary to have both the attributes given that they are semantically the same. However, if we want to rebuild the source schema with the target schema, this information will be very useful. For example, one can imagine that the attributes in the two ET have different names and without this mapping, it would be impossible to recover them.

Example :

Source schema :



Target schema :

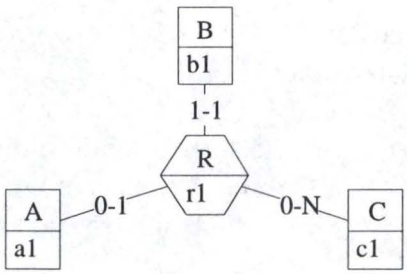


Mapping :

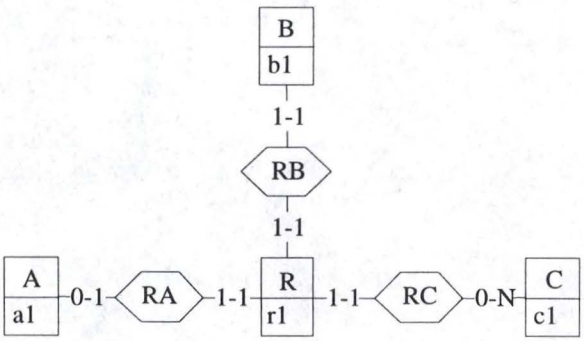
Film : corresp = Make-subtype (cartoon, movie)
Title : corresp = Make-subtype (cartoon.title, movie.title)
Year : corresp = Make-subtype (cartoon.year, movie.year)
Producer : corresp = Make-subtype (cartoon.producer, movie.producer)

5.1.5. RT-to-ET

RT-to-ET transforms a RT into an ET. There is no constraint on that transformation.



⇔



It's the inverse of the ET-to-RT transformation.

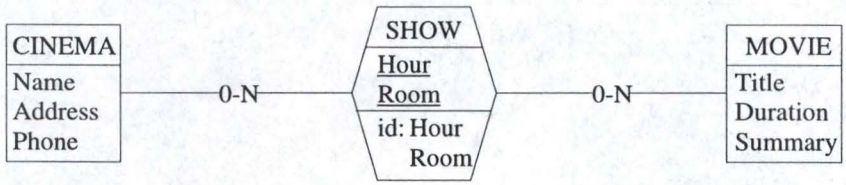
This transformation must be applied on a binary RT with no attribute and gives an ET with no attribute. Although such an ET is not very used in conceptual schemas, it is not rare to find a XML document where an element has no subgroup.

The mapping format for each ET and its attributes and each RT obtained by the ET-to-RT transformation is :

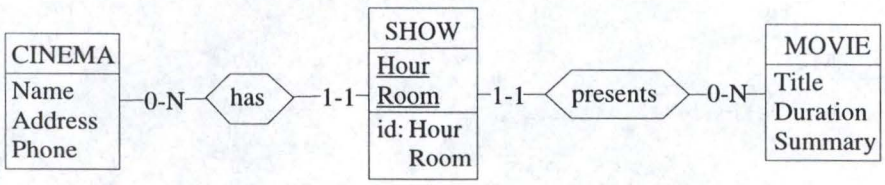
- For each ET : RT-to-ET (*rel*) where *rel* is the transformed RT.
- For each ET attribute : RT-to-ET (*rel,att*) where *att* is the corresponding *rel* attribute.
- For each new RT : RT-to-ET (*rel,ent*) where *ent* is the ET linked by rel.

Example :

Source schema :



Target schema :

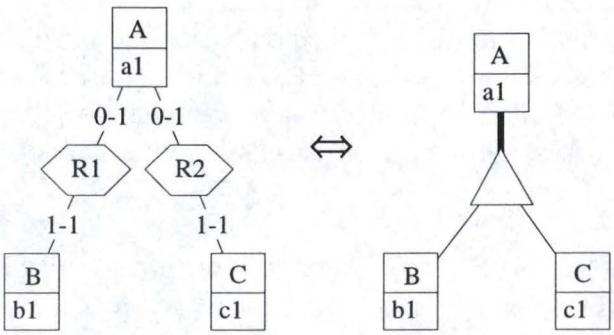


Mapping :

Show : corresp = RT-to-ET (show)
Hour : corresp = RT-to-ET (show, hour)
Room : corresp = RT-to-ET (show, room)
has : corresp = RT-to-ET (show, cinema)
presents : corresp = RT-to-ET (show, movie)

5.1.6. RT-to-ISA

RT-to-ISA transforms the common ET of a set of RT into a super type of the other ET under constraint of one-to-one RT.



Here again, this transformation would be accepted only in the XML conversion case because is-a constructs are only authorised there.

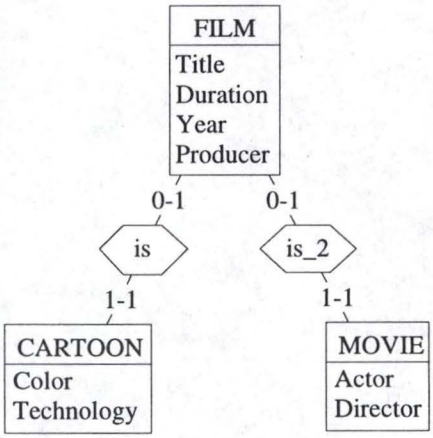
The RT-to-ISA transformation is used on the source schema transformed in order to have the XML target schema.

The mappings are, for each subtype :

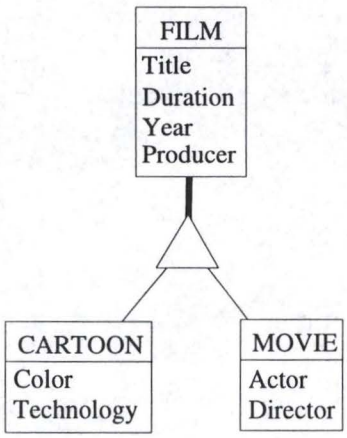
- RT-to-ISA (*ent, rel*) where *ent* is the name of the supertype and *rel* the RT that used to link the ET with *ent*.

Example :

Source schema :



Target schema :

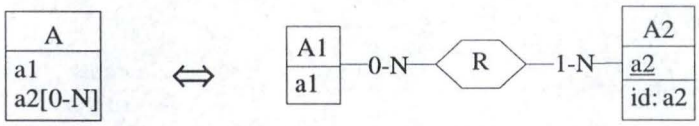


Mapping :

Cartoon : corresp = RT-to-ISA (film,is)
Movie : corresp = RT-to-ISA (film,is_2)

5.1.7. Att-to-ET/value

The Att-to-Et transformation is used to change an attribute into an ET. Two representations are possible for these attributes, one is by value and the other is by instance. With the Att-to-ET/value transformation, each new entity represents a distinct value of the attribute.

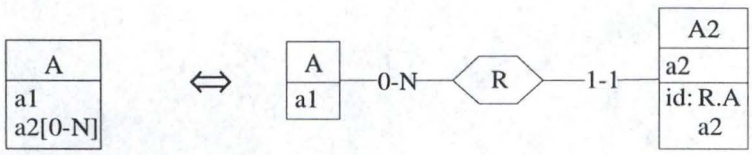


This is the ET-to-Att inverse transformation.

The Att-to-ET/value transformation always create a 1-N cardinality for a role. As this construct is not valid in the CM, this transformation is not allowed.

5.1.8. Att-to-ET/instance

With Att-to-ET/instance, an attribute is transformed into an ET and each new entity represents an instance of the attribute.



This is the ET-to-Att inverse transformation.

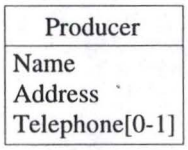
As multivalued attribute is not a valid construct in the CM, this transformation will only be used with optional or mandatory single valued attributes. Let's note that if the transformation is performed on a mandatory attribute, it creates a one-to-one RT with both minimum cardinalities equals to 1 which implies an unbearable constraint for the data insertion. This problem is exposed in more thoroughly in chapter 6.

The mapping format for each new RT, ET and its attributes is :

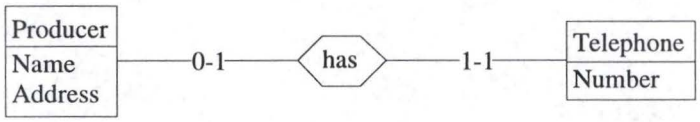
- For each ET : Att-to-ET/instance (*ent,att*) where *ent* is the source ET and *att* its transformed attribute.
- For each attribute : Att-to-ET/instance (*ent,att*)

Example :

Source schema :



Target schema :

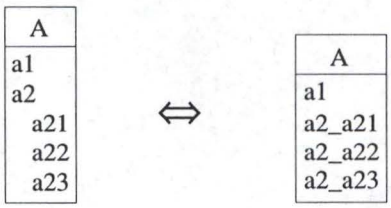


Mapping :

Telephone : corresp = Att-to-ET/ instance (producer,telephone)
Number : corresp = Att-to-ET/ instance (producer,telephone)
has : corresp = Att-to-ET/instance (producer,telephone)

5.1.9. Disaggregate

A compound attribute is replaced by its components using the Disaggregate transformation.



This is the aggregate inverse transformation.

No other requirements need to be fulfilled in order to perform this transformation on the source schema.

The mapping format for each attribute obtained by the Disaggregate transformation is :

- Disaggregate (*ent, coatt.att*) where *ent* is the ET that have the transformed compound attribute *coatt*. *Att* is the corresponding *coatt* attribute.

Example :

Source schema :

PRODUCER
Name
Telephone
Status
Address
Street
Number
Zip-code
City

Target schema :

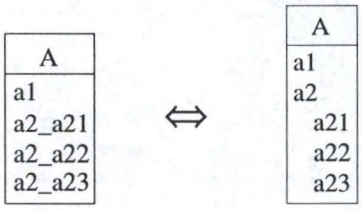
PRODUCER
Name
Telephone
Status
Add_Street
Add_Number
Add_Zip-code
Add_City

Mapping :

Add_street : corresp = Disaggregate (producer,address.street)
Add_number : corresp = Disaggregate (producer,address.number)
Add_zip-code : corresp = Disaggregate (producer,address.zip-code)
Add_city : corresp = Disaggregate (producer,address.city)

5.1.10. Aggregate

With Aggregate, a list of attributes is replaced by a compound attribute.



This is the disaggregate inverse transformation.

This transformation can be used on the source schema just as it is.

The mapping format for each compound attribute and its attributes obtained by the Aggregate transformation is :

- For each compound attribute : Aggregate (*ent, {att}*) where ent contains the attribute list {*att*}
- For each compound attribute attribute : Aggregate (*ent, att*) where *att* is the transformed attribute

Example :

Source schema :

PRODUCER
Name Telephone Status Street Number Zip-code City

Target schema :

PRODUCER

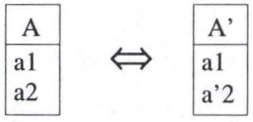
Name
Telephone
Status
Address
  Street
  Number
  Zip-code
  City

Mapping :

Address : corresp = Aggregate (producer, {street, number,zip-code,city})
Street : corresp = Aggregate (producer, Street)
Number : corresp = Aggregate (producer, Number)
Zip-code : corresp = Aggregate (producer, Zip-code)
City : corresp = Aggregate (producer, City)

5.1.11. Rename

The rename transformation is used to change an ET, RT or attribute name.



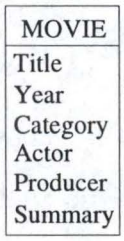
This transformation can be used without constraint on the source schema.

The mapping format for each TE, RT or attribute changed by the rename transformation is :

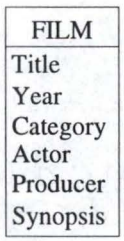
- For each ET : Rename (*ent*) where *ent* is the transformed ET
- For each RT : Rename (*rel*) where *rel* is the transformed RT
- For each attribute : Rename (*att*) where *att* is the transformed attribute

Example :

Source schema :



Target schema :

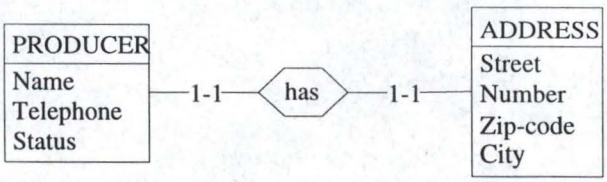


Mapping :

Film : corresp = Rename (movie)
Synopsis : corresp = Rename (Summary)

Sometimes, the same ET, attribute or RT of the source schema may have been transformed several times. In this case, the metaproperty of the attribute, ET or RT doesn't only contain one mapping but rather a list of mapping. If we compose the mappings that are in the metaproperty, we have the real correspondence between the source and the target construct.

Example :



The first transformation applied on this schema is ET-to-Att. The ADDRESS ET becomes a compound attribute in the PRODUCER ET. The schema looks like :

PRODUCER
Name
Telephone
Status
Address
Street
Number
Zip-code
City

Now the Address attribute is disaggregated using the disaggregate transformation. The schema becomes :

PRODUCER
Name
Telephone
Status
Street
Number
Zip-code
City

The metaproperty has therefore the values :

Street : Corresp = ET-to-Att (ADDRESS,Street,has)
Disaggregate (PRODUCER, Address.Street)

Number : Corresp = ET-to-Att (ADDRESS,Number,has)
Disaggregate (PRODUCER, Address.Number)

Zip-code : Corresp = ET-to-Att (ADDRESS,zip_code,has)
Disaggregate (PRODUCER, Address.Zip-code)

City : Corresp = ET-to-Att (ADDRESS,City,has)
Disaggregate (PRODUCER, Address.City)

5.2. Mapping analysis methodology

The mapping analysis is made in three steps :

- Metaproperties creation
- Mappings extraction
- Mappings insertion into the metaproperties.

5.2.1. Metaproperties creation

The metaproperty is the best way to add information for an ET, RT or attribute. A metaproperty is created for each ET, attribute and RT of the target conceptual schema. Its name is "*corresp*". The purpose of each *corresp* metaproperty is the storage of the mapping in the format defined above.
To add these metaproperties in the target repository, we use the Voyager 2 language.

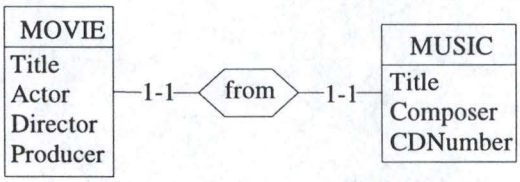
5.2.2. Mapping extraction

Before inserting the mappings into the metaproperties, it is necessary to extract them out of the transformation history. The DB-Main tool is used here in order to edit the historic into a log file, using the log edit option of the tool. Once this file is created, it is parsed using the Voyager 2 lexical analyzer. Key words are sought by the parser in order to find the correspondences between the source and the target schemas. Then, the converter generator stores in a temporary structure the information found. All these information are transformed in order to have the correct mapping format. Finally, the mappings are inserted into the metaproperties.

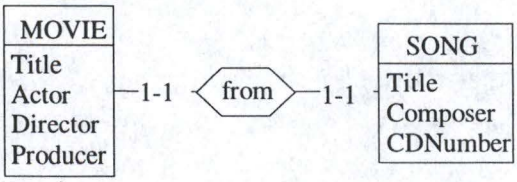
An important restriction has to be made in order to extract the correct mappings : the rename transformation must be the last one applied on the source schema. Without this condition, some name conflicts may appear and the metaproperty would give the wrong correspondence.

Example :

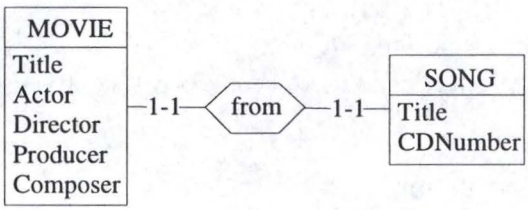
The following schema is going to be transformed using the split-merge transformation. In one case the name of the ET from which the attribute is taken is rename before the split-merge transformation is performed, and in the other case, after.
The source schema is :



In the first case, the rename transformation is performed first on the source schema. The MUSIC ET is renamed as SONG :



Next, the split-merge transformation is applied on the SONG ET : the Composer is now an attribute of the MOVIE ET :

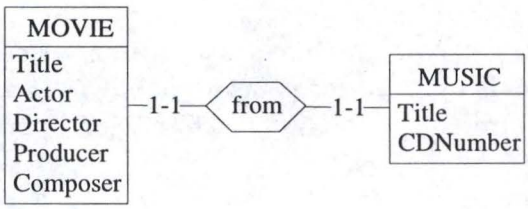


The mapping is therefore equal to :

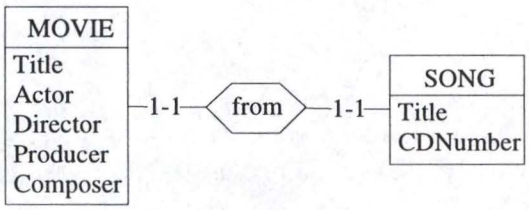
Composer : corresp = split-merge (song,producer,from)
Song : corresp = rename(music)

The extractor, in this case, will extract the attribute composer from a SONG ET in the source schema but this ET doesn't exist!

In the second case, the split-merge transformation is applied first. The Composer attribute is taken from the MUSIC ET and inserted into the MOVIE ET. The schema is now :



Next, the name of the MUSIC ET is changed. The schema becomes :



We have thus the same target schema
We see that, if the rename transformation is the last one applied, we have the same target schema, but this time, with the correct mapping :

Composer : corresp = split-merge(music,composer,from)

Song : corresp = rename(music)

The DB-to-DB converter will now extract the composer attribute from the MUSIC ET.

5.2.3. Mapping insertion

The final step of the mapping analyse process is the insertion of the mappings into the metaproperties. Each *corresp* metaproperty is filled with its corresponding mapping. If an element is not transformed, the metaproperty is not empty, it contains the name of the ET, RT or attribute[1] it is attached to. Once the mappings are stored into the metaproperties because the latter are easily accessible through Voyager 2 procedures or functions using predicative queries.

5.3. The AnalyseMap program

The AnalyseMap program designed to perform the mapping analysis is written in Voyager2. In its actual version, this program supports the following symmetrically reversible transformations :

- ET-to-Att
- split-merge
- Att-to-ET/instance
- disaggregate
- rename

Furthermore, only one transformation can be performed on the same construct.

The AnalyseMap source code in appendix D.

[1] The attribute name is prefixed by the ET name. E.g. : MOVIE.Title

Chapter 6
DB-to-DB converter generation

Chapter 6 : DB-to-DB converter generation

This chapter is dedicated to the explanation on how the DB-to-DB converter is generated on the basis of the analysis of the mappings. The program that generates the converter is written in Voyager2. The converter is a Java program[1].

The converter is divided in two modules : the extraction and the insertion modules. These modules are themselves divided in different steps as shown in figure 6.1. These modules are described below.

The DB-to-DB converter generator Voyager2 program is ConvertDB. Its source code can be found in appendix E.

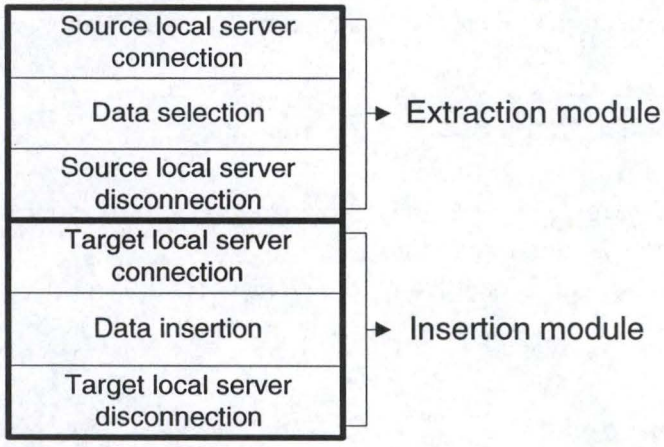


Figure 6.1. : DB-to-DB converter

6.1 Extraction module

The aim of the extraction module is to collect, on the basis of the mappings, the relevant data from the first DB. We can divide the extraction module in three components. The first thing to do is to create the part of the program devoted to the connection to the source DB through the local server. Afterwards, the next lines of code to generate are those which address the selection queries to the local server. The last thing to generate is the part of code designed to disconnect from the local server.

[1] We use the Java language because the local servers only support Java applications and the InterDB driver offers a JDBC like interface.

The extraction module corresponds to the GenExtractMod procedure of the ConvertDB program.

6.1.1 Connection to the source local server

To generate the connection to the source local server, we need to know some information, like the url of the server, the user name and password. These are collected by prompting questions on screen.

The connection to the source local server is made by the Connect procedure.

6.1.2 Data selection

Now that the lines dedicated to the connection have been written, the next part of the code to generate is the selection of the relevant data. To discover which data is likely to be picked up, we need to access the repository and collect the mapping in the meta-properties. Once this is done, we are able to create the selection queries that access the local server and store the data in ResultSet[2] Java objects.

The data selection and storage is achieved by the Extract procedure.

6.1.3 Disconnection to the source local server

The next instruction to generate is the disconnection from the source local server, which is fairly simple, and written in one line.

The disconnection is handled by the Disconnect procedure.

6.2 Insertion module

The insertion module fills in the target DB with the extracted data stored in the ResultSet java objects. This module can be parted in three components. The first one to be generated is the connection to the target DB through the local server. Afterwards, the next lines of code to produce are the insertion queries addressed at the target local server. Finally, the generator has to create the instruction that closes the connection.

The insertion module corresponds to the GenInsertMod procedure of the ConvertDB program.

[2] The rows that satisfy the conditions of a query are called the *result set*. The number of rows returned in a result set can be zero, one or many. One accesses the data in a result set one at a time, and a cursor provides the mean to do that. [Hamilton, 1997]. Let us note that the data are only physically extracted when the get method is invoked.

6.2.1 Connection to the target local server

Like for the connection to the source local server, we need to know the url of the server and the user name and password. The generator acts in the same way : it prompts questions on the screen. When all these information have been given, the connection instructions creation becomes fairly simple, just like pieces of a puzzle to rearrange in the right order.

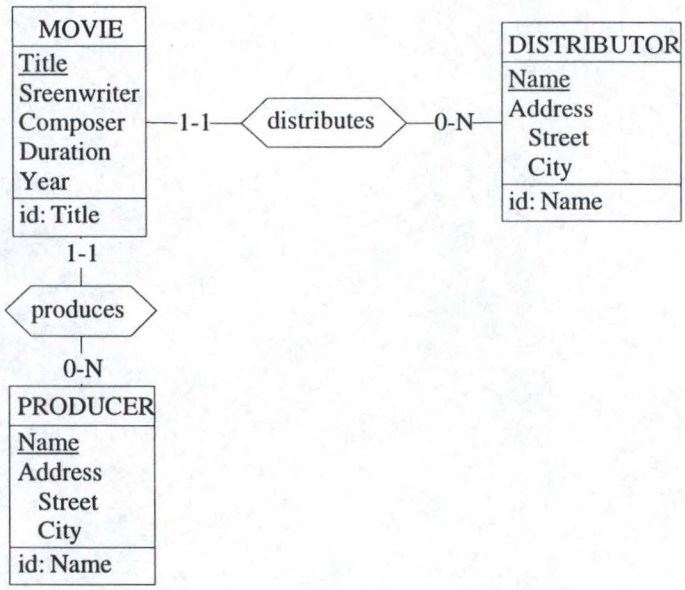
The connection to the target local server is performed by the same procedure as for the connection to the source local server : Connect

6.2.2 Data insertion

The next part of code to generate is related to the insertion of the data (stored in the ResultSet) into the target DB. We need to analyse the repository to bring to light where the data collected in the source DB are to be stored in the target DB. Once this is done, we are able to create the insertion queries that access the local server.

These queries must follow a predefined order. The first entities to be filled in are those which are never linked through roles with [1-1] cardinality. Then the converter loads the entities that are only linked with those previously filled in. We act this way to avoid an integrity constraint violation.

Example :



With this schema, the converter has first to insert the distributor and producer entities, then the movies and the links to the two other entities. Indeed, if a movie had been filled in without being linked to a distributor and a producer, an integrity constraint would have been violated.

The data insertion process is carried out by the Insert procedure.

6.2.3 Disconnection to the target local server

When all the data have been inserted in the target DB, the last thing to do is to create the disconnection instructions. Here again, these instructions are easy to generate.

The disconnection to the target local server is performed by... (can you believe it?) the famous Disconnect procedure.

Chapter 7
DB-to-XML converter generation

Chapter 7: DB-to-XML converter generation

The DB-to-XML converter is used to extract the data from a DB and to create a XML document containing all these data. The DB and XML document are both described by a conceptual schema. In this chapter, we explain the DB-to-XML converter generation process.
A Document Type Definition (DTD) gives the valid structure of the XML document, therefore, it is essential to dispose of it in the converter generation process.
This chapter first describes the DTD generation and then defines how to create the converter.

7.1. DTD generation

As explained in chapter 4, the DTD generation process is based on the target XML schema. The DTD contains the tree-structure of the XML document.

We first explain how to translate a conceptual schema into a DTD and then, we describe the DTD generation process.

7.1.1. Conceptual schema to DTD translation

The XML document has a tree structure, we have to place the ET and attributes in the nodes of the tree, i.e. to make them correspond to **elements**, while the RT are the links between the nodes. The link is in fact an **attribute** of the **element** with the **IDREF** type, what lays down the linked **element** to have an **attribute** with the **ID** type (to which the **IDREF** attribute refers). The root of the tree is always the name of the schema and its nodes are ETs whose nodes are their attributes (Figure 7.1.).

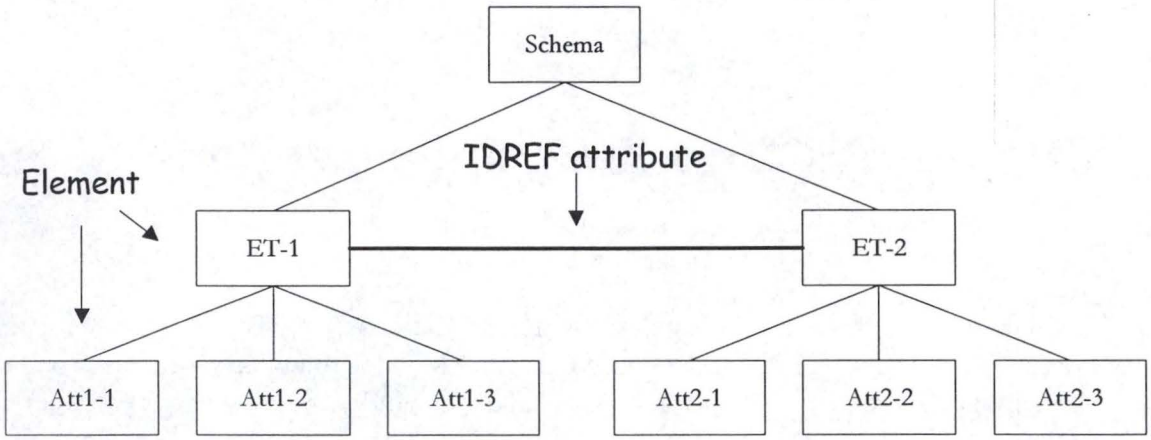


Figure 7.1. : XML document tree structure

The correspondence between the XML schema and the DTD constructs is shown in table 7.1.

XML model construct	DTD construct
ET	**Element**
Attribute	**Element**
RT	**Attribute**[1]
Is-a relationship	**Attribute**

Table 7.1. : Correspondence between XML schema and DTD

Each ET corresponds to one **element** composed of the ET attributes. Each role played by an ET in a RT is represented by an **attribute** in the corresponding **element**. There is an **attribute** for each is-a construct, it links the **element** with the **element** representing the supertype. The type of both **attributes** is **IDREF**.

In order to give the complete structure of the DTD, we use the BNF notation. A commented BNF description of the DTD is given below. The complete version can be found in appendix F.

```
DTD : := '<! DOCTYPE' schema '[' element_declaration ']>'
```

[1] The reader will pay attention to the attribute notion. An **attribute** is an **element** attribute while an attribute is an ET attribute.

A DTD is composed of a heading containing the name of the XML conceptual schema, and of the **elements** declaration.

```
element_declaration : := root_declaration (one_ET_declaration)*
(one_attribute_declaration)*
```

The element_declaration is composed of three declarations. The first one is the one for the tree root : the schema. The next two declarations are the ones for each ET and attributes. The format for each of them is different and given below.

```
root_declaration ::= '<!ELEMENT ' schema '(entity_list ')>'
entity_list := ET | entity_list
```

The **element** that corresponds to the schema is composed of the **elements** that represent each ETs.

```
one_ET_declaration : := '<! ELEMENT' ET '(' attributes_list ')>'
                        '<! ATTLIST' ET 'id ID #REQUIRED'
                                    (RT-ET reference '#' mandat-optional )*
                                    (isa-supertype IDREF # REQUIRED )*
                                    ('identifier CDATA #FIXED " ' identifier ' " ' ) ?
                                    'corresp CDATA #FIXED " ' mapping ' " >'
```

The **element** corresponding of one ET is defined as a sequence of **subelements** or **subgroups**. Each **subelement** is an atomic attribute, while the **subgroup** is for the compound attribute.

The **element attributes** are :

- the identifier id. This is not the ET identifier but a special XML identifier. This redundancy is necessary in order to identify each **element**. Indeed, we can reasonably imagine that two entities in different ET have the same value for their identifiers.
- a link to one or several **elements**. This link represents the role that plays the ET in the RT. The **attribute** type depends on the RT cardinality as it is explained below. There is such a link for each RT.
- a link to the **element** corresponding to the ET supertype. There is such a link for each is-a relationship of that ET.
- the ET primary identifier. This is a fixed **attribute** : its value is the ET identifier and it has not to be written into the XML document. This information not necessary for the DB-to-XML converter generator but one can imagine that the DTD may be used by a program that takes the data from a XML document and inserts them into a DB. In such a case, the name of the identifier is fundamental.

- the mapping. This is a also a fixed **attribute** whose value can be found the ET metaproperty "corresp".

```
attributes_list : := one_attribute | one_attribute ',' attributes_list
one_attribute : := attribute cardinality
cardinality : := ' ' | '?' | '*' | '+'
```

Each ET attributes are separated by a comma. If it is optional, it is followed by a '?', if it is a multivalued attribute, there is a '+' or a '*' afterwards according to whether the minimal cardinality is 1 or 0. Unfortunately, it is impossible to give the right maximum cardinality, it is always regarded as equal to N.
This format is also used for the compound attribute attributes.

```
reference ::= 'IDREF' | 'IDREFS'
mandat-optional ::= 'REQUIRED' | 'IMPLIED'
```

The reference and the mandat-optional value depends on the role cardinality is shown in table 7.2.

Cardinality	reference	mandat-optional
0-n	IDREFS	IMPLIED
1-n	IDREFS	REQUIRED
0-1	IDREF	IMPLIED
1-1	IDREF	REQUIRED

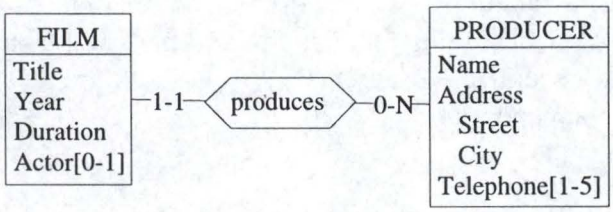
Table 7.2. : The reference and mandat-optional values

```
one_attribute_declaration : :=          atomic_attribute_declaration          |
compound_attribute_declaration
atomic_attribute_declaration ::=
            '< ! ELEMENT' attribute '#PCDATA>
            < ! ATTLIST attribute 'corresp CDATA #FIXED " ' mapping ' " >'
compound_attribute_declaration ::=
            '<! ELEMENT' attribute '(' attributes_list ')>
            < ! ATTLIST' attribute 'corresp CDATA #FIXED " ' mapping ' " >'
            (one_attribute_declaration)+
```

The ET attribute declaration depends on the attribute type. If it is an atomic attribute, it is represented by an element whose type is PCDATA. In the other case, if it is a compound attribute, it corresponds to a element defined as a sequence of subelements and/or subgroups that represent its attributes. Both elements have one attribute : the mapping between the target schema attribute and the source schema construct.

Example :

The following XML model :



is translated into the following DTD :

```
<! DOCTYPE cinema [
<! ELEMENT film (title, year, duration, actor?)>
<! ATTLIST film id ID #REQUIRED
                produces-producer IDREF #REQUIRED
                corresp CDATA #FIXED " ">
<! ELEMENT title #PCDATA>
<! ATTLIST title corresp CDATA #FIXED " ">
<! ELEMENT year #PCDATA>
<! ATTLIST year corresp CDATA #FIXED " ">
<! ELEMENT duration #PCDATA>
<! ATTLIST duration corresp CDATA #FIXED " ">

<! ELEMENT actor #PCDATA>
<! ATTLIST actor corresp CDATA #FIXED " ">
<! ELEMENT producer (Name, Address, telephone+) >
<!ATTLIST producer id ID #REQUIRED
                produces-film IDREFS #IMPLIED
                corresp CDATA #FIXED " ">
<!ELEMENT name #PCDATA>
<! ATTLIST name corresp CDATA #FIXED " ">
<!ELEMENT address (street, city)>
<! ATTLIST address corresp CDATA #FIXED " ">
<!ELEMENT street #PCDATA>
<! ATTLIST street corresp CDATA #FIXED " ">
<!ELEMENT city #PCDATA>
<! ATTLIST city corresp CDATA #FIXED " ">
<!ELEMENT telephone #PCDATA>
<! ATTLIST telephone corresp CDATA #FIXED " ">
]>
```

7.1.2. DTD generation

The DTD generation follows the strategy defined above. The first step is to query the target schema repository in order to retrieve all the information needed, i.e. the ET's, RT's, attributes, is-a relationships, identifiers and the mappings. Finally, all these information are written, according to the defined format, in the output file, the DTD.

The DTD is generated by the DTD Voyager2 program. The source code of this program is given in appendix G.

7.2. DB-to-XML converter generation

In this section, we describe the DB-to-XML converter and explain how and where the generator takes the information in order to create it.

The DB-to-XML converter is composed of two modules as described in figure 7.2.

The converter is generated by the ConvertXML program. This program can be found in appendix H.

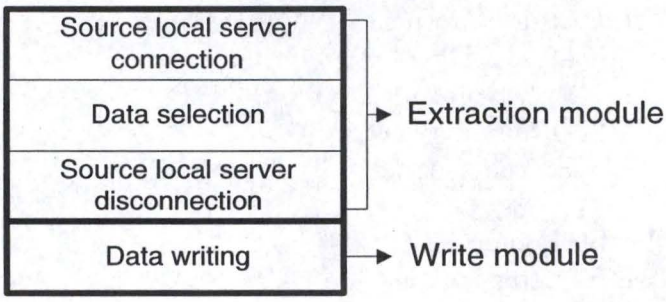


Figure 7.2. : DB-to-XML converter

7.2.1. Extraction module

The extraction module has three functions : source local server connection, data extraction and the source local server disconnection.

This converter module is the same than the DB-to-DB extraction module described in section 6.1.

7.2.2. Write module

The write module of the DB-to-XML converter has only one function : the data writing into the XML document.

In order to create the XML document and to insert the data into it, the converter has to know the document structure. The generator has thus to analyse the target schema repository in order to find the constructs it has to fill. It also has to parse the DTD in order to know how to insert the data in such a way that the XML document is well formed and valid.

The write module corresponds to the GenWriteMod procedure.

Chapter 8
Case Study

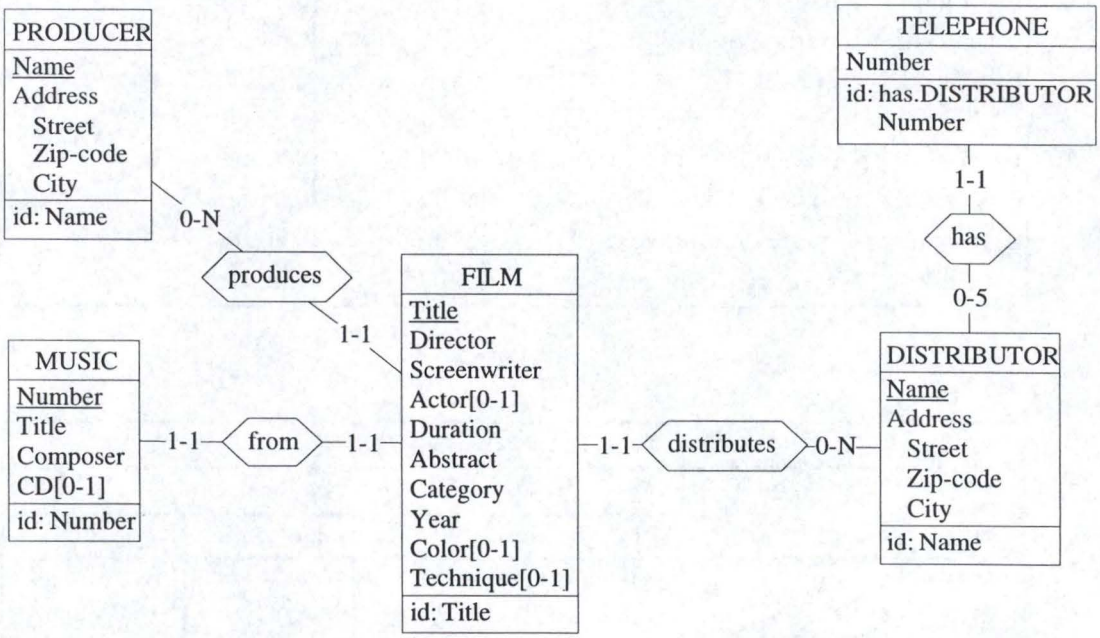
Chapter 8 : Case study

In this chapter, we present a case study for the converter generators. First, we present the case and then we show the steps to follow in order to create the DB-to-DB and the DB-to-XML converters. Some data are in the source DB and we will follow their route in their transfer process to the target DB or XML document.

8.1. Case presentation

For the case study, we stay in the wonderful movie world. The source conceptual schema shown below respects the CM constructs. It is composed of five ET's :

- Film
- Producer
- Music
- Distributor
- Telephone



Each film is characterised by a title, which identifies it among the other ones, a director, an actor, a screenwriter, the duration, expressed in minutes, an abstract, a category, the release year, the color and the technique. Some movie may not have an actor (actress). The optional attribute color is a boolean, that means that its only possible values are true and false (represented respectively by 1 and 0). The technique attribute is for cartoons only, it represents the drawing technique used by the drawer.
The producer and the distributor have the same attributes : the name which is their identifier, and the compound attribute address. The address is composed of a street, a zip-code and the name of the city.
Each distributor may have a most 5 telephone numbers.
The music is characterised by a number, the title of the song or theme and the composer. The music may be released on a CD, in this case, the CD number is given into the CD attribute.

This conceptual schema is the representation of a DB. In this source DB, there are some data that are shown in the following tables. Each columns represents an attribute. The links are also written in these tables as columns whose values are the identifier value of the related entity.

PRODUCER			
Name	Address		
	Street	Zip-code	City
Cameron	rue de Bruxelles, 22	5000	Namur
Spielberg	chaussée de Nivelles, 56	5140	Sombreffe
Eastwood	rue du Condroz, 1	5590	Ciney
Burton	Station, 17	5575	Gedinne
Di Novi	Avenue du diamant, 96	1030	Bruxelles
Besson	Avenue Bois l'Evêque, 33	5100	Wierde
Godfroid	rue du Chenois, 186	6000	Charleroi
Clements	Square de Quinaux, 19	5100	Wierde
Karmitz	Place franco-belge, 5	6200	Chatelet

DISTRIBUTOR			
Name	Address		
	Street	zip_code	City
20th Century Fox	Hollywood Blvd, 150	9000	Los Angeles
Universal	Hollywood Blvd, 151	9000	Los Angeles
Warner	Hollywood Blvd, 152	9000	Los Angeles
Buena Vista	Madison Avenue, 15	8000	New York

TELEPHONE	
Number	DISTRIBUTOR
1230450890	20th Century Fox
1230756452	Universal
1230085236	Warner
1500365000	Buena Vista
1230450891	20th Century Fox
1230756453	Universal
1230085237	Warner
1500365001	Buena Vista
1230085238	Warner
1500365002	Buena Vista

MUSIC			
Number	Title	Composer	CD
15492	Titanic BO	Horner	233100
12250	E.T.	Williams	233088
23560	Madison	Niehaus	233500
16500	Jurassic soundtrack	Williams	233087
33201	Mars Attacks	Elfman	
23655	The nightmare before Christmas	Elfman	233456
14587	Aladdin	Menken	233962
98664	Rouge	Priesner	145550
16545	Bleu	Priesner	145500

FILM												
Title	Director	Screenwritor	Actor	Dura.	Abstract	Category	Year	Color	Technique	PRODUC.	MUSIC	DISTRIBUTOR
Titanic	Cameron	Cameron	Di Caprio	195	The famous boat sinks	drama	1997			Cameron	15492	20th Century Fox
E.T.	Spielberg	Daviau	Barrymore	115	A kind E.T. on earth	Comedy	1982			Spielberg	12250	Universal
The bridges of Madison County	Eastwood	LaGravenese	Eastwood	135	Four days for an eternal love	drama	1997			Eastwood	23560	Warner
Jurassic Park	Spielberg	Crichton	Goldblum	126	Dinosaurs are alive	suspense	1993			Spielberg	16500	Universal
Mars Attacks!	Burton	Gems	Nicholson	103	Aliens attack earth	science-fiction	1996			Burton	33201	Warner
The nightmare before Christmas	Burton	McDowell		74	Christmas tale	cartoon	1993	1	animation	Di Novi	23655	Buena Vista
Aladdin	Clements	Clements		90	Aladdin has some problems	cartoon	1992	1	drawing	Clements	14587	Buena Vista
Trois couleurs : Bleu	Kieslowski	Kieslowski	Binoche	98	First part of the trilogy	drama	1993			Karmitz	16545	Universal

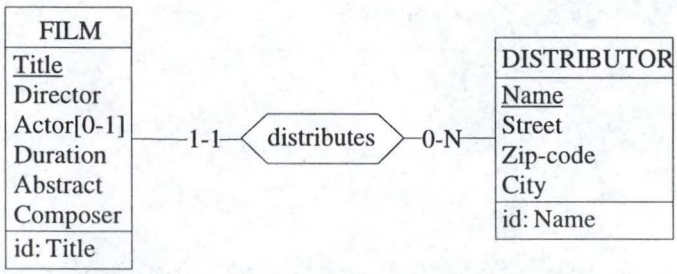
8.2. DB-to-DB case

In the DB-to-DB case, some data are taken from the source DB and inserted into the target DB. The target schema is a subset of the source schema. It is obtained by applying some of the symmetrically reversible transformations (exposed in chapter 5) on the source conceptual schema. The DB-Main transformation tool helps the user in his task to transform the source schema.

In our case study, the first operation performed on the source schema is the deletion of ET and attributes : the TELEPHONE ET is removed, just like the attributes screenwriter, category, year, color and technique of the FILM ET.
The next step of the schema transformation is the symmetrically reversible transformations performance. Is this case, two of them are made :

- split merge : the attribute composer is taken from the MUSIC ET and inserted into the FILM ET.
- disagregation : the compound attribute Address in the DISTRIBUTOR ET is disaggregated into the attributes Street, Zip-code and City.

The target schema is therefore :



The target DB is created on the basis of this target schema using the DB-Main forward engineering process.

Now that he has the two DB and schemas, the user has to follow this scenario :

- execute the AnalyseMap Voyager2 program
- execute the ConvertDB Voyager2 program
- compile the dbtodbconverter Java program
- execute the dbtodbconverter Java program

These programs are described below.

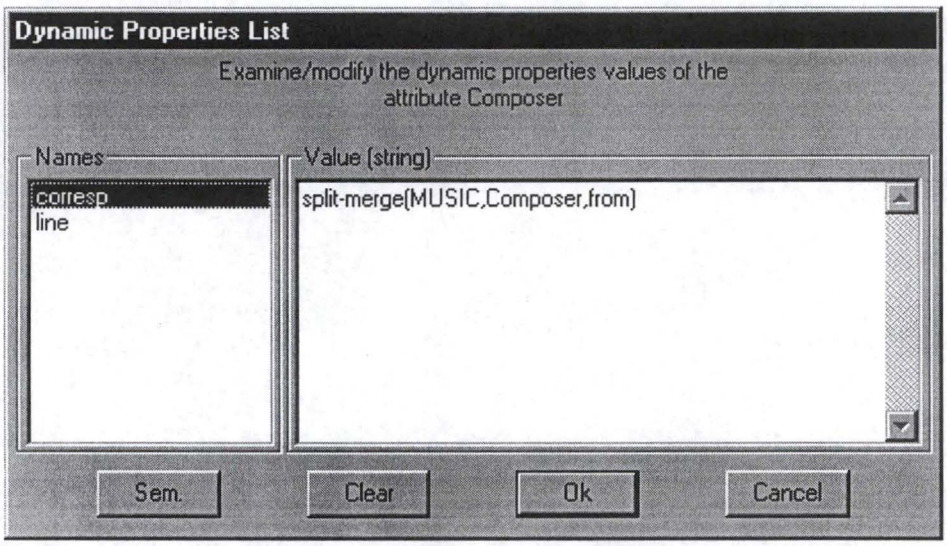
AnalyseMap

The AnalyseMap program, given in appendix D, analyses the mappings between the source and the target schema. This Voyager2 program uses a log file, that contains all the performed transformations, as input. The log file of the target schema is given in appendix I.

AnalyseMAp creates the metaproperty "corresp" and gives it as value the mapping representation. If an ET or attribute is not concerned by any of the transformations, than its mapping is just its name. For the attributes Composer, Street, Zip-code and City, the mapping is :

- MOVIE.Composer : corresp = split-merge (MUSIC, composer, from)
- DISTRIBUTOR Street : Corresp = disaggregate (DISTRIBUTOR, Address.Street)
- DISTRIBUTOR Zip-code : Corresp = disaggregate (DISTRIBUTOR, Address.zip-code)
- DISTRIBUTOR city : Corresp = disaggregate (DISTRIBUTOR, Address.City)

The mapping can be read in the DB-Main case tool. E.g., the mapping for Composer is given is the following window :



ConvertDB

On the enriched target schema, the user executes the ConvertDB program. The source code of this program is given in appendix E. ConvertDB generates the source code of the Java program named dbtodbconverter.java.

dbtodbconverter

The dbtodbconverter Java program is the data converter that extracts the relevant data from the source schema and inserts them into the target schema. The source code is given in appendix J.
Once this program has been compiled and executed by the user, the target DB contains the following data :

DISTRIBUTOR			
Name	Street	Zip-code	City
20th Century Fox	Hollywood Blvd, 150	9000	Los Angeles
Universal	Hollywood Blvd, 151	9000	Los Angeles
Warner	Hollywood Blvd, 152	9000	Los Angeles
Buena Vista	Madison Avenue, 15	8000	New York

FILM						
Title	Director	Actor	Dur.	Abstract	Composer	DISTRIBUTOR
Titanic	Cameron	Di Caprio	195	The famous boat sinks	Horner	20th Century Fox
E.T.	Spielberg	Barrymore	115	A kind E.T. on earth	Williams	Universal
The bridges of Madison County	Eastwood	Eastwood	135	Four days for an eternal love	Niehaus	Warner
Jurassic Park	Spielberg	Goldblum	126	Dinosaurs are alive	Williams	Universal
Mars Attacks!	Burton	Nicholson	103	Aliens attack earth	Elfman	Warner
The nightmare before Christmas	Burton		74	Christmas tale	Elfman	Buena Vista
Aladdin	Clements		90	Aladdin has some problems	Menken	Buena Vista
Trois couleurs : Bleu	Kieslowski	Binoche	98	First part of the trilogy	Priesner	Universal

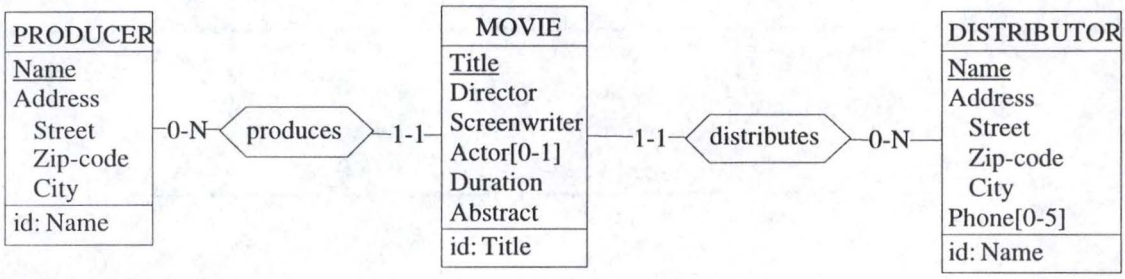
8.3. DB-to-XML case

In the DB-to-XML study case, the XML schema is based on the same source schema. Some operations are performed on it in order to create the XML target schema. The first thing to do is to delete the ET and attributes that are not necessary. In this case, we remove the MUSIC ET. The next step is the performance of symmetrically reversible transformation. We have decided to make two changes :

- et-to-att : The TELEPHONE ET is transformed into the multivalued attribute Phone in the DISTRIBUTOR ET.
- rename : the name of the FILM ET is changed, it is "movie" now.

Rename is the last transformation performed because it may create some name conflicts.
All these transformations are performed using the DB-Main transformation tool.

The XML target schema obtained is like :



Now that he has the two schemas, the user has to follow this scenario :

- execute the AnalyseMap Voyager2 program
- execute the dtd Voyager2 program
- execute the ConvertXML Voyager2 program
- compile the dbtoxmlconverter Java program
- execute the dbtoxmlconverter Java program

All these programs are described below.

AnalyseMap

The AnalyseMap Voyager2 program analyses the log file (given in appendix D), creates the metaproperty "corresp" and fills it with the mapping value.
The metaproperty, for the ET and attributes that have not been changed, simply contains their name whereas, for the transformed attribute Phone and the MOVIE ET, its value is :

- Movie : corresp = rename (FILM)
- Phone : corresp = ET-to-Att (TELEPHONE, Number, has)

The user has now an enriched target schema.

DTD

On the enriched target schema obtained, the user executes the dtd Voyager2 program, given in appendix G, that generates the DTD of the XML document described by the schema. The DTD, named movie.dtd, is given below.

```
<!DOCTYPE Movie [
<!ELEMENT Movie (DISTRIBUTOR, MOVIE, PRODUCER)
<! ELEMENT DISTRIBUTOR (DISTRIBUTOR.Name, DISTRIBUTOR.Address,
    DISTRIBUTOR.Phone*)>
<!ATTLIST DISTRIBUTOR
```

```
        identifiant CDATA #FIXED "Name"
        distributes.MOVIE IDREFS #IMPLIED
        id ID #REQUIRED
        Corresp CDATA #FIXED "DISTRIBUTOR">
<!ELEMENT DISTRIBUTOR.Name #PCDATA>
<!ATTLIST DISTRIBUTOR.Name
        corresp CDATA #FIXED "DISTRIBUTOR.Name">
<!ELEMENT DISTRIBUTOR.Phone #PCDATA>
<!ATTLIST DISTRIBUTOR.Phone
        corresp CDATA #FIXED "et-to-att(TELEPHONE,Number,has)">
<!ELEMENT DISTRIBUTOR.Address( DISTRIBUTOR.Address.Street,
DISTRIBUTOR.Address.Zip-code,    DISTRIBUTOR.Address.City ) >
<!ATTLIST DISTRIBUTOR.Address
        corresp CDATA #FIXED "DISTRIBUTOR.Address">
<!ELEMENT DISTRIBUTOR.Address.Street #PCDATA>
<!ATTLIST DISTRIBUTOR.Address.Street
        corresp CDATA #FIXED "DISTRIBUTOR.Address.Street">
<!ELEMENT DISTRIBUTOR.Address.Zip-code #PCDATA>
<!ATTLIST DISTRIBUTOR.Address.Zip-code
        corresp CDATA #FIXED "DISTRIBUTOR.Address.Zip-code">
<!ELEMENT DISTRIBUTOR.Address.City #PCDATA>
<!ATTLIST DISTRIBUTOR.Address.City
        corresp CDATA #FIXED "DISTRIBUTOR.Address.City">

<! ELEMENT MOVIE (MOVIE.Title, MOVIE.Director, MOVIE.Screenwriter,
   MOVIE.Actor?, MOVIE.Duration, MOVIE.Abstract)>
<!ATTLIST MOVIE
        identifiant CDATA #FIXED "Title"
        distributes.DISTRIBUTOR IDREF #REQUIRED
        produces.PRODUCER IDREF #REQUIRED
        id ID #REQUIRED
        Corresp CDATA #FIXED "rename(FILM)">
<!ELEMENT MOVIE.Title #PCDATA>
<!ATTLIST MOVIE.Title
        corresp CDATA #FIXED "MOVIE.Title">
<!ELEMENT MOVIE.Director #PCDATA>
<!ATTLIST MOVIE.Director
        corresp CDATA #FIXED "MOVIE.Director">
<!ELEMENT MOVIE.Screenwriter #PCDATA>
<!ATTLIST MOVIE.Screenwriter
        corresp CDATA #FIXED "MOVIE.Screenwriter">
<!ELEMENT MOVIE.Actor #PCDATA>
<!ATTLIST MOVIE.Actor
        corresp CDATA #FIXED "MOVIE.Actor">
<!ELEMENT MOVIE.Duration #PCDATA>
<!ATTLIST MOVIE.Duration
        corresp CDATA #FIXED "MOVIE.Duration">
<!ELEMENT MOVIE.Abstract #PCDATA>
<!ATTLIST MOVIE.Abstract
        corresp CDATA #FIXED "MOVIE.Abstract">

<! ELEMENT PRODUCER (PRODUCER.Name, PRODUCER.Address)>
```

```
<!ATTLIST PRODUCER
        identifiant CDATA #FIXED "Name"
        produces.MOVIE IDREFS #IMPLIED
        id ID #REQUIRED
        Corresp CDATA #FIXED "PRODUCER">
<!ELEMENT PRODUCER.Name #PCDATA>
<!ATTLIST PRODUCER.Name
        corresp CDATA #FIXED "PRODUCER.Name">
<!ELEMENT PRODUCER.Address( PRODUCER.Address.Street,
PRODUCER.Address.Zip-code, PRODUCER.Address.City ) >
<!ATTLIST PRODUCER.Address
        corresp CDATA #FIXED "PRODUCER.Address">
<!ELEMENT PRODUCER.Address.Street #PCDATA>
<!ATTLIST PRODUCER.Address.Street
        corresp CDATA #FIXED "PRODUCER.Address.Street">
<!ELEMENT PRODUCER.Address.Zip-code #PCDATA>
<!ATTLIST PRODUCER.Address.Zip-code
        corresp CDATA #FIXED "PRODUCER.Address.Zip-code">
<!ELEMENT PRODUCER.Address.City #PCDATA>
<!ATTLIST PRODUCER.Address.City
        corresp CDATA #FIXED "PRODUCER.Address.City">
]>
```

ConvertXML

The ConvertXML Voyager2 program uses the DTD and the enriched XML schema in order to generate the source code of the DB-to-XML converter named dbtoxmlconverter.

dbtoxmlconverter

The dbtoxmlconverter Java program extracts the relevant data form the source DB and write them in the XML document. The source code of that program is given in appendix L.

Once run and executed, the Java program creates a XML document. A piece of that document is shown below, the entire document is given in appendix M.

```
<? XML version ="1.0"?>
<!DOCTYPE Movie SYSTEM  "Movie.dtd">

<DISTRIBUTOR id = "1" distributes.MOVIE = "5">
        <DISTRIBUTOR.Name> 20th Century Fox </DISTRIBUTOR.Name>
        <DISTRIBUTOR.Address>
                <DISTRIBUTOR.Address.Street>Hollywood Blvd, 150
                        </DISTRIBUTOR.Address.Street>
                <DISTRIBUTOR.Address.Zip-code>9000
                        </DISTRIBUTOR.Address.Zip-code>
                <DISTRIBUTOR.Address.City> Los Angeles </DISTRIBUTOR.Address.City>
        </DISTRIBUTOR.Address>
        <DISTRIBUTOR.Phone>1230450890</DISTRIBUTOR.Phone>
```

```
        <DISTRIBUTOR.Phone>1230450891</DISTRIBUTOR.Phone>
</DISTRIBUTOR>

...


<MOVIE id ="5" produces.PRODUCER = "13" distributes.DISTRIBUTOR = "1">
        <MOVIE.Title>Titanic </MOVIE.Title>
        <MOVIE.Director>Cameron </MOVIE.Director>
        <MOVIE.Screenwriter>Cameron </MOVIE.Screenwriter>
        <MOVIE.Actor>Di Caprio </MOVIE.Actor>
        <MOVIE.Duration>195 </MOVIE.Duration>
        <MOVIE.Abstract>The famous boat sinks </MOVIE.Abstract>
</MOVIE>

....


</PRODUCER id="13" produces.MOVIE = "5">
        <PRODUCER.Name>Cameron</PRODUCER.Name>
        <PRODUCER.Address>
                <PRODUCER.Address.Street>rue de Bruxelles, 22
                        </PRODUCER.Address.Street>
                <PRODUCER.Address.Zip-code>5000 </PRODUCER.Address.Zip-code>
                <PRODUCER.Address.City>Namur </PRODUCER.Address.City>
        </PRODUCER.Address>
</PRODUCER>
```

Some modifications may be made on this document by the user in order to give instructions for the display of the XML document. In the following pages, we give an example of presentation. The distributor, movie and producer elements are listed and the links between the elements are represented by hyperlinks.

Cinema

-Distributor-

20th Century Fox

Address : Hollywood Blvd, 150 9000 Los Angeles
Phone : 1230450890 or 1230450891
distributes : Titanic

Universal

Address : Hollywood Blvd, 151 9000 Los Angeles
Phone : 1230756452 or 1230756453
Distributes : E.T., Jurassic Park, Trois couleurs : bleu

-Movie-

Titanic

Director : Cameron
Screenwriter Cameron
Actor : Di Caprio
Duration : 195 min
Abstract : The famous boat sinks
Producer : Cameron
Distributor : 20Th Century Fox

E.T.

Director ; Spielberg
Sreenwriter : Daviau
Actor : Barrymore
Duration: 115 min
Abstract : A kind E.T. on earth
Producer : Spielberg
Distributor : Universal

Jurassic Park

Director : Spielberg
Sreenwriter : Crichton
Actor : Goldblum
Duration : 126 min
Abstract : Dinosaurs are alive
Producer : Spielberg
Distributor : Universal

Trois couleurs : bleu

Director : Kieslowski
Screenwriter : Kieslowski
Actor : Binoche
Duration : 98 min
Abstract : Fist part of the trilogy
Producer : Karmitz
Distributor : Universal

-Producer-

Cameron

Address : rue de Bruxelles, 22 5000 Namur
Produces : Titanic

Spielberg

Address : chaussée de Nivelles, 56 5140 Sombreffe
Produces : E.T., Jurassic Park

Karmitz

Address : place franco-belge, 5 6200 Chatelet
Produces : Trois couleurs : Bleu

Conclusion

Conclusion

We have introduced the concepts of data migration and the problems that arise in this particular research area. We also outlined the notions of database heterogeneity and schematic heterogeneity for which we gave a classification of the conflicts one may encountered. We briefly defined schema integration, view integration and schema translation and delineate the context in which our work is situated.

Thereafter, we described the models we used, i.e. the Entity-Relationship and the XML model. Considering that we did work with a simplified E-R model (i.e. the Conceptual Model), in order to restrain the risk of ambiguity problem, we had to define the constructs we authorised. The transfer process being performed on the basis of the conceptual level of the DBs, we needed a query language that was able to address directly at the conceptual schemas : the CQL (Conceptual Query Language).

Since the framework was drawn, we were able to outline an architecture for the two cases of our converter (DB-to-DB and DB-to-XML) and concisely define the tools that support them. The architecture showed that the converter is located between the source and the target schemas and is made up of two components : the extraction module (common for the two cases of the converter) and the insertion module (for the DB-to-DB case) or the write module (for the DB-to-XML case).

The next thing we did was to describe the converter generation methodology, here again we had to divide it in two cases according to the type of converter that was generated. The two main processes of the converter generation were the mapping analysis and the generation in itself. An important part of the mapping analysis process was to examine the transformations performed between the source and target schemas, hence the need to introduce a section devoted to the set of symmetrically reversible transformations we could use. We also depicted the generator support tools, i.e. the DB-Main repository, the history manager and the Voyager2 programming language.

Right after, we developed more thoroughly the mapping analysis process. We exposed the mapping representation for the transformations we allowed (which were a subset of the set of transformations we previously introduced) in order to comply with the CM and XML model. Then we gave the mapping analysis methodology which was divided in metaproperties creation, mapping extraction and mapping insertion; and we briefly presented the program we implemented in order to physically perform the analysis.

The next step of our work was to describe in a more exhaustive way the converter generation process, first for the DB-to-DB case and then for the DB-to-XML case. For the latter, we only had to explain the second part of the process, i.e. the write module ; the first being identical to the DB-to-DB case first part. For both of the cases, we included a section to present the programs that actually perform the generation process.

Finally, the last thing we exposed was a small case study, including a scenario showing how the converters are really generated

Our work gives another brick in the huge wall of the data migration problem. We provide a tool that eases the transfer process among two databases but only if the target schema has been build from the source. Another limitation is that we had to restrict the set of transformations that can be performed on the source schema to obtain the target. That was necessary as the E-R model was semantically too rich and, thus, provides a great risk of ambiguity problems.

Nevertheless the solution we propose is rather original as we work directly on the conceptual level of the DB's and knowing only that the target schema was the subset of the source, we are able to retrieve the correspondences between the two schemas and then automatically produce the code for the converter.
Another interesting achievement of our thesis, is the useful tool we gave to automatically translate the contents of a physical database into a XML document.

An enhancement that could later be made would be to limit the restrictions as far as it is possible, for example to allow multiple transformations on the same attribute. Another would be to retrieve the correspondences between a source schema and a target that is no more subset of the source.

Bibliography

[Ahmed, 1991] Ahmed R., et al., *The Pegasus Heterogeneous Multidatabase System*, IEEE Computer 24(12) : 19-27(1991)

[Batini, 1986] Batini C., Lenzerini M., Navathe B., *A Comparative Analysis of Methodologies for Database Schema Integration,* ACM Computing Surveys 18(4) : 323-364, December 1986

[Bourret] *Declaring elements and attributes in an XML DTD*, Ron Bourret, Darmstadt University of Technology.
http://www.informatik.tu-darmstadt.de/DVS1/staff/bourret/xml/xmldtd.html

[Brodie, 1995] Brodie M.L., Stonebraker M., *Migrating Legacy Systems*, Morgan Kaufmann Publishers, San Francisco, 1995

[DB-Main, 1997] Hainaut J.L., *Database reverse engineering, Problems, Methods and Tools*, FUNDP, March 1997.

[DB-Main, 1998] Hainaut J.L., *The DB-Main database engineering CASE tool, version 4, Functions overview,* FUNDP, November 1998.

[Englebert, 1998] Englebert V., *Voyager 2 Reference manuel, Version 4 Release 0,* FUNDP, November 1998

[Hainaut, 1994] Hainaut J.L., *BD et Modèles de calcul, outils et méthodes pour l'utilisateur*, FUNDP, iia, 1994.

[Hainaut, 1995] Hainaut J.L., *Transformation-based database engineering*, VLDB'95, September 95.

[Hamilton, 1997] Hamilton G., Cattell R., Fisher M., *JDBC Database Access with Java : A Tutorial and Annotated Reference*, Addison Wesley Longman, Reading, 1997

[Ioannidis, 1993] Ioannidis Y.E., Miller R.J., Ramakrishnan R., *The Use of Information Capacity in Schema Integration and Translation*. VLDB 1993: 120-133

[Kim, 1991] Kim W., Seo J., *Classifying Schematic and Data Heterogeneity in Multidatabase Systems*. IEEE Computer 24(12): 12-18 (1991)

[Mc Brien 1997] Mc Brien P., Poulovassilis A., *A Formal Framework for ER Schema Transformation*, in Proceedings of ER'97, volume 1331 of LNCS : 408-421, 1997

[Microsoft,1998] Microsoft Corporation, *XML : Enabling next-generation Web applications*, April 1998.
http://msdn.microsoft.com/xml/articles/xmlwp2.asp

[Miller, 1993] Miller R.J., Ioannidis Y.E., Ramakrishnan R., *Understanding Schemas*. RIDE-IMS 1993: 170-173

[Miller, 1994] Miller R.J., Ioannidis Y.E., Ramakrishnan R., *Schema equivalence in heterogeneous systems: bridging theory and practice*. IS 19(1): 3-31 (1994)

[Miller, 1998] Miller R.J., *Using Schematically Heterogeneous Structures*, in ACM SIGMOD'98 conference, Seattle : 189-200

[Miller, 1999] Miller R.J., Haas L.M., et al., *Transforming Heterogeneous Data with Database Middleware: Beyond Integration*. IEEE Data Engineering Bulletin 22(1): 31-36 (1999)

[Özsu, 1991] Özsu T., Valduriez P., *Principles of Distributed Database Systems*, Prentice-Hall, EngleWood Cliffs, 1991

[Saltor, 1997] Saltor F., Rodriguez E., *On Intelligent Access to Heterogeneous Information*, in Proceedings of the 4th KRDB Workshop Athens, Greece, 30-August-1997

[Sheth, 1999] Shet A., Rusinkiewicz M., Elmagarmid A., *Management of Heterogeneous and Autonomous Database Systems,* Morgan Kaufmann Publishers, San Francisco, 1999

[Thiran, 1998] Thiran Ph., Hainaut J-L., Bodart S., Deflorenne A., Hick J-M., *Interoperation of Independent, Heterogeneous and Distributed Databases. Methodology and CASE Support: the InterDB Approach*. CoopIS 1998: 54-63

[Thiran,1999a] Thiran Ph., Chougrani A., Hainaut J.L., *InterDB driver, conceptual server with JDBC,* FUNDP, july 1999.

[Thiran, 1999b] Thiran Ph., Hainaut J-L., Hick J-M., Chougrani, A., *Generation of Conceptual Wrappers for Legacy Databases*, in Proceedings of the DEXA'99 conference, Florence, September 1999

[Walsh, 1998] Norman Walsh, *A technical introduction to XML*, October 1998.
http:// www.xml.com/xml/pub/98/10/guide0.html

[Walsh, 1999] Norman Walsh, *Schemas for XML,* July 1999.
http://www.xml.com/xml/pub/1999/07/schemas/index.html

[W3C, 1998] World Wide Web Consortium, *XML-QL : A query language for XML*, submission to the World Wide Web Consortium, August 1998.
http://www.w3.org/TR/NOTE-xml-ql

[W3C,1999a] World Wide Web Consortium, *XML Schema Part 1: Structures*, May 1999.
http://www.w3.org/1999/05/06-xmlschema-1/

[W3C,1999b] World Wide Web Consortium, *XML Schema Part 2: Datatypes*, May 1999.
http://www.w3.org/1999/05/06-xmlschema-2/

Appendix

Appendix A : Transformation plans

Definition

A transformation plan is an algorithm composed of steps of the form :

for each $o \in O$, such that $P(o)$, do $T_{\Sigma i}(o)$;

where O is an object type and P a predicate.
This algorithm gives what transformations to apply, in what order and on what objects.

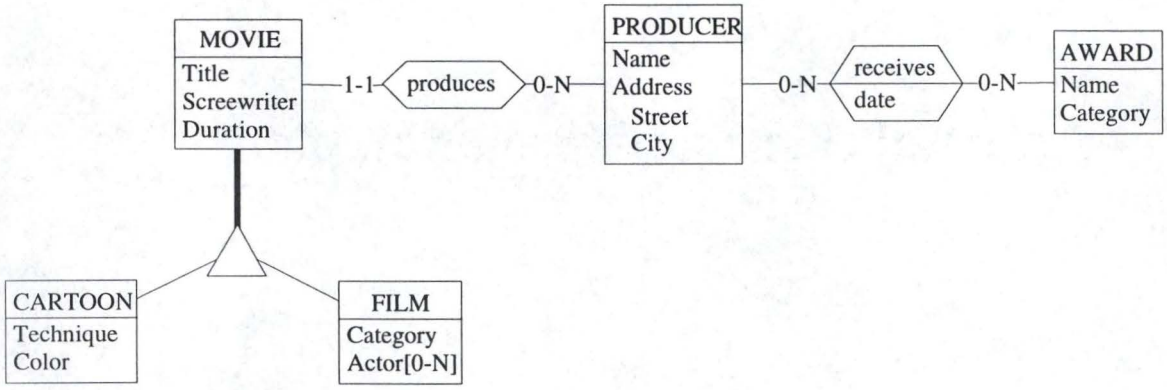
Transformation scripts

The following transformation script describes a transformation plan that transforms a ER schema into a CM schema.

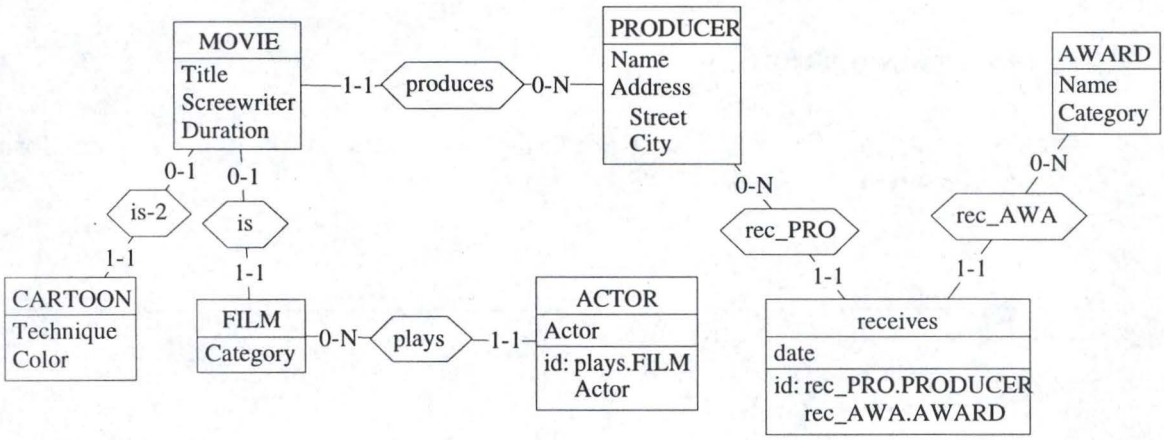
```
Let S be the current schema;
For each RT R such that ((R is n-ary) or (R has attributes)) do
        RT-to-ET(R)
For each RT R such that ((R is binary) and (R is many-to-many)) do
        RT-to-ET(R)
do
        for each attribute A such that  (A is multivalued) do
                Att-to-ET/Instance (A)
until there is no more mulitvalued attributes
For ET E such that ( E is a supertype) do
        ISA-to-RT(E)
```

Example :

The following ER schema :



is transformed into this CM schema :

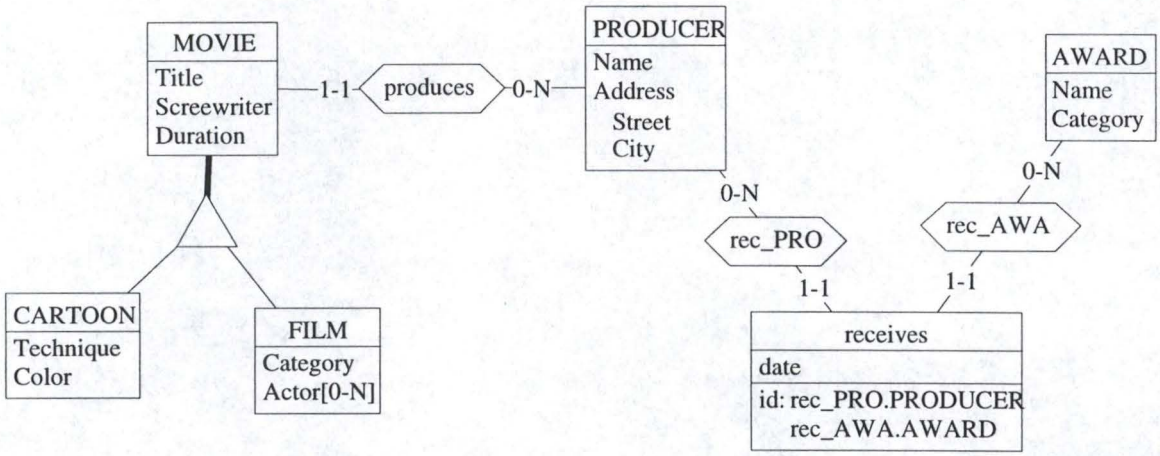


The following transformation script describes a transformation plan that transforms a ER schema into a XML model.

```
Let S be the current schema;
For each RT R such that ((R is n-ary) or (R has attributes)) do
        RT-to-ET(R)
For each RT R such that ((R is binary) and (R is many-to-many)) do
        RT-to-ET(R)
```

Example :

The ER schema given here above is transformed into the following XML model:



Appendix B : XML-QL

We give this introduction to XML-QL using examples that are based on the following DTD :

```
<!ELEMENT movie (title, producer+, actor)>
<!ATTLIST movie id ID #REQUIRED>
<!ELEMENT cartoon (producer+)>
<!ELEMENT producer (name, address)>
<!ATTLIST producer produces IDREF#REQUIRED>
<!ELEMENT actor (firstname?, name)>
<!ELEMENT title #PCDATA>
<!ELEMENT name #PCDATA>
<!ELEMENT address #PCDATA>
<!ELEMENT firstname #PCDATA>
```

XML-QL uses element patterns to match data in an XML document.

Example :

```
WHERE <movie>
          <actor>Binoche</actor>
          <producer> $p </producer>
          <Title> $t</Title>
      </movie> IN "cine.xml"
CONSTRUCT $p
```

This query matches every movie element, in the XML document cine.xml, that has at least one producer and one title and whose actor is "Binoche". For each match, it binds the variable t and p to every title and producer pair. The result is the list of all producer bound to p. Variable names are preceded by $ in order to make a distinction with the string literals (E.g : Binoche).

The construct part can be used to construct a new XML data, a data that doesn't exist in the input file.

Example :

```
WHERE <movie>
          <actor>Binoche</actor>
          <Title> $t </Title>
```

```
            <producer> $p </producer>
                 </movie> IN "cine.xml"
CONSTRUCT <result>
                 <Title> $t </Title>
                 <producer> $p </producer>
            </result>
```

This query returns both `title` and `producer` and builds a new element result that groups the two elements.

In order to group the result, we can use nested queries. The result construct contains a query.

Example :

```
WHERE <movie> $m </movie> IN "cine.xml",
            <title> $t </title>
            <actor>Binoche</actor> IN $m
CONSTRUCT <result>
                 <title>$t</title>
                 WHERE <producer>$p</producer> IN $m
                 CONSTRUCT <producer>$p</producer>
             </result>
```

This query produce a result for each `title` and contains a list of all its `producer`.

XML-QL can express joins by matching two or more elements that contain the same value.

Example :

```
WHERE <cartoon>
            <producer>
                 <name>$n</name>
                 <address>$a</address>
            </producer>
       </cartoon>
CONTENT_AS $c IN "cine.xml"
          <movie>
            <producer>
                 <name>$n</name>
                 <address>$a</address>
            </producer>
    </movie> IN "cine.xml"

   CONSTRUCT <cartoon> $c </cartoon>
```

This query gives all cartoons that have at least one producer that has also produced one movie.

To support element sharing, XML reserves an attribute of type ID, which allows a unique key to be associated with an element. An attribute of type IDREF allows an element to refer to another element with the designated key, and one of type IDREFS may refer to multiple elements. We can have joint query on these attributes.

Example :

```
WHERE <producer produces = $p>
          <name> </name> ELEMENT_AS $n
            </producer>
      <movie id=$i>
                <title>  </title> ELEMENT_AS $t
            </movie>,
CONSTRUCT <result> $n $t</result>
```

This query produces all name and title pairs by joining the producer element's IDREF attribute value with themovie element's ID attribute value.

Appendix C : Repository

Figure c.1. shows the repository'sschema. This figure is taken from [Englebert, 1998]

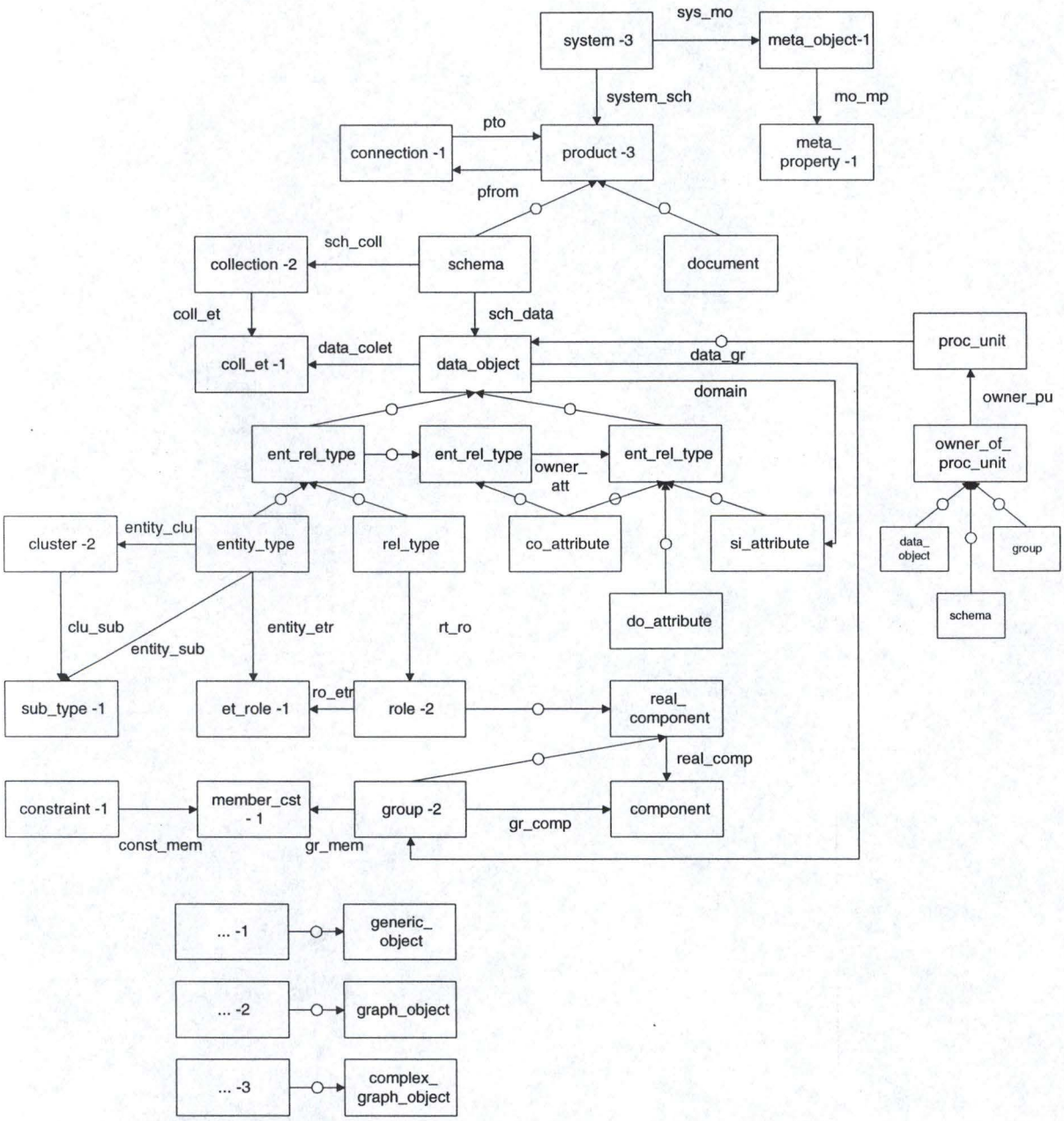


Figure c.1. : repository'sschema

Appendix D : AnalyseMap Program

```
/*****************************************************************
Programme Voyager2 qui analyse le fichier log pour en deduire les mappings
et insere ceux ci dans des metaproprietes
***************************************************************** */

string: des_ent1, des_att, des_pref,line2;
schema: sch;
file : f, map;
integer : rien,fin,cmpt2;
attribute : att;
owner_of_att : own;
entity_type : ent, ent1,ent2;
list : l_att,l_ent,l_ass, l_des_att;
cursor : c,c2;
string:nom_att;
integer:comp;

list : l_att1; // liste de liste composée de la transformation. de l'attribut dans le source et de l'attribut dans
le cible
list : l_ent1; //idem pour les entités
list : l_ass1; // idem pour les associations

/* ---------------------------------------------------------------
Fonction pour ouvrir un fichier. Retourne 1 si l'ouverture
s'est bien passée. 0 sinon
---------------------------------------------------------------*/
function integer OuvreFichier()
{ print("Nom du fichier: ");
 f:=OpenFile(read(_string),_R);
 if IsVoid(f) then {
  print("INCORRECT FILE !");
  return 0;
 }
 else {
  return 1;
 };
}

/* ---------------------------------------------------------------
Fonction pour passer des lignes dans un fichier
---------------------------------------------------------------*/

function integer SkipLine (string:s)

string : stemp;
integer : i,truc;
{
truc :=-1;
while eof(f)=0 and truc<0 do
```

```
        {
        stemp := readf (f,_string);
        truc := StrFindSubStr (stemp,0,s);
        i := i+1;
        }
return truc;
}

/* ----------------------------------------------------------------------
procédure cree_liste_ini
----------------------------------------------------------------------*/

procedure cree_liste_ini ()

{
if not(IsVoid (sch)) then
        {
        InitListeEnt ();
        InitListeAss ();
        }
}

/* ----------------------------------------------------
procédure pour remplir la liste des entités l_ent1
----------------------------------------------------*/

procedure InitListeEnt ()

entity_type : ent;
data_object : dat;
list : l;

{
for dat in DATA_OBJECT[dat]{ @SCH_DATA:[sch]with GetType(dat)=ENTITY_TYPE} do
        {
        ent := dat;
        l := LEnt(ent);
        AddFirst(l_ent1,l);
        InitListeAttCO(ent);
        InitListeAttSI(ent);
        }
}

/* ----------------------------------------------------
fonction qui donne une liste pour les entités
---------------------------------------------------- */

function list LEnt( entity_type:ent)

list :l;

{
AddFirst(l,"");
AddLast(l,ent.name);
AddLast(l,ent.name);
return l;
}

/* ----------------------------------------------
```

```
procédure pour mettre des attributs simples dans l_att1
------------------------------------------------------------ */

procedure InitListeAttSI (entity_type:ent)

attribute:att;
list:l;

{
for att in ATTRIBUTE[att]{ @OWNER_ATT:[ent]with GetType(att)=SI_ATTRIBUTE} do
        {
        l:= LAttSI(ent.name,att);
        AddFirst(l_att1,l);
        }
}

/* ------------------------------------------------------------
procédure pour mettre des attributs composés dans l_att1
------------------------------------------------------------ */

procedure InitListeAttCO (entity_type:ent)

attribute:att;
list:l;

{
for att in ATTRIBUTE[att]{ @OWNER_ATT:[ent] with GetType(att)=CO_ATTRIBUTE} do
        {
        LAttCO(ent.name,att);
        }
}

/* ------------------------------------------------------------
 procédure récursive pour la liste des attributs composés
 ------------------------------------------------------------ */

procedure LAttCO(string:nom, attribute:att)

list:l;
co_attribute:coatt;

{
if GetType(att)=SI_ATTRIBUTE
        then
        {
        AddFirst(l,"");
        nom := StrConcat(nom,StrConcat(".",att.name));
        AddLast(l,nom);
        AddLast(l,nom);
        AddFirst(l_att1,l);
        }
        else
        {
        nom := StrConcat(nom,StrConcat(".",att.name));
        coatt := att;
        AddFirst(l,"");
        AddLast(l,nom);
        AddLast(l,nom);
        AddFirst(l_att1,l);
```

```
        for att in ATTRIBUTE[att]{ @OWNER_ATT:[coatt]} do
                {
                LAttCO(nom,att);
                }
        }
}

/* ----------------------------------------------
fonction pour donner la liste des attributs simples
---------------------------------------------- */

function list LAttSI(string:nom, attribute:att)

list :l;

{
AddFirst(l,"");
AddLast(l,StrConcat(nom,StrConcat(".",att.name)));
AddLast(l,StrConcat(nom,StrConcat(".",att.name)));
return l;
}

/* ----------------------------------------------
procedure pour remplir la liste des associations l_ass1
---------------------------------------------- */

procedure InitListeAss ()

rel_type:rel;
data_object:dat;
list:l;

{
for dat in DATA_OBJECT[dat]{ @SCH_DATA:[sch]with GetType(dat)=REL_TYPE} do
        {
        rel := dat;
        l:=LRel(rel);
        AddFirst(l_ass1,l);
        }
}

/* ----------------------------------------------
fonction qui donne une liste pour les relations
---------------------------------------------- */

function list LRel(rel_type:rel)

list :l;

{
AddFirst(l,"");
AddLast(l,rel.name);
AddLast(l,rel.name);
return l;
}

/* ----------------------------------------------
procedure pour afficher le contenu d'une liste de liste
---------------------------------------------- */
```

```
/*
procedure Affiche (list : l)

list:li;
string:s;
integer:i,j;
cursor: c1,c2;

{
attach c1 to l;
for i in [1..Length(l)] do
        {
        li := get(c1);
        attach c2 to li;
        for j in [1..Length(li)] do
                {
                s:= get(c2);
                print([s,"\t"]);
                c2>>;
                }
        c1>>;
        print("\n");
        }
}

*/

/* ----------------------------------------------------------------------
Procédure pour analyser la désaggrégation
----------------------------------------------------------------------*/

procedure desag ()

string : line;
char : trash;
integer : cmpt, cmpt2,trash2;
list : fill;
cursor : c,c2;

{
line := readf (f,_string);
line := readf (f,_string);
SetParser (line);
SkipUntil (".");
trash := GetChar();
trash := GetChar();
des_ent1 := GetTokenUntil ("\"");
SkipUntil (".");
trash := GetChar();
trash := GetChar();
des_att := GetTokenUntil ("\"");
line := readf (f,_string);
if StrFindSubStr (line,0,"PRE") >-1 then
        {
        SetParser (line);
        SkipUntil ("\"");
        trash := GetChar();
        des_pref := GetTokenUntil ("\"");
```

```
        }
else
        {
        des_pref := "";
        }
cmpt :=1;
line := readf (f,_string);
while StrFindSubStr (line,0,"DEL COA") <0
do
{
        if StrFindSubStr (line,0,"NAM") > -1
        then
        {
        line :=readf(f,_string);
        trash2 :=SkipLine ("OID");
        line :=readf(f,_string);
        SetParser (line);
        if StrFindSubStr (line,0,"_")>-1 then
                {
                SkipUntil ("_");
                trash:=GetChar();
                AddLast (l_des_att,GetTokenUntil ("\""));
                }
        else
                {
                SkipUntil ("\"");
                trash:=GetChar();
                AddLast (l_des_att,GetTokenUntil ("\""));
                }
        line := readf (f,_string);
        cmpt:=cmpt+1;
        }
        else
        {
        line := readf (f,_string);
        }
}

cmpt2:=1;
attach c to l_des_att;
for cmpt2 in [1..cmpt-1] do
{
fill:=[];
AddLast (fill,"disaggregate");
AddLast (fill,des_ent1+"."+des_att+"."+get(c));
AddLast (fill,des_ent1+"."+des_pref+get(c));
c>>;
AddLast (l_att,fill);
}

}

/*-----------------------------------------------------------------
Procédure pour analyser le split_merge de deux entités
-----------------------------------------------------------------*/

procedure split_merge()

string : line,e1,e2,temp,rel;
```

```
integer : trash,truc,cmpt,cmpt2;
char : bin;
list : l_a;
cursor : c,c2;

{
trash := SkipLine ("BEG");
line := readf (f,_string);
SetParser (line);
SkipUntil ("\"");
bin :=GetChar();
e1 := GetTokenUntil ("\"");
trash := SkipLine ("MOD ENT");
trash := SkipLine ("MOD ENT");
trash := SkipLine ("OID");
line := readf (f,_string);
SetParser (line);
SkipUntil ("\"");
bin :=GetChar();
e2 := GetTokenUntil ("\"");

trash := SkipLine ("MOD REL");
trash := SkipLine ("OID");
line := readf (f,_string);
SetParser (line);
SkipUntil ("\"");
bin :=GetChar();
rel := GetTokenUntil ("\"");

trash := SkipLine ("MOD SIA");
line := readf (f,_string);
truc:=0;
cmpt:=0;
while truc>-1
do
        {
        trash:=SkipLine ("END");
        trash:=SkipLine ("OID");
        line := readf (f,_string);
        SetParser (line);
        SkipUntil ("\"");
        bin :=GetChar();
        temp := GetTokenUntil ("\"");
        line := readf (f,_string);
        if StrFindSubStr (line,0,"OWN")<0
        then
                {
                l_a := [];
                AddLast (l_a,"split-merge");
                AddLast (l_a,e1+"["+e2+"."+rel+"]"+"."+ temp);
                AddLast (l_a,e2+"."+temp);
                AddLast (l_att,l_a);
                line:= readf (f,_string);
                if StrFindSubStr (line,0,"END") >0 then {truc:=-1;}
                }
        else
                {
                truc:=SkipLine ("END");
                line:= readf (f,_string);
```

```
                if StrFindSubStr (line,0,"END") >0 then {truc:=-1;}
                }
        }
}

/*-----------------------------------------------------------------
Procédure pour analyser la transformation d'une entité en attribut
-----------------------------------------------------------------*/

procedure eta()

integer : bin;
char : trash;
string : line, e1, e2, att1, att2, rel;
list : l_a;

{
bin := SkipLine ("BEG");
line := readf (f,_string);
SetParser (line);
SkipUntil ("\"");
trash := GetChar ();
e1 := GetTokenUntil ("\"");
bin := SkipLine ("POY");
while StrFindSubStr (line,0,"*") =-1 do
{
line := readf (f,_string);
}
if StrFindSubStr (line,0,"CRE COA") >-1
        then {

            }
        else {
                bin := SkipLine ("OID");
                line := readf (f,_string);
                SetParser (line);
                SkipUntil ("\"");
                trash := GetChar ();
                att1 := GetTokenUntil ("\"");
                bin := SkipLine ("OWN");
                line := readf (f,_string);
                SetParser (line);
                SkipUntil (".");
                trash := GetChar ();
                trash := GetChar ();
                e2 := GetTokenUntil ("\"");
                bin := SkipLine ("OID");
                line := readf (f,_string);
                SetParser (line);
                SkipUntil ("\"");
                trash := GetChar ();
                att2 := GetTokenUntil ("\"");
                bin := SkipLine ("DEL REL");
                bin := SkipLine ("OID");
                line := readf (f,_string);
                SetParser (line);
                SkipUntil ("\"");
                trash := GetChar ();
                rel := GetTokenUntil ("\"");
```

```
                l_a := [];
                AddLast (l_a,"et-to-att");
                AddLast (l_a,e1+"["+e2+"."+rel+"]"+"."+att1);
                AddLast (l_a,e2+"."+att2);
                AddLast (l_att,l_a);

            }

}

/*-----------------------------------------------------------------
Procedure pour analyser la transformation d'un attribut en entité
-----------------------------------------------------------------*/

procedure ate()

integer : bin,bool;
char : trash;
string : line, coatt, e1, e2, rel, att;
list : l_a;

{
line := readf (f,_string);
line := readf (f,_string);
SetParser (line);
SkipUntil("\"");
trash := GetChar ();
coatt := GetTokenUntil ("\"");
line := readf (f,_string);
SetParser (line);
SkipUntil(".");
trash := GetChar ();
trash := GetChar ();
e1 := GetTokenUntil ("\"");
bin := SkipLine ("CRE ENT");
bin := SkipLine ("OID");
line := readf (f,_string);
SetParser (line);
SkipUntil("\"");
trash := GetChar ();
e2 := GetTokenUntil ("\"");
bin := SkipLine ("OID");
line := readf (f,_string);
SetParser (line);
SkipUntil("\"");
trash := GetChar ();
rel := GetTokenUntil ("\"");
bin := SkipLine ("desag");
bin := SkipLine ("MOD SIA");
bool := 0;
while bool>-1 do
        {
        bin := SkipLine ("OID");
        bin := SkipLine ("OID");
        line := readf (f,_string);
        SetParser (line);
        SkipUntil("\"");
        trash := GetChar ();
        att := GetTokenUntil ("\"");
```

```
        l_a := [];
        AddLast (l_a,"att-to-et\\inst");
        AddLast (l_a,e1+"."+coatt+"."+att);
        AddLast (l_a,e2+"["+e1+"."+rel+"]"+"."+ att);
        AddLast (l_att,l_a);
        bin := SkipLine ("END");
        line := readf (f,_string);
        SetParser (line);
        bool := StrFindSubStr (line,0,"MOD SIA");
        }
}

/*-----------------------------------------------------------------
Procédure pour analyser le renommage d'une entité
-----------------------------------------------------------------*/

procedure renam()

integer : bin,bool;
char : trash;
string : line, e1, e2;
list : l_a;

{
bin := SkipLine ("OID");
line := readf (f,_string);
SetParser (line);
SkipUntil("\"");
trash := GetChar ();
e1 := GetTokenUntil ("\"");
bin := SkipLine ("OID");
line := readf (f,_string);
SetParser (line);
SkipUntil("\"");
trash := GetChar ();
e2 := GetTokenUntil ("\"");
if StrCmp (e1,e2) = 0 then { }
                        else {
                                l_a := [];
                                AddLast (l_a,"rename");
                                AddLast (l_a,e1);
                                print ("FILM :"+e1);
                                AddLast (l_a,e2);
                                AddLast (l_ent,l_a);}
}

/* ---------------------------------------------------------
procedure d'intégration des deux listes
--------------------------------------------------------- */

procedure integre()
{

if not(IsVoid (sch)) then
        {
        l_att1 := Compare(l_att,l_att1);
        l_ent1 := Compare(l_ent,l_ent1);
        //l_ass1 := Compare(l_ass,l_ass1);
        }
```

```
}

/* ----------------------------------------
procédure pour comparer deux listes et remplacer
---------------------------------------- */

function list Compare (list : l, list : linit)

cursor : c1,c2;
cursor : ctrouv;
list : ltemp;
integer : i;
integer : poscro, poscro2;
string : s3;
string : temp;
list : ltrouv;

{
attach c1 to l;
for i in [1..Length(l)] do
        {
        ltemp := get(c1);
        s3 := GetLast(ltemp);
        poscro := StrFindSubStr(s3,0,"[");
        if poscro <> -1
                then
                {
                poscro2 := StrFindSubStr (s3,0,"]");
                temp := StrGetSubStr(s3,0,poscro)+StrGetSubStr(s3,poscro2+1,StrLength(s3)-poscro2-
1);
                ctrouv := rechercher(temp,linit);
                }
                else
                {
                ctrouv :=rechercher(s3,linit);
                }
        ltrouv := get(ctrouv);
        ltrouv := ltemp;
        AddFirst(linit,ltrouv);
        kill(ctrouv);
        c1>>;
        }
return linit;
}

/* ----------------------------------------------------
Fonction qui recherche la présence d'un string dans une liste
et renvoie le cursor indiquant sa position le cas échéant.
---------------------------------------------------- */

function cursor rechercher(string : s, list : l)

cursor : c;
integer: trouve;
integer : i;
string : s1;

{
```

```
attach c to l;
i := 1;
trouve := 0;
while trouve = 0 and i <= Length(l) do
        {
        s1 := GetLast(get(c));
        if StrCmp(s,s1)=0
                then
                {
                trouve := 1;
                return c;
                }
                else
                {
                c>>;
                i := i+1;
                }
        }
}

/* ----------------------------------------------------
procédure insert_map
---------------------------------------------------- */

procedure insert_map ()
{

if not(IsVoid (sch)) then
        {
        CreerMetaProp ();
        EMEntite ();
        EMRelation ();
        EMAttribut ();
        }

}

/* ----------------------------------------------
procédure pour créer les méta-propriétés
---------------------------------------------- */

procedure CreerMetaProp()

meta_object:mo;
meta_property:mp;
{
mo := GetFirst(META_OBJECT[mo]{mo.type=ENTITY_TYPE});
mp:= create(META_PROPERTY, name:"corresp",type:VARCHAR_ATT,@MO_MP:mo);
mo := GetFirst(META_OBJECT[mo]{mo.type=REL_TYPE});
mp:= create(META_PROPERTY, name:"corresp",type:VARCHAR_ATT,@MO_MP:mo);
mo := GetFirst(META_OBJECT[mo]{mo.type=SI_ATTRIBUTE});
mp:= create(META_PROPERTY, name:"corresp",type:VARCHAR_ATT,@MO_MP:mo);
mo := GetFirst(META_OBJECT[mo]{mo.type=CO_ATTRIBUTE});
mp:= create(META_PROPERTY, name:"corresp",type:VARCHAR_ATT,@MO_MP:mo);
}

/* ----------------------------------------
procédure pour écrire dans les méta-propriétés des entités
---------------------------------------- */
```

```
procedure EMEntite ()

cursor:c1,c2;
list:l;
string:s1,s2;
integer:i;
entity_type : ent;
data_object:dat;
string:t;

{
attach c1 to l_ent1;

for i in [1..Length(l_ent1)] do
        {
        l:= get(c1);
        attach c2 to l;
        t:=get(c2);
        c2>>;
        s1:= get(c2);
        c2>>;
        s2:= get(c2);
        for dat in DATA_OBJECT[dat] {@SCH_DATA:[sch] with GetType(dat)=ENTITY_TYPE } do
                {
                ent:=dat;
                if StrCmp(ent.name,s2)=0 then
                        {
                        ent."corresp" := mprop(t,s1);
                        }
                }
        c1>>;
        }
}

/* ---------------------------------------------------------
procédure pour écrire la méta propriété des relations
--------------------------------------------------------- */

procedure EMRelation ()

cursor:c1,c2;
list:l;
string:s1,s2;
integer:i;
rel_type : rel;
data_object:dat;
string:t;

{
attach c1 to l_ass1;

for i in [1..Length(l_ass1)] do
        {
        l:= get(c1);
        attach c2 to l;
        t:=get(c2);
        c2>>;
        s1:= get(c2);
```

```
        c2>>;
        s2:= get(c2);
        for dat in DATA_OBJECT[dat] { @SCH_DATA:[sch] with GetType(dat)=REL_TYPE } do
                {
                rel:=dat;
                if StrCmp(rel.name,s2)=0 then
                        {
                        rel."corresp":= mprop(t,s1);
                        }
                }
        c1>>;
        }
}

/* ----------------------------------------------------
 fonction pour donner la liste des TE du schéma
 ---------------------------------------------------- */

function list ListTE (schema:sch)

data_object : dat;
entity_type:ent;
list :l;

{

for dat in  DATA_OBJECT[dat] { @SCH_DATA:[sch] with GetType(dat)=ENTITY_TYPE } do
        {
        ent:= dat;
        AddFirst(l,ent);
        }
return l;
}

/* ----------------------------------------------------
procedure pour écrire la méta-propriété des attributs
---------------------------------------------------- */

procedure EMAttribut ()

cursor:c1,c2;
list:l;
string:s1,s2,s3;
integer:i;
entity_type : ent;
data_object:dat;
attribute:att;
string:nom_ent;
string:temp;

{
attach c1 to l_att1;
for i in [1..Length(l_att1)] do
        {
        l:=get(c1);
        attach c2 to l;
        s1:= get(c2); // type de transformation
        c2>>;
        s2:= get(c2); // attribut dans le schema de base
```

```
        c2>>;
        s3:= get(c2); // attribut dans le schema dérivé
        nom_ent := trouveent(s3);
        ent := GetFirst(ENTITY_TYPE[ent]{StrCmp(ent.name,nom_ent)=0 and
IsNoVoid(member(ListTE(sch),ent))});
        trouveatt(s3);
        if comp=0
                then
                {
                EcrireAttComp(nom_att,ent,s2,s1);
                }
                else
                {
                att := GetFirst(ATTRIBUTE[att]{@OWNER_ATT:[ent] with
StrCmp(att.name,nom_att)=0 });
                att."corresp" := mprop(s1,s2);
                }
        c1>>;
        }

}

/* ----------------------------------------------------
 fonction pour trouver le nom de l'entité dans s3
 ---------------------------------------------------- */

function string trouveent(string :s)

integer: pospoint;
integer: poscro;

{

pospoint := StrFindSubStr (s,0,".");
poscro := StrFindSubStr (s,0,"[");
if poscro <>-1
        then
        {
        return StrGetSubStr(s,0,poscro);
        }
        else
        {
        return StrGetSubStr(s,0,pospoint);
        }
}

/* ----------------------------------------------------
 fonction pour trouver le nom de l'attribut dans s3
 ---------------------------------------------------- */
procedure  trouveatt(string :s)

integer: pospoint;
integer: poscro;

{
pospoint := StrFindSubStr (s,0,".");
poscro := StrFindSubStr (s,0,"]");
if poscro <>-1
        then
```

```
        {
        nom_att:= StrGetSubStr(s,poscro+2,StrLength(s)-poscro-2);
        }
        else
        {
        nom_att:= StrGetSubStr(s,pospoint+1,StrLength(s)-pospoint-1);
        }
pospoint := StrFindSubStr (nom_att,0,".");
if pospoint = -1
        then
        {
        comp := 1;
        }
        else
        {
        comp :=0;
        }
}

/* ----------------------------------------------
procédure pour écrire la méta_propriété des attributs composés
---------------------------------------------- */

procedure EcrireAttComp (string:s, entity_type:ent, string:s2, string:s1)

integer:pospoint;
integer:pospoint2;
string:a,suite,b,reste;
attribute:att,att2;
co_attribute:coatt;

{
pospoint := StrFindSubStr (s,0,".");
a := StrGetSubStr(s,0,pospoint);
suite := StrGetSubStr(s,pospoint+1,StrLength(s)-pospoint-1);
pospoint2 := StrFindSubStr (suite,0,".");
if pospoint2 = -1
        then
        {
        att := GetFirst (ATTRIBUTE[att]{@OWNER_ATT:[ent] with StrCmp(att.name,a)=0 and
                                                        GetType(att)=CO_ATTRIBUTE})
                                                ;

        coatt := att;
        att := GetFirst(ATTRIBUTE[att]{@OWNER_ATT:[coatt] with StrCmp(att.name,suite)=0});
        att."corresp":=mprop(s1,s2);
        }
        else
        {
        att := GetFirst (ATTRIBUTE[att]{@OWNER_ATT:[ent] with StrCmp(att.name,a)=0 and
                                                        GetType(att)=CO_ATTRIBUTE})
                                                ;

        coatt := att;
        b := StrGetSubStr(suite,0,pospoint2);
        att2 := GetFirst (ATTRIBUTE[att]{@OWNER_ATT:[coatt] with StrCmp(att.name,b)=0});
        if GetType(att2)=CO_ATTRIBUTE
                then
                {
```

```
                reste:=StrGetSubStr(suite,pospoint2+1,StrLength(suite)-1-pospoint2);
                coatt:=att2;
                EcrireComp(coatt,reste,s2,s1);
                }
                else
                {
                att2."corresp" := mprop(s1,s2);
                }
        }
}

/* ----------------------------------------------------
pour ecrire les attributs des attributs composés---- recursif
---------------------------------------------------- */

procedure EcrireComp (co_attribute:coatt, string:s, string:s2, string:s1)

integer:pospoint;
attribute:att;
string:nom;
{
pospoint := StrFindSubStr (s,0,".");
if pospoint = -1
        then
        {
        att := GetFirst (ATTRIBUTE[att]{ @OWNER_ATT:[coatt] with StrCmp(att.name,s)=0});
        att."corresp":=mprop(s1,s2);
        }
        else
        {
        nom := StrGetSubStr(s,0,pospoint);
        att := GetFirst (ATTRIBUTE[att]{ @OWNER_ATT:[coatt] with StrCmp(att.name,nom)=0 });
        nom := StrGetSubStr(s,pospoint+1,StrLength(s)-pospoint-1);
        EcrireComp(att,nom,s2,s1);

        }
}

/* ----------------------------------------------------
fonction pour donner le bon format de la metapropriete
---------------------------------------------------- */

function string mprop(string:s1, string: s2)

integer:pospoint;
string:nom;
string:att;
string:rel;
integer:poscro;
integer:poscro2;
{
switch (s1)
        {
        case "rename" :
                pospoint := StrFindSubStr(s2,0,".");
                nom := StrGetSubStr(s2,pospoint+1,StrLength(s2)-pospoint-1);
                print ("FILM2 :"+nom);
                return s1+"("+nom+")";
        case "att-to-et\\inst" :
```

```
                pospoint := StrFindSubStr(s2,0,".");
                nom := StrGetSubStr(s2,0,pospoint);
                att:= StrGetSubStr(s2,pospoint+1,StrLength(s2)-pospoint-1);
                return s1+"("+nom+","+att+")";
        case "disaggregate" :
                pospoint := StrFindSubStr(s2,0,".");
                nom := StrGetSubStr(s2,0,pospoint);
                att:= StrGetSubStr(s2,pospoint+1,StrLength(s2)-pospoint-1);
                return s1+"("+nom+","+att+")";
        case "split-merge" :
                poscro := StrFindSubStr(s2,0,"[");
                nom := StrGetSubStr(s2,0,poscro);
                pospoint := StrFindSubStr(s2,0,".");
                poscro2 := StrFindSubStr(s2,pospoint,"]");
                rel := StrGetSubStr(s2,pospoint+1,poscro2-1);
                att := StrGetSubStr(s2,pospoint+poscro2+2,StrLength(s2)-pospoint-poscro2-2);
                return s1+"("+nom+","+att+","+rel+")";
        case "et-to-att" :
                poscro := StrFindSubStr(s2,0,"[");
                nom := StrGetSubStr(s2,0,poscro);
                pospoint := StrFindSubStr(s2,0,".");
                poscro2 := StrFindSubStr(s2,pospoint,"]");
                rel := StrGetSubStr(s2,pospoint+1,poscro2-1);
                att := StrGetSubStr(s2,pospoint+poscro2+2,StrLength(s2)-pospoint-poscro2-2);
                return s1+"("+nom+","+att+","+rel+")";
        otherwise :
                return s2;
        }

}

/*-----------------------------------------------------------------------------
Programme principal
-----------------------------------------------------------------------------*/

begin
if OuvreFichier () then
{
sch := GetCurrentSchema();
if not (IsVoid (sch)) then
        {
        cree_liste_ini();
        fin :=0;
        line2 := readf (f,_string);
        while fin =0 do
        {
        if StrFindSubStr (line2, 0,"*")>-1 then
                {
                if StrFindSubStr (line2, 0,"*TRF desagre_att")>-1 then
                        { desag ();
                        line2 := readf (f,_string);}
                                                        else
                {
                if StrFindSubStr (line2, 0,"*TRF att_to_et_inst")>-1 then
                        { ate ();
                        line2 := readf (f,_string);}
                                                        else
                {
                if StrFindSubStr (line2, 0,"*MOD ENT")>-1 then
```

```
                    { renam ();
                    line2 := readf (f,_string);}
                                                    else
                {
                if StrFindSubStr (line2, 0,"*TRF split_merge")>-1 then
                    { split_merge ();
                    line2 := readf (f,_string);}
                                                    else
                {
                if StrFindSubStr (line2, 0,"*TRF et_to_att")>-1 then
                    { eta ();
                    line2 := readf (f,_string);}
                                                    else
                {
                if StrFindSubStr (line2, 0,"*POT \"end-file\"")>-1 then
                    { fin:=1;
                    }
                                                    else
                {line2 := readf (f,_string);}}}}}}}
                                    else
                {line2 := readf (f,_string);}

            }
            integre();
            insert_map();
            }

CloseFile (f);


}
end
```

Appendix E : ConvertDB Program

```
/* ****************************************
 DB-to-DB converter generator
**************************************** */

schema: sch;
file : f;
string : select_clause, from_clause, where_clause;
integer: renomme;
string : ancien;
integer: philou;

list: l_first; // liste des entités à inserer telles quelles, sans tenir compte des liens.
list : l_follow; // liste de liste dont chaque element est une entite et le nom du resultset de la liste
correspondance.
list : l_link; // liste qui represente les liens many-to-many à inserer, c'est une liste de liste dont les éléments
sont les noms des entités et le resultset.
list: correspondance; // liste de liste dont chaque element est le nom du resultset (format : rsent1ent2), les
deux entites et les attributs identifiants.

/* ------------------------
Fonction pour créer un fichier dans lequel sera écrit le DTD.
Retourne 1 si la création s'est bien passée, 0 sinon.
------------------------*/

function integer OuvreFichier()
{
 f:=OpenFile("dbtodbconverter.java",_W);
 if IsVoid(f) then {
  print("INCORRECT FILE !");
  return 0;
 }
 else {
  return 1;
 }
}

/* ----------------------------------------
procedure qui donne la liste triee des entités à traiter
---------------------------------------- */

procedure initlistes ()

data_object:dat;
entity_type:ent,ent1;
role:r;
et_role:er;
integer:ok;
rel_type:rel;
string : rsname;
```

```
cursor:c,c2;
integer:i;
list:temp;

{
for dat in DATA_OBJECT[dat] {@SCH_DATA:[sch] with GetType(dat)=ENTITY_TYPE } do
        {
        ok := 1;
        ent := dat;
        for r in ROLE[r]{RO_ETR: ET_ROLE[er] {@ENTITY_ETR :[ent]}}
                do
                {
                if r.min_con = 1 and r.max_con =1 then { ok := 0; }
                }
        if ok = 1
                then
                {
                AddFirst (l_first,ent.name);
                }
                else
                {
                listefollow(ent);
                }
        }

for dat in DATA_OBJECT[dat]{@SCH_DATA:[sch] with GetType(dat)=REL_TYPE} do
        {
        rel:=dat;
        if Length(ROLE[r]{@RT_RO:[rel] with r.min_con =0 and r.max_con <>1}) = 2
                then
                {
                r:= GetFirst(ROLE[r]{@RT_RO:[rel]});
                ent:=GetFirst(ENTITY_TYPE[ent]{ENTITY_ETR: ET_ROLE[er]{@RO_ETR:[r]}});
                ent1 := relation(ent,rel);
                rsname:="rs"+ent.name+ent1.name;
                rsname:=changeRSname(rsname,ent.name,ent1.name);
                AddFirst(l_link,[ent.name,ent1.name,rsname]);
                }
        }
}

/* ----------------------------------------------
procedure pour remplir la liste qui contient les entités qu'il faut inserer en respectant les liens
---------------------------------------------- */

procedure listefollow (entity_type : ent)

rel_type:rel;
role:r;
et_role:er;
entity_type:ent1;
string:rsname;

{
for r in ROLE[r]{RO_ETR: ET_ROLE[er] {@ENTITY_ETR :[ent]}}
                do
                {
                if r.min_con = 1 and r.max_con =1
```

```
                        then
                        {
                        rel := GetFirst(REL_TYPE[rel]{RT_RO:[r]});
                        ent1 := relation(ent,rel);
                        rsname := "rs"+ent.name+ent1.name;
                        rsname := changeRSname(rsname,ent.name,ent1.name);
                        AddFirst(l_follow,[ent.name,rsname]);
                        }
                }
}

/* ----------------------------------------
procedure d'affichage des listes globales
appel dans initlistes
---------------------------------------- */

procedure affiche ()

cursor: c,c1;
cursor: c2;
integer:i,j;
list:temp;

{
attach c to l_first;
for i in [1..Length(l_first)] do
        {
        print(get(c)+"\n");
        c>>;
        }
print("\n");

attach c1 to l_follow;
print("\n");
for i in [1..Length(l_follow)] do
        {
        temp:= get(c1);
        attach c2 to temp;
        print(get(c2)+"\t");
        c2>>;
        print(get(c2)+"\n");
        c1>>;
        }
print("\n");
attach c1 to l_link;
for i in [1..Length(l_link)] do
        {
        temp:=get(c1);
        attach c2 to temp;
        print(get(c2)+"\t");
        c2>>;
        print(get(c2)+"\t");
        c2>>;
        print(get(c2)+"\t");
        c1>>;
        }

attach c to correspondance;
print(Length(correspondance));
```

```
for i in [1..Length(correspondance)] do
        {
        temp:=get(c);
        attach c2 to temp;
        print(get(c2)+"\n");
        c>>;
        }
}

/* ----------------------------------------------------------
fonction qui verifie qui change le nom du resultset de facon à ce qu'il soit bien ecrit
---------------------------------------------------------- */

function string changeRSname(string:rsname, string:ent, string:ent1)

list : ltemp;

{
ltemp := recherche(rsname);
if Length(ltemp)=0
        then
        {
        return "rs"+ent1+ent;
        }
        else
        {
        return rsname;
        }
}

/* ----------------------------------------------------------
fonction qui donne la liste qui correspond au resultset dans correspondance
---------------------------------------------------------- */

function list recherche (string:rsname)

cursor: c1, c2;
list: temp;
integer:i;

{

attach c1 to correspondance;
for i in [1..Length(correspondance)] do
        {
        temp := get(c1);
        attach c2 to temp;
        if StrCmp(get(c2),rsname)=0 then { return temp;}
        c1>>;
        }
}

/* ----------------------------------------
Procedure pour générer le module d'extraction
---------------------------------------- */

procedure GenExtractMod ()

{
```

```
Connect();
Extract();
extract2();
initlistes();
Disconnect();
}

/* ----------------------------------------
procedure pour initialiser les resultsets
non optimise car cree trop de resultsets
---------------------------------------- */

procedure initresultset ()

data_object:dat;
entity_type:ent,ent1;
role:r;
et_role:er;
rel_type:rel;

{

for dat in DATA_OBJECT[dat]{@SCH_DATA:[sch] with GetType(dat)=ENTITY_TYPE}do
        {
        ent:=dat;
        printf(f,"ResultSet rs"+ent.name+" = null;\n");
        }
for dat in DATA_OBJECT[dat]{@SCH_DATA:[sch] with GetType(dat)=ENTITY_TYPE}do
        {
        ent:=dat;
        for rel in REL_TYPE[rel] {RT_RO: ROLE[r] {RO_ETR: ET_ROLE[er] {@ENTITY_ETR
:[ent]}}} do
                {
                ent1 := relation(ent, rel);
                printf(f,"ResultSet rs"+ent.name+ent1.name+" = null;\n");
                }
        }

}

/* ----------------------------------
Procedure pour générer la connection au serveur local
---------------------------------- */

procedure Connect ()


string: url, login, pswd;
{
printf (f, "// Connection au serveur local\n");

printf(f,"try {\n");
printf (f,"Class.forName(\"jdbc.interDB.cql\");\n");
printf(f,"} catch (java.lang.ClassNotFoundException e) {\n");
printf(f," System.err.print (\"ClassNotFoundException: \");\n");
printf(f," System.err.println (e.getMessage());\n");
printf(f,"}\n");

print("URL du serveur local: ");
```

```
url:=read(_string);
print("\nLogin : ");
login:=read(_string);
print("\nMot de passe : ");
pswd:=read(_string);
print("\n");
initresultset();
printf (f,"try {\n");
printf (f,["Connection con = DriverManager.getConnection(\"",url,"\",\"",login,"\",\"",pswd,"\");\n\n"]);

}

/* ----------------------------------------
Procedure pour générer la connection au serveur local
---------------------------------------- */

procedure Connect2 ()

string: url, login, pswd;

{
printf (f, "// Connection au serveur local\n");

printf(f,"try {\n");
printf (f,"Class.forName(\"jdbc.interDB.cql\");\n");
printf(f,"} catch (java.lang.ClassNotFoundException e) {\n");
printf(f," System.err.print (\"ClassNotFoundException: \");\n");
printf(f," System.err.println (e.getMessage());\n");
printf(f,"}\n");

print("URL du serveur local: ");
url:=read(_string);
print("\nLogin : ");
login:=read(_string);
print("\nMot de passe : ");
pswd:=read(_string);
print("\n");
printf (f,"try {\n");
printf (f,["Connection con = DriverManager.getConnection(\"",url,"\",\"",login,"\",\"",pswd,"\");\n\n"]);

}

/* ----------------------------------------
procedure pour générer la deconnection au serveur local
---------------------------------------- */

procedure Disconnect ()

{
printf (f,"// deconnection au serveur local\n\n");
printf (f,"con.close();\n\n");
printf (f,"}\n");
printf(f,"catch(SQLException ex) {\n");
printf(f,"System.err.println(\"SQLException: \" + ex.getMessage());}\n");

}
```

```
/* ----------------------------------------
Procedure d'extraction, crée un resultSet pour chaque entité.
---------------------------------------- */

procedure Extract ()

entity_type:ent;
data_object:dat;
attribute:att,att1;
string:m;

{

printf (f,"// Extraction des données par TE\n");
printf (f,"Statement stmt = con.createStatement();\n");
for dat in DATA_OBJECT[dat] {@SCH_DATA:[sch] with GetType(dat)=ENTITY_TYPE }
do
        {
        init();
        renomme :=0;
        ancien :="";
        ent:=dat;
        m := ent."corresp";
        analysemapET(m);
        for att in ATTRIBVTE[att] {@OWNER_ATT:[ent] with GetType(att)=SI_ATTRIBUTE} do
                {
                m := att."corresp";
                analysemapATT(m,ent.name);
                }
        printf (f,"rs"+ent.name+" = stmt.ExecuteQuery");
        printf (f,"(\"SELECT "+select_clause+" FROM "+from_clause);
        if StrCmp(where_clause,"")<>0
                then
                {
                printf (f," WHERE "+where_clause+";\");\n");
                }
                else
                {
                printf (f,";\");\n");
                }
        for att in ATTRIBVTE[att] {@OWNER_ATT:[ent] with GetType(att)=CO_ATTRIBUTE} do
                {
                init();
                from_clause:=ancien;
                m :=att."corresp";
                analysemapCOATT(m, ent.name, att.name);
                printf (f,"ResultSet rs"+att.name+" = stmt.ExecuteQuery");
                printf (f,"(\"SELECT "+select_clause+" FROM "+from_clause);
                if StrCmp(where_clause,"")<>0
                then
                {
                printf (f," WHERE "+where_clause+";\");\n");
                }
                else
                {
                printf (f,";\");\n");
                }
                }
```

```
        }
}

/* procedure pour initialiser les variables globales */

procedure init()

{
select_clause:="";
from_clause:="";
where_clause:="";
}


/*-----------------------------------------------
procedure qui analyse le mapping d'un attributdecomposé

ne fonctionne que pour la transformation rename.
aucune transformation permise pour ses attributs

modifie les clauses select-from-where
----------------------------------------------- */

procedure analysemapCOATT(string:m, string:ent, string:coatt)

integer:pospar,pospoint;
string:t;
string:nom, nomCA;

{

pospar := StrFindSubStr (m,0,"(");
if pospar <>-1
        then
        {
        t := StrGetSubStr(m,0,pospar);
        }
        else
        {
        t := ("No");
        }
switch (t)
        {
        case "No" :
                pospoint := StrFindSubStr(m,0,".");
                nom:=StrGetSubStr(m,0,pospoint);
                nomCA:=StrGetSubStr(m,pospoint+1,StrLength(m)-pospoint);
                if renomme = 1 then {nom := ancien;}
                if StrCmp(select_clause,"")=0
                        then
                        {
                        select_clause := nom+"."+nomCA+" ";
                        }
                        else
                        {
                        select_clause := select_clause+" , "+nom+"."+nomCA+" ";
                        }
                if ismember(from_clause,nom)=0 and renomme = 0 then
                        {
```

```
                        if StrCmp(from_clause,"")=0
                        then
                        {
                        from_clause := nom+" ";
                        }
                        else
                        {
                        if ismember(from_clause,nom)
                                then {from_clause:= from_clause+" , "+nom+" ";}
                        }
                        }
        case "rename" :
                nom := ent;
                if renomme = 1 then {nom := ancien;}
                if StrCmp(select_clause,"")=0
                        then
                        {
                        select_clause := nom+"."+StrGetSubStr(m,pospar+1,StrLength(m)-pospar-2)+"
";
                        }
                        else
                        {
                        select_clause := select_clause+" ,
"+nom+"."+StrGetSubStr(m,pospar+1,StrLength(m)-
                                                                                pospar-
                                                                        2)+" ";
                        }
                if ismember(from_clause,nom)=0 and renomme = 0 then
                        {
                        if StrCmp(from_clause,"")=0
                        then
                        {
                        from_clause := nom+" ";
                        }
                        else
                        {
                        if ismember(from_clause,nom)
                                then {from_clause:= from_clause+" , "+nom+" ";}
                        }
                        }
        otherwise : print(t+" est une transformation non traitée");
        }

}

/* ----------------------------------------
procedure qui analyse le mapping d'un attribut
modifie les clauses select-from-where
---------------------------------------- */

procedure analysemapATT(string:m, string:ent)

integer:pospar;
string:t;
string:nom,nomA,nomCA,nomR;
integer:pospoint,posvir,posvir2;

{
```

```
pospar := StrFindSubStr (m,0,"(");
if pospar <>-1
        then
        {
        t := StrGetSubStr(m,0,pospar);
        }
        else
        {
        t := ("No");
        }
switch(t)
        {
        case "rename" :
                nom := ent;
                if renomme = 1 then {nom := ancien;}
                if StrCmp(select_clause,"")=0
                        then
                        {
                        select_clause := nom+"."+StrGetSubStr(m,pospar+1,StrLength(m)-pospar-2)+"
";
                        }
                        else
                        {
                        select_clause := select_clause+" ,
"+nom+"."+StrGetSubStr(m,pospar+1,StrLength(m)-
                                                                                pospar-
                                                                        2)+" ";
                        }
        case "att-to-et\\inst" :
                posvir := StrFindSubStr(m,0,",");
                nom := StrGetSubStr(m,pospar+1,posvir-pospar-1);
                nomA := StrGetSubStr(m,posvir+1,StrLength(m)-posvir-2);
                if renomme = 1 then {nom := ancien;}
                if ismember(from_clause,nom)=0 then
                        {
                        from_clause:= from_clause+" , "+nom+" ";
                        }
                if StrCmp(select_clause,"")=0
                        then
                        {
                        select_clause := nom+"."+nomA+" ";
                        }
                        else
                        {
                        select_clause := select_clause+" , "+nom+"."+nomA+" ";
                        }
        case "disaggregate" :
                posvir := StrFindSubStr(m,0,",");
                nom := StrGetSubStr(m,pospar+1,posvir-pospar-1);
                pospoint := StrFindSubStr(m,0,".");
                nomCA := StrGetSubStr(m,posvir+1,pospoint-posvir-1);
                nomA := StrGetSubStr(m,pospoint+1,StrLength(m)-pospoint-2);
                if renomme = 1 then {nom := ancien;}
                if ismember(from_clause,nom)=0 then
                        {
                        from_clause:= from_clause+" , "+nom+" ";
                        }
                if StrCmp(select_clause,"")=0
                        then
```

```
                {
                select_clause := nom+"."+nomCA+"."+nomA+" ";
                }
                else
                {
                select_clause := select_clause+" , "+nom+"."+nomCA+"."+nomA+" ";
                }
case "split-merge" :
        posvir := StrFindSubStr(m,0,",");
        nom := StrGetSubStr(m,pospar+1,posvir-pospar-1);
        posvir2 := StrFindSubStr(m,posvir+1,",");
        nomA := StrGetSubStr(m,posvir+1,posvir2);
        nomR := StrGetSubStr(m,posvir+posvir2+2,StrLength(m)-posvir2-posvir-3);
        if StrCmp(select_clause,"")=0
                then
                {
                select_clause := nom+"."+nomA;
                }
                else
                {
                select_clause := select_clause+" , "+nom+"."+nomA;
                }
        if ismember(from_clause,nom)=0 then
                {
                from_clause:= from_clause+" , "+nom+" ";
                }
        if renomme = 1 then {ent := ancien;}
        if StrCmp(where_clause,"")=0
                then
                {
                where_clause := nom+" "+nomR+" "+ent+" ";
                }
                else
                {
                if ismember(where_clause,nom+" "+nomR+" "+ent)=0
                 then {where_clause := where_clause+" and "+nom+" "+nomR+" "+ent+" ";}
                }
case "et-to-att" :
        posvir := StrFindSubStr(m,0,",");
        nom := StrGetSubStr(m,pospar+1,posvir-pospar-1);
        posvir2 := StrFindSubStr(m,posvir+1,",");
        nomA := StrGetSubStr(m,posvir+1,posvir2);
        nomR := StrGetSubStr(m,posvir+posvir2+2,StrLength(m)-posvir2-posvir-3);
        if StrCmp(select_clause,"")=0
                then
                {
                select_clause := nom+"."+nomA;
                }
                else
                {
                select_clause := select_clause+" , "+nom+"."+nomA;
                }
        if ismember(from_clause,nom)=0 then
                {
                from_clause:= from_clause+" , "+nom+" ";
                }
        if renomme = 1 then {ent := ancien;}
        if StrCmp(where_clause,"")=0
                then
```

```
                    {
                    where_clause := nom+" "+nomR+" "+ent+" ";
                    }
                    else
                    {
                    if ismember(where_clause,nom+" "+nomR+" "+ent)=0
                     then {where_clause := where_clause+" and "+nom+" "+nomR+" "+ent+" ";}
                    }
        case "No" :
                pospoint := StrFindSubStr(m,0,".");
                nom:=StrGetSubStr(m,0,pospoint);
                nomA:=StrGetSubStr(m,pospoint+1,StrLength(m)-pospoint);
                if renomme = 1 then {nom := ancien;}
                if StrCmp(select_clause,"")=0
                        then
                        {
                        select_clause := nom+"."+nomA;
                        }
                        else
                        {
                        select_clause := select_clause+" , "+nom+"."+nomA+" ";
                        }
                if ismember(from_clause,nom)=0 and renomme = 0 then
                        {
                        if StrCmp(from_clause,"")=0
                        then
                        {
                        from_clause := nom+" ";
                        }
                        else
                        {
                        if ismember(from_clause,nom)
                                then {from_clause:= from_clause+" , "+nom+" ";}
                        }
                        }
        otherwise : print(t+" est une transformation non traitée");
        }

}

/* ----------------------------------
procedure d'analyse d'une métapropriété d'un ET
modifie from_clause
---------------------------------- */

procedure analysemapET (string : m)

integer: pospar,pospar2,posvir;
string:t; // type de transformation


{
pospar := StrFindSubStr (m,0,"(");
if pospar <>-1
        then
        {
        t := StrGetSubStr(m,0,pospar);
        }
        else
```

```
        {
        t := ("No");
        }
switch (t)
        {
        case "rename" :
                pospar2:=StrFindSubStr(m,pospar,")");
                from_clause:= StrGetSubStr(m,pospar+1,pospar2-1)+" ";
                renomme := 1;
                ancien := StrGetSubStr(m,pospar+1,pospar2-1);
        case "No" :
                from_clause:=m+" ";
        case "att-to-et\\inst" :
                posvir:=StrFindSubStr(m,pospar,",");
                from_clause := StrGetSubStr(m,pospar+1,posvir-1)+" ";
                philou := 1;
        otherwise : print ([t," est une transformation non traitée"]);
        }

}

/* fonction pour dire si un string est inclus dans un autre string */

function integer ismember (string:s, string:s2)

{
s2 := s2+" ";
if StrFindSubStr (s,0,s2) >=0
        then
        {
        return 1;
        }
        else
        {
        return 0;
        }
}

/* ------------------------------------------------------------
procedure qui analyse le maping d'un RT
modifie where_clause
------------------------------------------------------------ */

procedure analysemapR (string : m, string : ent, string: ent1)

integer:pospar,pospar2;
string:t;

{

pospar := StrFindSubStr (m,0,"(");
if pospar <>-1
        then
        {
        t := StrGetSubStr(m,0,pospar);
        }
        else
        {
        t := ("No");
```

```
        }
switch(t)
        {
        case "rename" :
                pospar2:=StrFindSubStr(m,pospar,")");
                where_clause := ent+" "+StrGetSubStr(m,pospar+1,pospar2-1)+" "+ent1+" ";
        case "No" :
                where_clause := ent+" "+m+" "+ent1+" ";
        otherwise : print (t+" est une transformation non traitée\n");
        }
}

/*-----------------------------------------------------------
procedure qui cree un ResultSet avec les ET reliées par un RT
en ne prenant que les identifiants
----------------------------------------------------------- */

procedure extract2 ()

data_object:dat;
attribute:att,att1;
rel_type:rel;
entity_type:ent,ent1;
role:r;
et_role:er;
string:me,ma,mr;
string:from_clause2;
integer:renomme1;
string:ancien1;
string:nom, nom1;
list:l;

list:lnee;

{
for dat in DATA_OBJECT[dat] {@SCH_DATA:[sch] with GetType(dat)=ENTITY_TYPE}
        do
        {
        philou :=0;
        ent:=dat;
        if visite(l,ent.name) =0
                then
                {
                for rel in REL_TYPE[rel] {RT_RO: ROLE[r] {RO_ETR: ET_ROLE[er]
{@ENTITY_ETR :[ent]}}}
                        do
                        {
                        ent1 := relation(ent, rel);
                        if visite(l,ent1.name)=0
                                then
                                {
                                init();
                                renomme:=0;
                                ancien:="";

                                me := ent."corresp";
                                philou:=0;
                                analysemapET(me);
```

```
                    if philou <> 1
                        then
                        {
                        renomme1 := renomme;
                        ancien1 := ancien;
                        from_clause2 := from_clause;
                        att := getid(ent);
                        ma := att."corresp";
                        analysemapATT(ma,ent.name);

                        renomme:=0;
                        ancien:="";
                        me := ent1."corresp";
                        analysemapET(me);
                            if philou <> 1
                            then
                            {
                            att1 := getid(ent1);
                            ma := att1."corresp";
                            analysemapATT(ma,ent1.name);
                            mr := rel."corresp";
                            if renomme1 =1 then {nom := ancien1;} else {nom
:=
                                                                    ent.nam
                                                                e;}
                            if renomme =1 then { nom1 := ancien;} else {nom1
:=
                                                                    ent1.na
                                                                me;}
                            analysemapR(mr,nom,nom1);
                            from_clause2 := from_clause2+" , "+from_clause;
                            printf (f,"rs"+ent.name+ent1.name+" =
stmt.ExecuteQuery");
                            printf (f,"(\"SELECT "+select_clause+" FROM
                                                    "+from_clause2);
                            printf (f," WHERE "+where_clause+";\");\n");
                            lnee :=
                                ["rs"+ent.name+ent1.name,ent.name,ent1.na
                            me,att.
                                name,att1.name,rel.name];
                            AddFirst(correspondance,lnee);

                            }
                        }
                    }
                }
            AddFirst(l,ent.name);
            }
        }
}

/*-----------------------------------------------------
 fonction qui donne la deuxieme entite d'une relation
-----------------------------------------------------*/

function entity_type relation (entity_type:ent, rel_type:rel)

et_role:er;
role:r;
```

```
entity_type:ent1;

{
for ent1 in ENTITY_TYPE[ent1]{ENTITY_ETR : ET_ROLE[ er]{ @RO_ETR : ROLE[r]{ @RT_RO :
[rel]}}} do
        {
        if ent1.name <> ent.name
                then
                {
                return ent1;
                }
        }
}

/*-----------------------------------------------------
fonction qui donne l'attribut identifiant d'un ET
----------------------------------------------------- */

function attribute getid (entity_type:ent)

attribute:att;
group:gr;
real_component:rc;
component:co;
integer:typrc;

{
for gr in GROUP[gr]{ @DATA_GR:[ent] with gr.primary} do
        {
        for rc in REAL_COMPONENT[rc]{REAL_COMP:COMPONENT[co]{ @GR_COMP:[gr]}} do
                {
                if GetType(rc)=SI_ATTRIBUTE
                        then
                        {
                        att := rc;
                        return att;
                        }
          }
        }
}

/*-----------------------------------------
fonction qui dit si une entite a ete visitee
----------------------------------------- */

function integer visite (list : l, string:ent)

{

if IsVoid(member(l,ent)) then { return 0;} else {return 1;}
}

/*-----------------------------------
fonction qui donne le type d'un attribut à partir de son nom et de celui de
son entité
si type <> String ou int alors  elle retourne un string vide
----------------------------------- */

function string donnetype (string:nomatt, string:noment)
```

```
entity_type:ent;
attribute:att;
si_attribute:siatt;

{
ent:= GetFirst(ENTITY_TYPE[ent]{ent.name=noment});
att:=GetFirst(ATTRIBUTE[att]{@OWNER_ATT:[ent]with GetType(att)=SI_ATTRIBUTE
and att.name=nomatt});
siatt:=att;
switch (siatt.type)
        {
        case CHAR_ATT : return "String";
        case VARCHAR_ATT : return "String";
        case NUM_ATT : return  "int";
        otherwise : return "";
        }
}

/* ------------------------------------
Procedure pour générer le module d'insertion
------------------------------------ */

procedure GenInsertMod ()

{
Connect2();
insert();
Disconnect();
}

/*------------------------------------------
fonction qui génere le code pour insérer les données
------------------------------------------ */

procedure insert ()

integer : itof,jtof;
cursor : ctof,ctof2,ctof3,ctof4;
string : trash,tmp,ent1,ent2,ident1,ident2,relat;
list : l_visit, l_tmp,l_rech;

{
//-------------------------------------------------------
// Déclaration des variables nécessaires au programme Java

printf (f, "stmt = con.createStatement();\n");
printf (f, "String ident2 = new String (\"\");\n");
printf (f, "String ident1 = new String (\"\");\n");
printf (f, "String tmp = new String (\"\");\n");
printf (f, "boolean bool;\n");
printf (f, "int id2,id1;\n");

//-------------------------------------------------------------
// Initialisation de la liste qui reprend les entités déjà insérées

l_visit :=[];

//-------------------------------------------------------
```

```
// Boucle qui permet de créer les lignes de codes pour insérer
// les entites qui n'ont pas besoin d'être liées à d'autres.


attach ctof to l_first;
for itof in [1..Length(l_first)] do
{

AddLast(l_visit,get(ctof)); // On remplit la liste des entités déjà insérées

printf (f, get(ctof)+" "+get(ctof)+"1;\n");
printf (f, "while (rs"+get(ctof)+".next())\n\t");
printf (f,"{\n\t");
printf (f, get(ctof)+"1=("+get(ctof)+")rs"+get(ctof)+".getAll();\n\t");
printf (f, "stmt.ExecuteQuery(\"insert object "+get(ctof)+"1 into "+get(ctof)+"\");\n\t");
printf (f,"}\n\n");
printf (f,"bool = rs"+get(ctof)+".First();\n");
ctof>>;
}

//-----------------------------------------------------------
// Début de la boucle qui va permettre d'insérer les entités
// qui sont liées à d'autres par des relations one-to-many

attach ctof to l_follow;

for itof in [1..Length(l_follow)] do
{
if visite (l_visit,GetFirst(get(ctof)))=0 then //Teste si l'entité a déjà été insérée

        {
        ctof2 := ctof;
        l_tmp := [];
        trash := GetFirst(get(ctof));
        AddLast(l_visit,trash);
        AddLast(l_tmp,GetLast(get(ctof)));
        ctof2>>;

//-----------------------------------------------------------
// Début de la boucle qui recherche les entités liées à
// l'entité en cours de l_follow

        for jtof in [itof+1..Length(l_follow)] do
                {
                if StrCmp(GetFirst(get(ctof2)),trash)=0 then
                        {
                        AddLast(l_tmp,GetLast(get(ctof2)));
                        }
                ctof2>>;
                }

//-----------------------------------------------------------

        attach ctof3 to l_tmp;

//-----------------------------------------------------------
// Début de la boucle qui va analyser tous les liens de l'entité
// en cours d'insertion et qui va ensuite créer
// les lignes de code chargées d'insérer cette entité
```

```
for jtof in [1..Length(l_tmp)] do
        {

//-----------------------------------------------------------
// Teste si on analyse le premier lien ou pas

        if jtof = 1 then
                {

//-----------------------------------------------------------
// Recherche de la liste de correspondance attachée aux deux
// entités dont on cherche le lien

                l_rech:=recherche (get(ctof3));
                attach ctof4 to l_rech;
                ctof4>>;

//-----------------------------------------------------------
// Test pour voir laquelle des deux entités est du côté
// 1-1 de la relation

                if IsVoid(member(l_first,get(ctof4))) then
                        {
                        ent1 := get(ctof4); // Nom de la première entité
                        ctof4>>;
                        ent2 := get(ctof4); // Nom de la deuxième entité
                        ctof4>>;
                        ident1 := get(ctof4); // Nom du première identifiant
                        ctof4>>;
                        ident2 := get(ctof4); // Nom du deuxième identifiant
                        ctof4>>;
                        relat  := get(ctof4); // Nom de la relation
                        }
                        else
                        {
                        ent2 := get(ctof4);
                        ctof4>>;
                        ent1 := get(ctof4);
                        ctof4>>;
                        ident2 := get(ctof4);
                        ctof4>>;
                        ident1 := get(ctof4);
                        ctof4>>;
                        relat  := get(ctof4);
                        }

//-----------------------------------------------------------
// Cherche le type de l'identifiant de l'entité à insérer

                tmp := donnetype (ident1,ent1);

//-----------------------------------------------------------
// Selon que le type est String ou int...

                if StrCmp (tmp,"String") =0 then
                        {
                        printf (f, ent1+" "+ent1+"1;\n");
                        printf (f, "String "+ent1+"2;\n");
```

```
                                        printf (f, "while ("+get(ctof3)+".next())\n{\n\t");
                                        printf (f, "ident1
="+get(ctof3)+".getString("+ent1+"."+ident1+");\n\t");
                                        printf (f, "while
                                                (ident1.compareTo(rs"+ent1+".getString("+ent1+"."+ident1+
"))!=0)\ n\t{\n\t");
                                        printf (f, "bool =rs"+ent1+".next();\n\t}\n\t");
                                        printf (f, ent1+"1 =rs"+ent1+".getAll();\n\t");
                                        printf (f, "bool = rs"+ent1+".First();\n\t");

//-----------------------------------------------------------------------
// Cherche le type de l'identifiant de l'entité à laquelle l'entité à
// insérer est liée

                                        tmp := donnetype (ident2,ent2);

//-----------------------------------------------------------------------
// Selon que le type est String ou int...

                                        if StrCmp (tmp,"String") =0 then
                                                {

                                                printf (f, "ident2
                                                        ="+get(ctof3)+".getString("+ent2+"."+ident2+");\n\t
");
                                                printf (f, "while
                                                        (ident2.compareTo(rs"+ent2+".getString("+ent2+"."
+ident2 +"))!=0)\n\t{\n\t");
                                                printf (f, "bool =rs"+ent2+".next();\n\t}\n\t");
                                                printf (f, ent2+"1 =rs"+ent2+".getAll();\n\t");
                                                printf (f, "bool = rs"+ent2+".First();\n\t");
                                                }
                                                else
                                                {
                                                printf (f, "id2
="+get(ctof3)+".getInt("+ent2+"."+ident2+");\n\t");
                                                printf (f, "while
                                                        (id2!=rs"+ent2+".getInt("+ent2+"."+ident2+"))\n\t{\
n\t");
                                                printf (f, "bool =rs"+ent2+".next();\n\t}\n\t");
                                                printf (f, ent2+"1 =rs"+ent2+".getAll();\n\t");
                                                printf (f, "bool = rs"+ent2+".First();\n\t");
                                                }
                                        }
                                        else
                                        {
                                        printf (f, ent1+" "+ent1+"1;\n");
                                        printf (f, "String "+ent1+"2;\n");
                                        printf (f, "while ("+get(ctof3)+".next())\n{\n\t");
                                        printf (f, "id1 ="+get(ctof3)+".getInt("+ent1+"."+ident1+");\n\t");
                                        printf (f, "while (id1 !=
rs"+ent1+".getInt("+ent1+"."+ident1+"))\n\t{\n\t");
                                        printf (f, "bool =rs"+ent1+".next();\n\t}\n\t");
                                        printf (f, ent1+"1 =rs"+ent1+".getAll();\n\t");
                                        printf (f, "bool = rs"+ent1+".First();\n\t");

//-----------------------------------------------------------------------
// Cherche le type de l'identifiant de l'entité à laquelle l'entité à
// insérer est liée
```

```
                    tmp := donnetype (ident2,ent2);

//-----------------------------------------------------------------------
// Selon que le type est String ou int...

                    if StrCmp (tmp,"String") =0 then
                        {

                        printf (f, "ident2
                                ="+get(ctof3)+".getString("+ent2+"."+ident2+");\n\t
                        ");
                        printf (f, "while
                                (ident2.compareTo(rs"+ent2+".getString("+ent2+"."
                        +ident2  +"))!=0)\n\t{\n\t");
                        printf (f, "bool =rs"+ent2+".next();\n\t}\n\t");
                        printf (f, ent2+"1 =rs"+ent2+".getAll();\n\t");
                        printf (f, "bool = rs"+ent2+".First();\n\t");
                        }
                        else
                        {
                        printf (f, "id2
="+get(ctof3)+".getInt("+ent2+"."+ident2+");\n\t");
                        printf (f, "while
                                (id2!=rs"+ent2+".getInt("+ent2+"."+ident2+"))\n\t{\
                        n\t");
                        printf (f, "bool =rs"+ent2+".next();\n\t}\n\t");
                        printf (f, ent2+"1 =rs"+ent2+".getAll();\n\t");
                        printf (f, "bool = rs"+ent2+".First();\n\t");
                        }

                    }
                    printf (f, ent1+"2 = \"INSERT OBJECT "+ent1+"1 INTO "+ent1+"
LINKED
                        WITH "+ent2+"1 VIA "+relat+"\";\n\t");

                }

//-----------------------------------------------------------------------
// On n'est plus dans le cas du premier lien
// Idem que pour le premier lien à la différence que la ligne de code
// pour la requête CQL n'est pas la même

                else
                {
                l_rech:=recherche (get(ctof3));
                attach ctof4 to l_rech;
                ctof4>>;
                if IsVoid(member(l_first,get(ctof4))) then
                    {
                    ent1 := get(ctof4);
                    ctof4>>;
                    ent2 := get(ctof4);
                    ctof4>>;
                    ident1 := get(ctof4);
                    ctof4>>;
                    ident2 := get(ctof4);
                    ctof4>>;
                    relat  := get(ctof4);
```

```
                    }
                    else
                    {
                    ent2 := get(ctof4);
                    ctof4>>;
                    ent1 := get(ctof4);
                    ctof4>>;
                    ident2 := get(ctof4);
                    ctof4>>;
                    ident1 := get(ctof4);
                    ctof4>>;
                    relat  := get(ctof4);
                    }

          tmp := donnetype (ident1,ent1);
          if StrCmp (tmp,"String") =0 then
                    {
                    printf (f, "while
                              (ident1.compareTo("+get(ctof3)+".getString("+ent1+"."+iden
                    t1+"))!=
                              0)\n\t{\n\t");
                    printf (f, "bool ="+get(ctof3)+".next();\n\t}\n\t");
                    tmp := donnetype (ident2,ent2);
                    if StrCmp (tmp,"String") =0 then
                    {

                              printf (f, "ident2
                                        ="+get(ctof3)+".getString("+ent2+"."+ident2+");\n\t
                              ");
                              printf (f, "while
                                        (ident2.compareTo(rs"+ent2+".getString("+ent2+"."
                              +ident2  +"))!=0)\n\t{\n\t");
                              printf (f, "bool =rs"+ent2+".next();\n\t}\n\t");
                              printf (f, ent2+"1 =rs"+ent2+".getAll();\n\t");
                              printf (f, "bool = rs"+ent2+".First()\n\t");
                              }
                              else
                              {
                              printf (f, "id2
="+get(ctof3)+".getInt("+ent2+"."+ident2+");\n\t");
                              printf (f, "while
                                        (id2!=rs"+ent2+".getInt("+ent2+"."+ident2+"))\n\t{\
                              n\t");
                              printf (f, "bool =rs"+ent2+".next();\n\t}\n\t");
                              printf (f, ent2+"1 =rs"+ent2+".getAll();\n\t");
                              printf (f, "bool = rs"+ent2+".First();\n\t");
                              }
                    }
                    else
                    {
                    printf (f, "while (id1 !=
                              "+get(ctof3)+".getInt("+ent1+"."+ident1+")))!=0\n\t{\n\t");
                    printf (f, "bool ="+get(ctof3)+".next();\n\t}\n\t");
                    tmp := donnetype (ident2,ent2);
                    if StrCmp (tmp,"String") =0 then
                    {

                              printf (f, "ident2
```

```
                                        ="+get(ctof3)+".getString("+ent2+"."+ident2+");\n\t
                                ");
                                printf (f, "while
                                        (ident2.compareTo(rs"+ent2+".getString("+ent2+"."
                                +ident2  +"))!=0)\n\t{\n\t");
                                printf (f, "bool =rs"+ent2+".next();\n\t}\n\t");
                                printf (f, ent2+"1 =rs"+ent2+".getAll();\n\t");
                                printf (f, "bool = rs"+ent2+".First();\n\t");
                                }
                                else
                                {
                                printf (f, "id2
="+get(ctof3)+".getInt("+ent2+"."+ident2+");\n\t");

                                printf (f, "while
                                        (id2!=rs"+ent2+".getInt("+ent2+"."+ident2+"))\n\t{\
                                n\t");
                                printf (f, "bool =rs"+ent2+".next();\n\t}\n\t");
                                printf (f, ent2+"1 =rs"+ent2+".getAll();\n\t");
                                printf (f, "bool = rs"+ent2+".First();\n\t");
                                }

                        }
                        printf (f, ent1+"2.concat (\" AND "+ent2+"1 VIA "+relat+"\");\n\t");
                        printf (f, "bool = "+get(ctof3)+".First();\n\t");

                }

        ctof3>>;
        }
    printf (f, "stmt.executeQuery("+ent1+"2);\n}\n");
    }

ctof>>;
}

}

/* ------------------------------
procédure pour générer le convertisseur
-----------------------------------*/

procedure GenConverter ()

{
printf(f,"import java.io.*;\nimport java.lang.*;\nimport java.sql.*;\nimport java.util.Vector;\n");
printf(f,"public class dbtodbconverter\n");
printf(f,"{\n");
printf(f,"public static void main (String args[])\n");
printf(f,"\t{\n");
GenExtractMod();
GenInsertMod();
printf(f,"\t}\n");
printf(f,"}\n");
}

/* ************************
 programme principal
*************************** */
```

```
begin
if OuvreFichier () then
{
sch := GetCurrentSchema();
if not(IsVoid (sch)) then
        {
        GenConverter ();
        //print ("Ok\n");
        }
CloseFile (f);
}
end
```

Appendix F : BNF of the DTD generation

```
DTD : := '<! DOCTYPE' schema '[' element_declaration ']>'
element_declaration : := (one_ET_declaration)* (one_attribute_declaration)*
one_ET_declaration : := '<! ELEMENT' TE '(' attributes_list ')>'
                        '<! ATTLIST' TE 'id ID #REQUIRED'
                                (RT-TE reference '#' mandat-optional )*
                                (isa-supertype IDREF # REQUIRED )*
                                ('identifier CDATA #FIXED " ' identifier ' " ' ) ?
                                 'corresp CDATA #FIXED " ' mapping ' " >'
attributes_list : := one_attribute | one_attribute ',' attributes_list
one_attribute : := attribute cardinality
cardinality : := ' ' | '?' | '*' | '+'
reference ::= 'IDREF' | 'IDREFS'
mandat-optional ::= 'REQUIRED' | 'IMPLIED'
one_attribute_declaration : := atomic_attribute_declaration |
compound_attribute_declaration
atomic_attribute_declaration ::=
            '< ! ELEMENT' attribute '#PCDATA>
            < ! ATTLIST attribute 'corresp CDATA #FIXED " ' mapping ' " >'
compound_attribute_declaration ::=
            '<! ELEMENT' attribute '(' attributes_list ')>
            < ! ATTLIST' attribute 'corresp CDATA #FIXED " ' mapping ' " >'
            (one_attribute_declaration)+
```

Appendix G : DTD Program

```
/* ***********************************************
Programme de génération du DTD à partir d'un schéma conceptuel
*********************************************** */

schema: sch;
file : f;

/* ---------------------------
Fonction pour créer un fichier dans lequel sera écrit le DTD.
Retourne 1 si la création s'est bien passée, 0 sinon.
---------------------------*/

function integer OuvreFichier()
{
 f:=OpenFile(sch.name+".dtd",_W);
 if IsVoid(f) then {
  print("INCORRECT FILE !");
  return 0;
 }
 else {
  return 1;
 };
}

/* ---------------------------
procedure pour générer le DTD
---------------------------*/

procedure GenererDTD ()
{
// ecriture de l'entete
printf (f, ["<!DOCTYPE ",sch.name," [\n"]);
// ecriture de l'element schema qui est compose des TE
initschema ();
// ecriture de elements ( un par TE et attributs)
entite ();
printf (f, "]>");
}

/*---------------------------
procedure pour ecrire le schema
---------------------------*/

procedure initschema ()

data_object:dat;
entity_type:ent;
{
printf(f,"<!ELEMENT "+sch.name+"(");
```

```
//ecriture de la premiere entite
dat:=GetFirst(DATA_OBJECT[dat] {@SCH_DATA:[sch] with GetType(dat)=ENTITY_TYPE});
ent:=dat;
printf(f,ent.name);
// ecriture des autres separees par une virgule
for dat in DATA_OBJECT[dat] {@SCH_DATA:[sch] with GetType(dat)=ENTITY_TYPE and
dat.name<>ent.name}
do
        {
        ent:= dat;
        printf(f,", "+ent.name);
        }
printf(f,")\n");
}

/*-----------------------------------
 procedure pour ecrire les TE
-----------------------------------*/

procedure entite ()

entity_type: ent;
data_object: dat;

// pour chaque entite
{
for dat in DATA_OBJECT[dat] {@SCH_DATA:[sch] with GetType(dat)=ENTITY_TYPE }
do
        {
        ent:=dat;
        printf (f, ["<! ELEMENT ", ent.name]);
        //ecriture des elements qui le composent (un par attributs)
        attribut (ent);
        printf (f, ">\n");
        //ecriture des attributs de l'element dans leattlist
        attlist (ent);
        //ecriture des elements correpondant aux attributs simples
        AttSi (ent);
        //ecriture des elements correspondant aux attributs composés
        AttCo (ent);
        printf (f,"\n");
        }
}

/*-----------------------------------------
procedure pour ecrire la liste des attributs d'un TE
-----------------------------------------*/

procedure attribut (entity_type : ent)

attribute : att;
{
if IsNoVoid(_GetFirst(OWNER_ATT,ent))
        then
        {
        printf (f," (");
        for att in ATTRIBUTE[att]{@OWNER_ATT:[ent]}
                do
                {
```

```
                printf (f, [ent.name,".",att.name]);
                cardinalite(att);
                if IsNoVoid(_GetNext (OWNER_ATT, ent, att))then
                        {
                        printf (f,", ");
                        }
                }
        printf (f,")");
        }
        else
        {
        printf(f," #PCDATA");
        }
}

/*-----------------------------------------
procedure pour ecrire la cardinalité d'un attribut
-----------------------------------------*/

procedure cardinalite (attribute : att)

{
if att.min_rep = 0
                then
                {
                if att.max_rep = 1 then     {printf ( f,"?");}
                                    else {printf (f,"*");};
                }
                else
                {
                if att.max_rep <> 1 then {printf ( f,"+");};
                };
}

/*-----------------------------------------
 procedure pour ecrire un attlist d'un TE
-----------------------------------------*/

procedure attlist(entity_type : ent)

group : gr;
rel_type : rel;
role : r;
et_role : er;
entity_type:temp;
cluster:clu;
sub_type:sub;
attribute:att;

{
printf (f,["<!ATTLIST ",ent.name,"\n"]);

/* ecriture de l'identifiant primaire */

att:=getid(ent);
printf (f,["\tidentifiant CDATA #FIXED \"",att.name,"\"\n"]);

/* ecriture des relations */
```

```
for rel in REL_TYPE[rel] {RT_RO: ROLE[r] {RO_ETR: ET_ROLE[er] {@ENTITY_ETR :[ent]}}}
        do
        {
        temp:= relation(ent,rel);
        printf (f,["\t",rel.name,".",temp.name]);
        cardimax (r);
        cardimin (r);
        printf (f,"\n");
        }

/* ecriture des relations isa */

/*for sub in SUB_TYPE[sub]{@CLU_SUB : CLUSTER[clu]{@ENTITY_CLU :[ent]}}
        do
        {
        printf(f,["\tisa.",clu.name," IDREF #REQUIRED\n"]);
        }
*/
for clu in CLUSTER[clu]{CLU_SUB : SUB_TYPE[sub]{@ENTITY_SUB :[ent]}}
        do
        {
        printf (f,"\tisa");
        isa(clu);
        printf (f,[".",clu.name," IDREF #REQUIRED\n"]);
        }
/* ecriture de l'identifiant XML */

printf (f,"\tid ID #REQUIRED \n");

/* ecriture de la meta propriete corresp */

printf (f,["\tCorresp CDATA #FIXED \"",ent."corresp","\">\n"]);

}

/* ----------------------------------------
fonction pour donner la deuxieme entité d'une relation
---------------------------------------- */

function entity_type relation (entity_type:ent, rel_type:rel)

et_role:er;
role:r;
entity_type:ent1;

{
for ent1 in ENTITY_TYPE[ent1]{ENTITY_ETR : ET_ROLE[ er]{@RO_ETR : ROLE[r]{@RT_RO :
[rel]}}} do
        {
        if ent1.name <> ent.name
                then
                {
                return ent1;
                }
        }
}

/* ----------------------------------------
procedure pour ecrire la cardinalité maximale d'un role
```

```
------------------------------------------------------------ */

procedure cardimax (role: r)

{
if r.max_con <>1
        then
        {
        printf (f, " IDREFS ");
        }
        else
        {
        printf (f, " IDREF ");
        }
}

/* ------------------------------------------------------------
procedure pour ecrire la cardinalité minimale d'un role
------------------------------------------------------------ */

procedure cardimin (role : r)

{
if r.min_con = 0
        then
        {
        printf (f, "#IMPLIED");
        }
        else
        {
        printf (f, "#REQUIRED");
        }
}

/* ------------------------------------------------------------
procedure pour ecrire le type de relation isa
T (total), P (partition), D (disjoint)
------------------------------------------------------------ */

procedure isa (cluster:clu)

{
if clu.total
        then
        {
        if clu.disjoint
                then
                {
                printf(f,"P");
                }
                else
                {
                printf(f,"T");
                }
        }
        else
        {
        if clu.disjoint
                then
```

```
                {
                printf(f,"D");
                }
        }
}

/*-----------------------------------------------
procedure pour ecrire la liste des attributs d'un attribut composé
----------------------------------------------- */

procedure attributco (co_attribute : coatt, string : prefixe)

attribute:att;
{
printf (f,"( ");
for att in ATTRIBUTE[att]{ @OWNER_ATT :[coatt]}
        do
        {
        printf (f,[prefixe,".",att.name]);
        cardinalite(att);
        if IsNoVoid(_GetNext (OWNER_ATT, coatt, att))then
                        {
                        printf (f,", ");
                        }
        }
printf (f," )");
}

/*-----------------------------------------
procedure pour ecrire un attribut d'un TE
----------------------------------------*/

procedure AttSi (entity_type: ent)

attribute: att;
{
for att in ATTRIBUTE[att]{ @OWNER_ATT:[ent] with GetType(att)=SI_ATTRIBUTE }
do
        {
        printf (f,["<!ELEMENT ",ent.name,".",att.name," #PCDATA>\n"]);
        printf (f,["<!ATTLIST ",ent.name,".",att.name,"\n\t corresp CDATA #FIXED
\"",att."corresp","\">\n"]);
        }
}

/*-----------------------------------------
procedure pour ecrire les attributs composés d'un TE
----------------------------------------*/

procedure AttCo (entity_type: ent)

attribute: coatt;
string : prefixe;
{
for coatt in ATTRIBUTE[coatt] { @OWNER_ATT:[ent] with GetType(coatt)=CO_ATTRIBUTE}
do
        {
        printf (f,["<!ELEMENT ",ent.name,".",coatt.name]);
        prefixe := StrConcat(ent.name,StrConcat(".",coatt.name));
```

```
        attributco(coatt,prefixe);
        printf (f, " >\n");
        printf (f,["<!ATTLIST ",ent.name,".",coatt.name,"\n\t corresp CDATA #FIXED
\"",coatt."corresp","\">\n"]);
        AttCoCo(coatt,prefixe);
        AttSiCo(coatt,prefixe);
        }
}

/* ------------------------------
procédure pour avoir les attributs des attributs composés
------------------------------ */

procedure AttSiCo (co_attribute: coatt, string:prefixe)

attribute : att;
{
for att in ATTRIBUTE[att]{ @OWNER_ATT:[coatt]with GetType(att)=SI_ATTRIBUTE}
        do
        {
        printf (f,["<!ELEMENT ",prefixe,".",att.name," #PCDATA>\n"]);
        printf (f,["<!ATTLIST ",prefixe,".",att.name,"\n\t corresp CDATA #FIXED
\"",att."corresp","\">\n"]);
        }
}

/* ------------------------------
procédure pour avoir les attributs composés des attributs composés
------------------------------ */

procedure AttCoCo (co_attribute: coatt, string:prefixe)

attribute : att;
co_attribute : coatt1;
{
for att in ATTRIBUTE[att]{ @OWNER_ATT:[coatt]with GetType(att)=CO_ATTRIBUTE}
        do
        {
        printf (f,["<!ELEMENT ",prefixe,".",att.name]);
        coatt1:=att;
        attributco(coatt1,StrConcat(prefixe,StrConcat(".",coatt1.name)));
        printf (f, " >\n");
        printf (f,["<!ATTLIST ",prefixe,".",coatt1.name,"\n\t corresp CDATA #FIXED
\"",coatt1."corresp","\">\n"]);
        AttCoCo(coatt1,StrConcat(prefixe,StrConcat(".",coatt1.name)));
        AttSiCo(coatt1,StrConcat(prefixe,StrConcat(".",coatt1.name)));
        }
}

/*------------------------------------------
fonction qui donne l'attribut identifiant d'un ET
------------------------------------------ */

function attribute getid (entity_type:ent)

attribute:att;
group:gr;
real_component:rc;
component:co;
```

```
integer:typrc;

{
for gr in GROUP[gr]{@DATA_GR:[ent] with gr.primary} do
        {
        for rc in REAL_COMPONENT[rc]{REAL_COMP:COMPONENT[co]{@GR_COMP:[gr]}} do
                {
                if GetType(rc)=SI_ATTRIBUTE
                        then
                        {
                        att := rc;
                        return att;
                        }
          }
        }
}

/* ***********************************
 programme principal
*********************************** */

begin
sch := GetCurrentSchema();
if OuvreFichier () then
{
if not(IsVoid (sch)) then
        {
        GenererDTD ();
        print("Ok\n");
        }
CloseFile (f);
}
end
```

Appendix H : ConvertXML program

```
/* ****************************************
 DB-to-XML converter generator
**************************************** */

schema: sch;
file : f;
string : select_clause, from_clause, where_clause;
integer: renomme;
string : ancien;
integer: philou;
string : select_name; // contient la partie de la select_clause qui correspond à l'attribut
list: correspondance; // liste de liste dont chaque élement est le nom du resultset (format : rsent1ent2), les
deux entites et les attributs identifiants.


/* ------------------------
Fonction pour créer le fichier dbtoxmlconverter.java dans lequel sera écrit le code java.
Retourne 1 si la création s'est bien passée, 0 sinon.
-------------------------*/

function integer OuvreFichier()
{
 f:=OpenFile("dbtoxmlconverter.java",_W);
 if IsVoid(f) then {
   print("INCORRECT FILE !");
   return 0;
 }
 else {
   return 1;
 }
}

/* ----------------------------
procédure pour générer le convertisseur
-----------------------------------*/

procedure GenConverter ()

{
intro();

printf(f,"public class dbtoxmlconverter\n");
printf(f,"{\n");

declProcEcriture();
printf(f,"\n");
printf(f,"\npublic static void main (String args[])\n");
printf(f,"\t{\n");
```

```
GenExtractMod();
printf(f,"\n");
GenWriteMod();

printf(f,"\t}\n");
printf(f,"}\n");
declClass();

}

/* ------------------------------
procedure pour ecrire les classes
------------------------------ */

procedure intro ()

{
printf(f,"import java.io.*;\n");
printf(f,"import java.lang.*;\n");
printf(f,"import java.sql.*;\n");
printf(f,"import java.util.Vector;\n");
}

/*------------------------------
procedure pour declarer les procédures d'ecriture dans un fichier
------------------------------ */

procedure declProcEcriture ()

{

// declaration de la procedure pour ecrire une ligne dans un fichier

printf(f,"//Declaration de la procédure pour écrire dans un fichier\n\n");
printf(f,"\npublic static void writln ( String file)\n");
printf(f,"{\n");
printf(f,"\ttry\n");
printf(f,"\t{\n");
printf(f,"\tFileOutputStream fos = new FileOutputStream (file,true);\n");
printf(f,"\tfos.write(10);\n");
printf(f,"\t}\n");
printf(f,"\tcatch (IOException e)\n");
printf(f,"\t{\n");
printf(f,"\tSystem.err.println (e);\n");
printf(f,"\t}\n");
printf(f,"}\n\n");

// declaration de la Procedure pour passer une ligne dans un fichier

printf(f,"// Procedure pour passer une ligne dans un fichier\n\n");
printf(f,"public static void writ (String args, String file)\n");
printf(f,"{\n");
printf(f,"\ttry\n");
printf(f,"\t{\n");
printf(f,"\tFileOutputStream fos = new FileOutputStream (file,true);\n");
printf(f,"\tint length = args.length();\n");
printf(f,"\tchar[] temp = new char[length+1];\n");
printf(f,"\targs.getChars (0,length,temp,0);\n");
printf(f,"\tint i = 0;\n");
```

```
printf(f,"\tbyte[] b = new byte [length+1];\n");
printf(f,"\twhile (length>=0)\n");
printf(f,"\t\t{\n");
printf(f,"\t\tb[i] = (byte) temp[i];\n");
printf(f,"\t\ti = i+1;\n");
printf(f,"\t\tlength=length-1;\n");
printf(f,"\t\t}\n");
printf(f,"\tfos.write (b);\n");
printf(f,"\t}\n");
printf(f,"\tcatch (IOException e)\n");
printf(f,"\t{\n");
printf(f,"\tSystem.err.println (e);\n");
printf(f,"\t}\n");
printf(f,"}\n\n");

}

/* ----------------------------------------
Procedure pour générer le module d'extraction
---------------------------------------- */

procedure GenExtractMod ()

{
Connect();
Extract();
extract2();
Disconnect();
}

/* ----------------------------------------
Procedure pour générer la connection au serveur local
---------------------------------------- */

procedure Connect ()


string: url, login, pswd;

{
printf (f, "// Connection au serveur local\n");

printf(f,"try {\n");
printf (f,"Class.forName(\"jdbc.interDB.cql\");\n");
printf(f,"} catch (java.lang.ClassNotFoundException e) {\n");
printf(f," System.err.print (\"ClassNotFoundException: \");\n");
printf(f," System.err.println (e.getMessage());\n");
printf(f,"}\n");

print("URL du serveur local: ");
url:=read(_string);
print("\nLogin : ");
login:=read(_string);
print("\nMot de passe : ");
pswd:=read(_string);
print("\n");
initresultset();
printf (f,"try {\n");
printf (f,["Connection con = DriverManager.getConnection(\"",url,"\",\"",login,"\",\"",pswd,"\");\n\n"]);
```

```
}

/* ----------------------------------------
Procedure d'extraction, crée un resultSet pour chaque entité.
---------------------------------------- */

procedure Extract ()

entity_type:ent;
data_object:dat;
attribute:att,att1;
string:m;

{

printf (f,"// Extraction des données par TE\n");
printf (f,"Statement stmt = con.createStatement();\n");

for dat in DATA_OBJECT[dat] {@SCH_DATA:[sch] with GetType(dat)=ENTITY_TYPE }
do
        {
        init();
        renomme :=0;
        ancien :="";
        ent:=dat;
        m := ent."corresp";
        analysemapET(m);
        for att in ATTRIBVTE[att] {@OWNER_ATT:[ent]} do
                {
                m := att."corresp";
                if GetType(att)=SI_ATTRIBUTE
                        then
                        {analysemapATT(m,ent.name);}
                        else
                        {
                        if GetType(att)=CO_ATTRIBUTE then
                                {analysemapCOATT(m,ent.name,att);}
                        }
                }
        printf (f,"rs"+ent.name+" = stmt.executeQuery");
        printf (f,"(\"SELECT "+select_clause+" FROM "+from_clause);
        if StrCmp(where_clause,"")<>0
                then
                {
                printf (f," WHERE "+where_clause+";\");\n");
                }
                else
                {
                printf (f,";\");\n");
                }
        }
}

/* ----------------------------------------
procedure pour initialiser les variables globales
---------------------------------------- */

procedure init()
```

```
{
select_clause:="";
from_clause:="";
where_clause:="";
}


/*-----------------------------------------------
procedure qui analyse le mapping d'un attributdecomposé

ne fonctionne que pour la transformation rename.
aucune transformation permise pour ses attributs

modifie les clauses select-from-where
----------------------------------------------- */

procedure analysemapCOATT(string:m, string:ent, co_attribute:coatt)

integer:pospar,pospoint;
string:t;
string:nom, nomCA;
attribute:att;

{

pospar := StrFindSubStr (m,0,"(");
if pospar <>-1
        then
        {
        t := StrGetSubStr(m,0,pospar);
        }
        else
        {
        t := ("No");
        }
switch (t)
        {
        case "No" :
                pospoint := StrFindSubStr(m,0,".");
                nom:=StrGetSubStr(m,0,pospoint);
                nomCA:=StrGetSubStr(m,pospoint+1,StrLength(m)-pospoint-1);
                if renomme = 1 then {nom := ancien;}
                for att in ATTRIBUTE[att]{@OWNER_ATT:[coatt] with
GetType(att)=SI_ATTRIBUTE} do
                        {
                        if StrCmp(select_clause,"")=0
                                then
                                {
                                select_clause := nom+"."+nomCA+"."+att.name;
                                }
                                else
                                {
                                select_clause := select_clause+" , "+nom+"."+nomCA+"."+att.name;
                                }
                        }
                select_clause := select_clause+" ";
                select_name:= nom+"."+nomCA;
```

```
                if ismember(from_clause,nom)=0 and renomme = 0 then
                        {
                        if StrCmp(from_clause,"")=0
                        then
                        {
                        from_clause := nom+" ";
                        }
                        else
                        {
                        if ismember(from_clause,nom)
                                then {from_clause:= from_clause+" , "+nom+" ";}
                        }
                        }
        case "rename" :
                nom := ent;
                if renomme = 1 then {nom := ancien;}

                select_name :=  nom+"."+StrGetSubStr(m,pospar+1,StrLength(m)-pospar-2);

                for att in ATTRIBUTE[att]{@OWNER_ATT:[coatt] with
GetType(att)=SI_ATTRIBUTE} do
                        {
                        if StrCmp(select_clause,"")=0
                                then
                                {
                                select_clause := nom+"."+StrGetSubStr(m,pospar+1,StrLength(m)-
pospar-
                                                                                        2)+"."+att.name;
                                }
                                else
                                {
                                select_clause := select_clause+" ,
                                 "+nom+"."+StrGetSubStr(m,pospar+1,StrLength(m)-pospar-
                        2)+"."+att.name;
                                }
                        }
                select_clause := select_clause+" ";
                if ismember(from_clause,nom)=0 and renomme = 0 then
                        {
                        if StrCmp(from_clause,"")=0
                        then
                        {
                        from_clause := nom+" ";
                        }
                        else
                        {
                        if ismember(from_clause,nom)
                                then {from_clause:= from_clause+" , "+nom+" ";}
                        }
                        }
        otherwise : print(t+" est une transformation non traitée");
        }

}

/* ----------------------------------------
procedure qui analyse le mapping d'un attribut
modifie les clauses select-from-where
---------------------------------------- */
```

```
procedure analysemapATT(string:m, string:ent)

integer:pospar;
string:t;
string:nom,nomA,nomCA,nomR;
integer:pospoint,posvir,posvir2;

{

pospar := StrFindSubStr (m,0,"(");
if pospar <>-1
        then
        {
        t := StrGetSubStr(m,0,pospar);
        }
        else
        {
        t := ("No");
        }
switch(t)
        {
        case "rename" :
                nom := ent;
                if renomme = 1 then {nom := ancien;}
                select_name := nom+"."+StrGetSubStr(m,pospar+1,StrLength(m)-pospar-2);
                if StrCmp(select_clause,"")=0
                        then
                        {
                        select_clause := nom+"."+StrGetSubStr(m,pospar+1,StrLength(m)-pospar-2)+"
";
                        }
                        else
                        {
                        select_clause := select_clause+" ,
"+nom+"."+StrGetSubStr(m,pospar+1,StrLength(m)-
                                                                                pospar-
                                                                        2)+" ";
                        }
        case "att-to-et\\inst" :
                posvir := StrFindSubStr(m,0,",");
                nom := StrGetSubStr(m,pospar+1,posvir-pospar-1);
                nomA := StrGetSubStr(m,posvir+1,StrLength(m)-posvir-2);
                if renomme = 1 then {nom := ancien;}
                if ismember(from_clause,nom)=0 then
                        {
                        from_clause:= from_clause+" , "+nom;
                        }
                select_name :=nom+"."+nomA;
                if StrCmp(select_clause,"")=0
                        then
                        {
                        select_clause := nom+"."+nomA+" ";
                        }
                        else
                        {
                        select_clause := select_clause+" , "+nom+"."+nomA+" ";
                        }
        case "disaggregate" :
```

```
            posvir := StrFindSubStr(m,0,",");
            nom := StrGetSubStr(m,pospar+1,posvir-pospar-1);
            pospoint := StrFindSubStr(m,0,".");
            nomCA := StrGetSubStr(m,posvir+1,pospoint-posvir-1);
            nomA := StrGetSubStr(m,pospoint+1,StrLength(m)-pospoint-2);
            if renomme = 1 then {nom := ancien;}
            if ismember(from_clause,nom)=0 then
                        {
                        from_clause:= from_clause+" , "+nom+" ";
                        }
            select_name := nom+"."+nomCA+"."+nomA;
            if StrCmp(select_clause,"")=0
                        then
                        {
                        select_clause := nom+"."+nomCA+"."+nomA+" ";
                        }
                        else
                        {
                        select_clause := select_clause+" , "+nom+"."+nomCA+"."+nomA+" ";
                        }
    case "split-merge" :
            posvir := StrFindSubStr(m,0,",");
            nom := StrGetSubStr(m,pospar+1,posvir-pospar-1);
            posvir2 := StrFindSubStr(m,posvir+1,",");
            nomA := StrGetSubStr(m,posvir+1,posvir2);
            nomR := StrGetSubStr(m,posvir+posvir2+2,StrLength(m)-posvir2-posvir-3);
            select_name := nom+"."+nomA;
            if StrCmp(select_clause,"")=0
                        then
                        {
                        select_clause := nom+"."+nomA+" ";
                        }
                        else
                        {
                        select_clause := select_clause+" , "+nom+"."+nomA+" ";
                        }
            if ismember(from_clause,nom)=0 then
                        {
                        from_clause:= from_clause+" , "+nom+" ";
                        }
            if renomme = 1 then {ent := ancien;}
            if StrCmp(where_clause,"")=0
                        then
                        {
                        where_clause := nom+" "+nomR+" "+ent+" ";
                        }
                        else
                        {
                        where_clause := where_clause+" and "+nom+" "+nomR+" "+ent+" ";
                        }
    case "et-to-att" :
            posvir := StrFindSubStr(m,0,",");
            nom := StrGetSubStr(m,pospar+1,posvir-pospar-1);
            posvir2 := StrFindSubStr(m,posvir+1,",");
            nomA := StrGetSubStr(m,posvir+1,posvir2);
            nomR := StrGetSubStr(m,posvir+posvir2+2,StrLength(m)-posvir2-posvir-3);
            select_name := nom+"."+nomA;
            if StrCmp(select_clause,"")=0
                        then
```

```
                {
                select_clause := nom+"."+nomA+" ";
                }
                else
                {
                select_clause := select_clause+" , "+nom+"."+nomA+" ";
                }
        if ismember(from_clause,nom)=0 then
                {
                from_clause:= from_clause+" , "+nom+" ";
                }
        if renomme = 1 then {ent := ancien;}
        if StrCmp(where_clause,"")=0
                then
                {
                where_clause := nom+" "+nomR+" "+ent+" ";
                }
                else
                {
                where_clause := where_clause+" and "+nom+" "+nomR+" "+ent+" ";
                }
    case "No" :
        pospoint := StrFindSubStr(m,0,".");
        nom:=StrGetSubStr(m,0,pospoint);
        nomA:=StrGetSubStr(m,pospoint+1,StrLength(m)-pospoint);
        if renomme = 1 then {nom := ancien;}
        select_name := nom+"."+nomA;
        if StrCmp(select_clause,"")=0
                then
                {
                select_clause := nom+"."+nomA+" ";
                }
                else
                {
                select_clause := select_clause+" , "+nom+"."+nomA+" ";
                }
        if ismember(from_clause,nom)=0 and renomme = 0 then
                {
                if StrCmp(from_clause,"")=0
                then
                {
                from_clause := nom+" ";
                }
                else
                {
                if ismember(from_clause,nom)
                        then {from_clause:= from_clause+" , "+nom+" ";}
                }
                }
    otherwise : print(t+" est une transformation non traitée");
    }

}

/* ------------------------------------
procedure d'analyse d'une métapropriété d'un ET
modific from_clause
------------------------------------ */
```

```
procedure analysemapET (string : m)

integer: pospar,pospar2,posvir;
string:t; // type de transformation

{
pospar := StrFindSubStr (m,0,"(");
if pospar <>-1
        then
        {
        t := StrGetSubStr(m,0,pospar);
        }
        else
        {
        t := ("No");
        }
switch (t)
        {
        case "rename" :
                pospar2:=StrFindSubStr(m,pospar,")");
                from_clause:= StrGetSubStr(m,pospar+1,pospar2-1)+" ";
                renomme := 1;
                ancien := StrGetSubStr(m,pospar+1,pospar2-1);
        case "No" :
                from_clause:=m;
        case "att-to-et\\inst" :
                posvir:=StrFindSubStr(m,pospar,",");
                from_clause := StrGetSubStr(m,pospar+1,posvir-1)+" ";
                philou := 1;
        otherwise : print ([t," est une transformation non traitée"]);
        }

}

/* ------------------------------------------------------------------
fonction qui verifie qui change le nom du resultset de facon à ce qu'il soit bien ecrit
------------------------------------------------------------------ */

function string changeRSname(string:rsname, string:ent, string:ent1)

list : ltemp;

{
ltemp := recherche(rsname);
if Length(ltemp)=0
        then
        {
        return "rs"+ent1+ent;
        }
        else
        {
        return rsname;
        }
}

/* ------------------------------------------------------------------
fonction qui donne la liste dans correspondance qui correspond au nom de la relation
------------------------------------------------------------------ */
```

```
function list recherche (string:rel)

cursor: c1, c2;
list: temp;
integer:i;

{

attach c1 to correspondance;
for i in [1..Length(correspondance)] do
        {
        temp := get(c1);
        attach c2 to temp;
        if StrCmp(get(c2),rel)=0 then { return temp;}
        c1>>;
        }
}

/* ----------------------------------------
procedure pour initialiser les resultsets
non optimise car cree trop de resultsets
---------------------------------------- */

procedure initresultset ()

data_object:dat;
entity_type:ent,ent1;
role:r;
et_role:er;
rel_type:rel;

{

for dat in DATA_OBJECT[dat]{ @SCH_DATA:[sch] with GetType(dat)=ENTITY_TYPE}do
        {
        ent:=dat;
        printf(f,"ResultSet rs"+ent.name+" = null;\n");
        }
for dat in DATA_OBJECT[dat]{ @SCH_DATA:[sch] with GetType(dat)=ENTITY_TYPE}do
        {
        ent:=dat;
        for rel in REL_TYPE[rel] {RT_RO: ROLE[r] {RO_ETR: ET_ROLE[er] { @ENTITY_ETR
:[ent]}}} do
                        {
                        ent1 := relation(ent, rel);
                        printf(f,"ResultSet rs"+ent.name+ent1.name+" = null;\n");
                        }
        }

}

/*----------------------------------------------------------
procedure qui cree un ResultSet avec les ET reliées par un RT
en ne prenant que les identifiants
---------------------------------------------------------- */

procedure extract2 ()
```

```
data_object:dat;
attribute:att,att1;
rel_type:rel;
entity_type:ent,ent1;
role:r;
et_role:er;
string:me,ma,mr;
string:from_clause2;
integer:renomme1;
string:ancien1;
string:nom, nom1;
list:l;

list:lnee;

{
for dat in DATA_OBJECT[dat] {@SCH_DATA:[sch] with GetType(dat)=ENTITY_TYPE}
        do
        {
        philou :=0;
        ent:=dat;
        if visite(l,ent.name) =0
                then
                {
                for rel in REL_TYPE[rel] {RT_RO: ROLE[r] {RO_ETR: ET_ROLE[er]
{@ENTITY_ETR :[ent]}}}
                        do
                        {
                        ent1 := relation(ent, rel);
                        if visite(l,ent1.name)=0
                                then
                                {
                                init();
                                renomme:=0;
                                ancien:="";

                                me := ent."corresp";
                                philou:=0;
                                analysemapET(me);

                                if philou <> 1
                                        then
                                        {
                                        renomme1 := renomme;
                                        ancien1 := ancien;
                                        from_clause2 := from_clause;
                                        att := getid(ent);
                                        ma := att."corresp";
                                        analysemapATT(ma,ent.name);

                                        renomme:=0;
                                        ancien:="";
                                        me := ent1."corresp";
                                        analysemapET(me);
                                                if philou <> 1
                                                then
                                                {
                                                att1 := getid(ent1);
                                                ma := att1."corresp";
```

```
                                                    analysemapATT(ma,ent1.name);
                                                    mr := rel."corresp";
                                                    if renomme1 =1 then {nom := ancien1;} else {nom
:=
                                                                                        ent.nam
                                                                                    e;}
                                                    if renomme =1 then { nom1 := ancien;} else {nom1
:=
                                                                                        ent1.na
                                                                                    me;}
                                                    analysemapR(mr,nom,nom1);
                                                    from_clause2 := from_clause2+" , "+from_clause;
                                                    printf (f,"rs"+ent.name+ent1.name+" =
stmt.executeQuery");
                                                    printf (f,"(\"SELECT "+select_clause+" FROM
                                                                                "+from_clause2);
                                                    printf (f," WHERE "+where_clause+";\");\n");
                                                    lnee := [rel.name,"rs"+ent.name+ent1.name];
                                                    AddFirst(correspondance,lnee);

                                                    }
                                        }
                                }
                        }
                AddFirst(l,ent.name);
                }
        }
}

/* ----------------------------------------
procedure pour générer la deconnection au serveur local
---------------------------------------- */

procedure Disconnect ()

{
printf (f,"// deconnection au serveur local\n\n");
printf (f,"con.close();\n\n");
printf (f,"}\n");
printf(f,"catch(SQLException ex) {\n");
printf(f,"System.err.println(\"SQLException: \" + ex.getMessage());}\n");

}

/* ----------------------------------------
 fonction pour dire si un string est inclus dans un autre string
 ----------------------------------------*/

function integer ismember (string:s, string:s2)

{
s2 := s2+" ";
if StrFindSubStr (s,0,s2) >=0
        then
        {
        return 1;
        }
        else
        {
```

```
        return 0;
        }
}


/* ----------------------------------------------------
procedure qui analyse le maping d'un RT
modifie where_clause
---------------------------------------------------- */

procedure analysemapR (string : m, string : ent, string: ent1)

integer:pospar,pospar2;
string:t;

{

pospar := StrFindSubStr (m,0,"(");
if pospar <>-1
        then
        {
        t := StrGetSubStr(m,0,pospar);
        }
        else
        {
        t := ("No");
        }
switch(t)
        {
        case "rename" :
                pospar2:=StrFindSubStr(m,pospar,")");
                where_clause := ent+" "+StrGetSubStr(m,pospar+1,pospar2-1)+" "+ent1;
        case "No" :
                where_clause := ent+" "+m+" "+ent1;
        otherwise : print (t+" est une transformation non traitée\n");
        }
}

/* ----------------------------------------------------
fonction qui donne la deuxieme entite d'une relation
---------------------------------------------------- */

function entity_type relation (entity_type:ent, rel_type:rel)

et_role:er;
role:r;
entity_type:ent1;

{
for ent1 in ENTITY_TYPE[ent1]{ENTITY_ETR : ET_ROLE[ er]{ @RO_ETR : ROLE[r]{ @RT_RO :
[rel]}}} do
        {
        if ent1.name <> ent.name
                then
                {
                return ent1;
                }
        }
```

```
}

/*-----------------------------------------------------
fonction qui donne l'attribut identifiant d'un ET
----------------------------------------------------- */

function attribute getid (entity_type:ent)


attribute:att;
group:gr;
real_component:rc;
component:co;
integer:typrc;

{
for gr in GROUP[gr]{@DATA_GR:[ent] with gr.primary} do
        {
        for rc in REAL_COMPONENT[rc]{REAL_COMP:COMPONENT[co]{@GR_COMP:[gr]}} do
                {
                if GetType(rc)=SI_ATTRIBUTE
                        then
                        {
                        att := rc;
                        return att;
                        }
          }
        }
}

/*-----------------------------------------------
fonction qui dit si une entite a ete visitee
----------------------------------------------- */

function integer visite (list : l, string:ent)

{

if IsVoid(member(l,ent)) then { return 0;} else {return 1;}
}

/*-----------------------------------------------
procedure pour générer le module d'écriture dans le fichier XML
----------------------------------------------- */

procedure GenWriteMod ()

entity_type:ent,ent2;
data_object:dat;
attribute:att,att2,attco;
co_attribute:coatt;
string:nomrel;
rel_type:rel;
role:r;
et_role:er;
list:l;
cursor:c;
string:nomrs;
string:me, ma;
```

```
string:nomatt,nomatt2;
string:sugus;

list:l_nomatt; // liste qui pour chaque attribut simple contient son select_name


{
printf(f,"//insertion des données dans le fichier XML\n\n");
printf(f,"String s = new String (\"\");\n");
printf(f,"String valatt= new String (\"\");\n");
printf(f,"String valid= new String (\"\");\n");
printf(f,"String valid2= new String (\"\");\n");
printf(f,"String validtemp= new String (\"\");\n");
printf(f,"String sidref= new String (\"\");\n");
printf(f,"String idx = new String (\"\");\n");
printf(f,"String idx2 = new String (\"\");\n");
printf(f,"Boolean b;\n");
printf(f,"String fout = \""+sch.name+".xml\";\n");

printf(f,"\n");

printf(f,"//Initialisation du vecteur tidxml qui contient les identifiants XML(entre autre)\n\n");
printf(f,"tabididref tidxml = new tabididref();\n");
for dat in DATA_OBJECT[dat] {@SCH_DATA:[sch] with GetType(dat)=ENTITY_TYPE} do
        {
        ent:=dat;
        // recherche de la requete qui correspond à l'identifiant cad select_name
        select_name := "";
        att:= getid(ent);
        ma := att."corresp";
        init();
        renomme :=0;
        ancien :="";
        me := ent."corresp";
        analysemapET(me);
        analysemapATT(ma,ent.name);
        printf(f,"tidxml.addelem( \""+ent.name+"\",\""+select_name+"\",rs"+ent.name+");\n"     );
        }

printf(f,"\n//Ecriture de l'entete du fichier xml : reference au DTD\n\n");
printf(f,"s = \"<?xml version = \\" + "\"1.0"+"\\" + "\"?>\";\n");
printf(f,"writ(s,fout);\n");
printf(f,"writln(fout);\n");
printf(f,"s = \"<!DOCTYPE "+sch.name+" SYSTEM  "+ "\\" + "\""+sch.name+".dtd"+"\\"+"\">\";\n");
printf(f,"writ(s,fout);\n");
printf(f,"writln(fout);\n");
printf(f,"s = \"<"+sch.name+">\";\n");
printf(f,"writ(s,fout);\n");
printf(f,"writln(fout);\n");

printf(f,"\n//Ecriture des entités\n\n");
for dat in DATA_OBJECT[dat] {@SCH_DATA:[sch] with GetType(dat)=ENTITY_TYPE}
        do
        {
        select_name:="";
        init();
        renomme :=0;
        ancien :="";
        ent:=dat;
```

```
me := ent."corresp";
analysemapET(me);
att:= getid(ent);
ma := att."corresp";
analysemapATT(ma,ent.name);
nomatt:= select_name;
printf(f,"b = rs"+ent.name+".first();\n");
        // while qui ecrit chaque entité dans le fichier XML
printf(f,"while (rs"+ent.name+".next())\n");
printf(f,"\t{\n");
printf(f,"\tvalid = (String) rs"+ent.name+".getString(\""+nomatt+"\");\n");
printf(f,"\ttidxml.getidxml(\""+ent.name+"\",valid,idx);\n");

printf(f,"\t// s= l'entite avec son identifiant\n");
printf(f,"\ts = \"<"+ent.name+" id=\"+idx+\" \";\n");
for rel in REL_TYPE[rel] {RT_RO: ROLE[r] {RO_ETR: ET_ROLE[er] {@ENTITY_ETR
:[ent]}}}
do
        {
        select_name:="";
        init();
        renomme :=0;
        ancien :="";
        ent2:= relation(ent,rel);
        nomrel:= rel.name+"."+ent2.name;
        att2:=getid(ent2);
        ma := att2."corresp";
        analysemapATT(ma,ent2.name);
        nomatt2:= select_name;
        l := recherche(rel.name);
        attach c to l;
        c>>;
        nomrs:=get(c);
        printf(f,"\n\tsidref = \"\";\n");
        printf(f,"\tb = "+nomrs+".first();\n");


        printf(f,"\n\t// while qui donne la liste des idref\n");
        printf(f,"\twhile ("+nomrs+".next())\n");
        printf(f,"\t\t{\n");
        printf(f,"\t\tvalid2 = (String) "+nomrs+".getString(\""+nomatt2+"\");\n");
        printf(f,"\t\tvalidtemp = (String) "+nomrs+".getString(\""+nomatt+"\");\n");
        printf(f,"\t\tif (validtemp == valid)\n");
        printf(f,"\t\t\t{\n");
        printf(f,"\t\t\ttidxml.getidxml(\""+ent2.name+"\",valid2,idx2);\n");
        printf(f,"\t\t\tsidref=sidref.concat(\" \\"+"\"\"+idx2+\"\\"+"\" \");\n");
        printf(f,"\t\t\t}\n");
        printf(f,"\t\t}\n");
                // fin du while des idref
        printf(f,"\ts = s.concat(\""+rel.name+"."+ent2.name+" = \"+sidref);\n");
        }

printf(f,"\twrit(s,fout);\n");
printf(f,"\twrit(\">\",fout);\n");
printf(f,"\twritln(fout);\n");


for att in ATTRIBUTE[att]{@OWNER_ATT:[ent] with GetType(att)=SI_ATTRIBUTE} do
        {
```

```
                select_name := "";
                ma := att."corresp";
                analysemapATT(ma,ent.name);
                sugus:=ent.name+"."+att.name;
                AddFirst(l_nomatt,[sugus,select_name]);
                }
        for att in ATTRIBUTE[att]{ @OWNER_ATT:[ent] with GetType(att)=CO_ATTRIBUTE} do
                {
                coatt:=att;
                select_name := "";
                ma := coatt."corresp";
                analysemapCOATT(ma,ent.name,coatt);
                for attco in ATTRIBUTE[attco]{ @OWNER_ATT:[coatt] with
GetType(attco)=SI_ATTRIBUTE} do
                        {
                        sugus:=ent.name+"."+coatt.name+"."+attco.name;
                        AddFirst(l_nomatt,[sugus,select_name+"."+attco.name]);
                        }
                }
        printf(f,"\n\t// ecriture de chaque attribut simple de l'entite\n\n");

        for att in ATTRIBUTE[att]{ @OWNER_ATT:[ent] with GetType(att)=SI_ATTRIBUTE} do
                {
                sugus := donneNA(l_nomatt,ent.name+"."+att.name);
                printf(f,"\tvalatt = (String) rs"+ent.name+".getString(\""+sugus+"\");\n");
                printf(f,"\ts =
                        \"<"+ent.name+"."+att.name+">\"+valatt+\"</"+ent.name+"."+att.name+">\";\
                n");
                printf(f,"\twrit(s,fout);\n");
                printf(f,"\twritln(fout);\n");
                }

        printf(f,"\n\t// ecriture de chaque attribut composé de l'entite\n\n");
        // un attribut d'un attribut compose ne peut etre composé (cas non traite)
        for att in ATTRIBUTE[att]{ @OWNER_ATT:[ent] with GetType(att)=CO_ATTRIBUTE} do
                {
                coatt:=att;
                printf(f,"\ts =\"<"+ent.name+"."+coatt.name+">\";\n");
                printf(f,"\twrit(s,fout);\n");
                for attco in ATTRIBUTE[attco]{ @OWNER_ATT:[coatt] with
                                                GetType(attco)=SI_ATTRIBUTE} do
                        {
                        sugus := donneNA(l_nomatt,ent.name+"."+coatt.name+"."+attco.name);
                        printf(f,"\tvalatt = (String) rs"+ent.name+".getString(\""+sugus+"\");\n");
                        printf(f,"\ts =
                                \"<"+ent.name+"."+coatt.name+"."+attco.name+">\"+valatt+\"</"+ent
                        .name+  "."+coatt.name+"."+attco.name+">\";\n");
                        printf(f,"\twrit(s,fout);\n");
                        printf(f,"\twritln(fout);\n");
                        }
                printf(f,"\ts =\"</"+ent.name+"."+coatt.name+">\";\n");
                printf(f,"\twrit(s,fout);\n");
                printf(f,"\twritln(fout);\n");
                }

        printf(f,"\ts = \"</"+ent.name+">\";\n");
        printf(f,"\twrit(s,fout);\n");
        printf(f,"\twritln(fout);\n");
        printf(f,"\t}\n");
```

```
                    // fin du while qui ecrit chaque entite


          }
printf(f,"s = \"</"+sch.name+">\";\n");
printf(f,"writ(s,fout);\n");
printf(f,"writln(fout);\n");

}

/* ----------------------------
fonction qui donne le nomatt d'un attribut qui est dans la liste l_nomatt
---------------------------------- */

function string donneNA(list: l_nomatt, string:s)

cursor:c,c2;
integer:i;
list:l;

{
attach c to l_nomatt;
for i in [1..Length(l_nomatt)] do
          {
          l := get(c);
          attach c2 to l;
          if StrCmp(get(c2),s)=0
                    then { c2>>; return get(c2);}
          c>>;
          }
}

/*-----------------------------------------
procedure pour declarer les classes qui contiennent les id et idref
----------------------------------------- */

procedure declClass ()

entity_type:ent;
attribute:att;
data_object:dat;
string:ma,me;
{
printf(f,"/* Class qui contient le nom de l'entité, un vecteur avec les valeurs de son identifiant, un vecteur
avec la valeur de l'identifiant xml*/\n");
printf(f,"class ididref\n");
printf(f,"\t{\n");
printf(f,"\tpublic String entite;\n");
printf(f,"\tpublic Vector idDB;\n");
printf(f,"\tpublic Vector idxml;\n\n");

printf(f,"\n//constructor\n\n");
printf(f,"\tpublic ididref ()\n");
printf(f,"\t\t{\n");
printf(f,"\t\tidDB = new Vector ();\n");
printf(f,"\t\tidxml = new Vector ();\n");
printf(f,"\t\t}\n\n");

printf(f,"\tpublic ididref (String e, Vector d, Vector x)\n");
```

```
printf(f,"\t\t{\n");
printf(f,"\t\tthis();\n");
printf(f,"\t\tentite = e;\n");
printf(f,"\t\tidDB = d;\n");
printf(f,"\t\tidxml = x;\n");
printf(f,"\t\t}\n");

printf(f,"\n//methodes \n");
printf(f,"\n\tpublic Vector getidDB ()\n");
printf(f,"\t\t{\n");
printf(f,"\t\treturn idDB;\n");
printf(f,"\t\t}\n");

printf(f,"\n\tpublic Vector getidx ()\n");
printf(f,"\t\t{\n");
printf(f,"\t\treturn idxml;\n");
printf(f,"\t\t}\n");

printf(f,"\n\tpublic String getentite ()\n");
printf(f,"\t\t{\n");
printf(f,"\t\treturn entite;\n");
printf(f,"\t\t}\n");

printf(f,"\t}\n\n");

printf(f,"//Class qui est un vecteur de  ididref\n");
printf(f,"\nclass tabididref\n");
printf(f,"\t{\n");
printf(f,"\tpublic Vector tid;\n");

printf(f,"\n// constructor\n");
printf(f,"\n\tpublic tabididref ()\n");
printf(f,"\t\t{\n");
printf(f,"\t\ttid = new Vector ();\n");
printf(f,"\t\t}\n");

printf(f,"\n//methode pour ajouter un element dans le vecteur \n");
printf(f,"\n\tpublic void addelem (String ent, String query, ResultSet rs)\n");
printf(f,"\t\t{\n");
printf(f,"\t\tint i=0;\n");
printf(f,"\t\tint cpt=0;\n");
printf(f,"\t\tboolean b;\n");

printf(f,"\n\t\t//Recherche du nombre d'identifiants XMLdeja attribues\n\n");
printf(f,"\t\twhile (i < tid.size())\n");
printf(f,"\t\t\t{\n");
printf(f,"\t\t\tcpt = cpt + (((ididref)tid.elementAt(i)).getidx()).size();\n");
printf(f,"\t\t\ti=i+1;\n");
printf(f,"\t\t\t}\n");

printf(f,"\n\t\t//création de l'ididref à insérer dans tid\n\n");
printf(f,"\t\tString temp = new String ();\n");
printf(f,"\t\tb=rs.first();\n");
printf(f,"\t\tVector vid = new Vector ();\n");
printf(f,"\t\tVector vidx = new Vector ();\n");

printf(f,"\n\t\t//initialisation de vid et vidx\n\n");
printf(f,"\t\twhile (rs.next())\n");
printf(f,"\t\t\t{\n");
```

```
printf(f,"\t\t\ttemp = (String) rs.getString(query);\n");
printf(f,"\t\t\tvid.addElement(temp);\n");
printf(f,"\t\t\tcpt = cpt +1;\n");
printf(f,"\t\t\tvidx.addElement(cpt.toString());\n");
printf(f,"\t\t\t}\n");

printf(f,"\n\t\t// creation de idf\n");
printf(f,"\t\tididref idf = new ididref (ent, vid, vidx);\n");

printf(f,"\n\t\t// ajout de idf dans tid\n");
printf(f,"\t\ttid.addElement(idf);\n");

printf(f,"\t\t}\n");

printf(f,"// methode pour rechercher l'identifiant xml qui correspond à l'entite\n");
printf(f,"\n\t\tpublic void getidxml(String ent, String ident, String idx)\n");
printf (f,"\t\t{\n");
printf (f,"\t\tint cmpt=0;\n");
printf (f,"\t\tint cmpt2=0;\n");
printf (f,"\t\twhile (ent.compareTo(((ididref)tid.elementAt(cmpt)).getentite())!=0)\n\t");
printf (f,"\t\t{\n\t");
printf (f,"\t\tcmpt=cmpt+1;\n\t");
printf (f,"\t\t}\n");
printf (f,"\t\twhile
(ident.compareTo((String)((Vector)(((ididref)tid.elementAt(cmpt)).getidDB())).elementAt(cmpt2))!=0)\n\
t");
printf (f,"\t\t{\n\t");
printf (f,"\t\tcmpt2=cmpt2+1;\n\t");
printf (f,"\t\t}\n");
printf (f,"\t\tidx = (String)((Vector)((ididref)tid.elementAt(cmpt)).getidx()).elementAt(cmpt2);\n");
printf (f,"\t\t}\n\n");

printf(f,"\t}\n\n");

}

/* ***************************
 programme principal
***************************** */

begin
if OuvreFichier () then
{
sch := GetCurrentSchema();
if not(IsVoid (sch)) then
        {
        GenConverter ();
        print ("Ok\n");
        }
CloseFile (f);
}
end
```

Appendix I : DB-to-DB Log file

```
*POT "begin-file"
*DEL GRP
%BEG
        %OID 2392
        %NAM "IDTELEPHONE"
        %OWN 3 "Cinebase"/"conceptual-12"."TELEPHONE" 2391
        %COM 8 "Cinebase"/"conceptual-12"."has"."DISTRIBUTOR" 2431
        %COM 6 "Cinebase"/"conceptual-12"."TELEPHONE"."Number" 2400
        %TYP A
        %FLA "P"
        %MET "line" "" "V"
        &MOD GRP
        %BEG
                *OLD GRP
                %BEG
                        %OID 2392
                        %NAM "IDTELEPHONE"
                        %OWN 3 "Cinebase"/"conceptual-12"."TELEPHONE" 2391
                %END
                %OID 2392
                %NAM "IDTELEPHONE"
        %END
%END
*DEL SIA
%BEG
        %OID 2400
        %NAM "Number"
        %OWN 3 "Cinebase"/"conceptual-12"."TELEPHONE" 2391
        %CAR 1-1
        %TYP N
        %SET S
        %LEN 12
        %DEC 0
        %MET "corresp" "" "VU"
        %MET "line" "397:117" "V"
%END
*DEL ROL
%BEG
        %OID 2429
        %POX 158679
        %POY 38793
        %OWN 2 "Cinebase"/"conceptual-12"."has" 2406
        %ETR 3 "Cinebase"/"conceptual-12"."TELEPHONE" 2391
        %CAR 1-1
        &MOD ROL
        %BEG
                *OLD ROL
                %BEG
                        %OID 2429
```

```
                %OWN 2 "Cinebase"/"conceptual-12"."has" 2406
                %ETR 3 "Cinebase"/"conceptual-12"."TELEPHONE" 2391
        %END
        %OID 2429
        %OWN 2 "Cinebase"/"conceptual-12"."has" 2406
    %END
%END
*DEL ROL
%BEG
    %OID 2431
    %POX 159060
    %POY 54734
    %OWN 2 "Cinebase"/"conceptual-12"."has" 2406
    %ETR 3 "Cinebase"/"conceptual-12"."DISTRIBUTOR" 2411
    %CAR 0-5
    &MOD ROL
    %BEG
        *OLD ROL
        %BEG
            %OID 2431
            %OWN 2 "Cinebase"/"conceptual-12"."has" 2406
            %ETR 3 "Cinebase"/"conceptual-12"."DISTRIBUTOR" 2411
        %END
        %OID 2431
        %OWN 2 "Cinebase"/"conceptual-12"."has" 2406
    %END
%END
*DEL REL
%BEG
    %OID 2406
    %NAM "has"
    %SNA "D_T"
    %POX 158870
    %POY 46763
    %OWN 1 "Cinebase"/"conceptual-12" 2379
    %MET "corresp" "" "VU"
%END
*DEL ENT
%BEG
    %OID 2391
    %NAM "TELEPHONE"
    %POX 158342
    %POY 24604
    %OWN 1 "Cinebase"/"conceptual-12" 2379
    %MET "corresp" "" "VU"
    %MET "line" "" "V"
%END
*DEL GRP
%BEG
    %OID 2399
    %NAM "IDPRODUCTEUR"
    %OWN 3 "Cinebase"/"conceptual-12"."PRODUCER" 2398
    %COM 6 "Cinebase"/"conceptual-12"."PRODUCER"."Name" 2403
    %TYP A
    %FLA "P"
    %MET "line" "397:115" "V"
    &MOD GRP
    %BEG
        *OLD GRP
```

```
                %BEG
                        %OID 2399
                        %NAM "IDPRODUCTEUR"
                        %OWN 3 "Cinebase"/"conceptual-12"."PRODUCER" 2398
                %END
                %OID 2399
                %NAM "IDPRODUCTEUR"
        %END
%END
*DEL SIA
%BEG
        %OID 2418
        %NAM "City"
        %OWN 6 "Cinebase"/"conceptual-12"."PRODUCER"."Address" 2422
        %PRV 6 "Cinebase"/"conceptual-12"."PRODUCER"."Address"."Zip-code" 2386
        %CAR 1-1
        %TYP A
        %SET S
        %LEN 20
        %MET "corresp" "" "VU"
        %MET "line" "" "V"
%END
*DEL SIA
%BEG
        %OID 2386
        %NAM "Zip-code"
        %OWN 6 "Cinebase"/"conceptual-12"."PRODUCER"."Address" 2422
        %PRV 6 "Cinebase"/"conceptual-12"."PRODUCER"."Address"."Street" 2394
        %CAR 1-1
        %TYP N
        %SET S
        %LEN 4
        %DEC 0
        %MET "corresp" "" "VU"
        %MET "line" "" "V"
%END
*DEL SIA
%BEG
        %OID 2394
        %NAM "Street"
        %OWN 6 "Cinebase"/"conceptual-12"."PRODUCER"."Address" 2422
        %CAR 1-1
        %TYP A
        %SET S
        %LEN 50
        %MET "corresp" "" "VU"
        %MET "line" "" "V"
%END
*DEL COA
%BEG
        %OID 2422
        %NAM "Address"
        %OWN 3 "Cinebase"/"conceptual-12"."PRODUCER" 2398
        %PRV 6 "Cinebase"/"conceptual-12"."PRODUCER"."Name" 2403
        %CAR 1-1
        %SET S
        %MET "corresp" "" "VU"
%END
*DEL SIA
```

```
%BEG
        %OID 2403
        %NAM "Name"
        %OWN 3 "Cinebase"/"conceptual-12"."PRODUCER" 2398
        %CAR 1-1
        %TYP A
        %SET S
        %LEN 20
        %MET "corresp" "" "VU"
        %MET "line" "397:113" "V"
%END
*DEL ROL
%BEG
        %OID 2425
        %POX 73725
        %POY 60675
        %OWN 2 "Cinebase"/"conceptual-12"."produces" 2397
        %ETR 3 "Cinebase"/"conceptual-12"."FILM" 2408
        %CAR 1-1
        &MOD ROL
        %BEG
                *OLD ROL
                %BEG
                        %OID 2425
                        %OWN 2 "Cinebase"/"conceptual-12"."produces" 2397
                        %ETR 3 "Cinebase"/"conceptual-12"."FILM" 2408
                %END
                %OID 2425
                %OWN 2 "Cinebase"/"conceptual-12"."produces" 2397
        %END
%END
*DEL ROL
%BEG
        %OID 2427
        %POX 52877
        %POY 44991
        %OWN 2 "Cinebase"/"conceptual-12"."produces" 2397
        %ETR 3 "Cinebase"/"conceptual-12"."PRODUCER" 2398
        %CAR 0-N
        &MOD ROL
        %BEG
                *OLD ROL
                %BEG
                        %OID 2427
                        %OWN 2 "Cinebase"/"conceptual-12"."produces" 2397
                        %ETR 3 "Cinebase"/"conceptual-12"."PRODUCER" 2398
                %END
                %OID 2427
                %OWN 2 "Cinebase"/"conceptual-12"."produces" 2397
        %END
%END
*DEL REL
%BEG
        %OID 2397
        %NAM "produces"
        %POX 63301
        %POY 52833
        %OWN 1 "Cinebase"/"conceptual-12" 2379
        %MET "corresp" "" "VU"
```

```
%END
*DEL ENT
%BEG
        %OID 2398
        %NAM "PRODUCER"
        %POX 34924
        %POY 31485
        %OWN 1 "Cinebase"/"conceptual-12" 2379
        %MET "corresp" "" "VU"
        %MET "line" "397:112" "V"
%END
*TRF split_merge
%BEG
        %NAM "MUSIC"
        %OWN 1 "Cinebase"/"conceptual-12" 2379
        %CAN
        %OID 2404
        %POX 31484
        %POY 103715
        %MET "corresp" "" "VU"
        %MET "line" "397:105" "V"
%END
*DEL SIA
%BEG
        %OID 2396
        %NAM "Screenwriter"
        %OWN 3 "Cinebase"/"conceptual-12"."FILM" 2408
        %PRV 6 "Cinebase"/"conceptual-12"."FILM"."Director" 2414
        %CAR 1-1
        %TYP A
        %SET S
        %LEN 20
        %MET "corresp" "" "VU"
        %MET "line" "397:95" "V"
%END
*DEL SIA
%BEG
        %OID 2420
        %NAM "Category"
        %OWN 3 "Cinebase"/"conceptual-12"."FILM" 2408
        %PRV 6 "Cinebase"/"conceptual-12"."FILM"."Abstract" 2424
        %CAR 1-1
        %TYP A
        %SET S
        %LEN 20
        %MET "corresp" "" "VU"
        %MET "line" "397:99" "V"
%END
*DEL SIA
%BEG
        %OID 2388
        %NAM "Year"
        %OWN 3 "Cinebase"/"conceptual-12"."FILM" 2408
        %PRV 6 "Cinebase"/"conceptual-12"."FILM"."Abstract" 2424
        %CAR 1-1
        %TYP N
        %SET S
        %LEN 4
        %DEC 0
```

```
        %MET "corresp" "" "VU"
        %MET "line" "397:100" "V"
%END
*DEL SIA
%BEG
        %OID 2416
        %NAM "Color"
        %OWN 3 "Cinebase"/"conceptual-12"."FILM" 2408
        %PRV 6 "Cinebase"/"conceptual-12"."FILM"."Abstract" 2424
        %CAR 0-1
        %TYP B
        %SET S
        %MET "corresp" "" "VU"
        %MET "line" "397:101" "V"
%END
*DEL SIA
%BEG
        %OID 2393
        %NAM "Technique"
        %OWN 3 "Cinebase"/"conceptual-12"."FILM" 2408
        %PRV 6 "Cinebase"/"conceptual-12"."FILM"."Abstract" 2424
        %CAR 0-1
        %TYP A
        %SET S
        %LEN 15
        %MET "corresp" "" "VU"
        %MET "line" "397:102" "V"
%END
*TRF split_merge
%BEG
        %NAM "MUSIC"
        %OWN 1 "Cinebase"/"conceptual-12" 2379
        %OID 2404
        %POX 31484
        %POY 103715
        %MET "corresp" "" "VU"
        %MET "line" "397:105" "V"
        *MOD ENT
        %BEG
                *OLD ENT
                %BEG
                        %OID 2404
                        %NAM "MUSIC"
                        %OWN 1 "Cinebase"/"conceptual-12" 2379
                %END
                %OID 2404
                %NAM "MUSIC"
                %OWN 1 "Cinebase"/"conceptual-12" 2379
        %END
        *MOD ENT
        %BEG
                *OLD ENT
                %BEG
                        %OID 2408
                        %NAM "FILM"
                        %OWN 1 "Cinebase"/"conceptual-12" 2379
                %END
                %OID 2408
                %NAM "FILM"
```

```
        %OWN 1 "Cinebase"/"conceptual-12" 2379
%END
*MOD REL
%BEG
        *OLD REL
        %BEG
                %OID 2407
                %NAM "from"
                %OWN 1 "Cinebase"/"conceptual-12" 2379
        %END
        %OID 2407
        %NAM "from"
        %OWN 1 "Cinebase"/"conceptual-12" 2379
%END
*MOD ROL
%BEG
        *OLD ROL
        %BEG
                %OID 2435
                %OWN 2 "Cinebase"/"conceptual-12"."from" 2407
                %ETR 3 "Cinebase"/"conceptual-12"."MUSIC" 2404
        %END
        %OID 2435
        %POX 48117
        %POY 95507
        %OWN 2 "Cinebase"/"conceptual-12"."from" 2407
        %ETR 3 "Cinebase"/"conceptual-12"."MUSIC" 2404
        %CAR 1-1
%END
*MOD ROL
%BEG
        *OLD ROL
        %BEG
                %OID 2433
                %OWN 2 "Cinebase"/"conceptual-12"."from" 2407
                %ETR 3 "Cinebase"/"conceptual-12"."FILM" 2408
        %END
        %OID 2433
        %POX 73193
        %POY 83137
        %OWN 2 "Cinebase"/"conceptual-12"."from" 2407
        %ETR 3 "Cinebase"/"conceptual-12"."FILM" 2408
        %CAR 1-1
%END
*MOD SIA
%BEG
        *OLD SIA
        %BEG
                %OID 2401
                %NAM "Number"
                %OWN 3 "Cinebase"/"conceptual-12"."MUSIC" 2404
        %END
        %OID 2401
        %NAM "Number"
        %OWN 3 "Cinebase"/"conceptual-12"."MUSIC" 2404
        %CAR 1-1
        %TYP N
        %SET S
        %LEN 5
```

```
            %DEC 0
            %MET "corresp" "" "VU"
            %MET "line" "397:106" "V"
        %END
        *MOD SIA
        %BEG
            *OLD SIA
            %BEG
                %OID 2389
                %NAM "Title"
                %OWN 3 "Cinebase"/"conceptual-12"."MUSIC" 2404
            %END
            %OID 2389
            %NAM "Title"
            %OWN 3 "Cinebase"/"conceptual-12"."MUSIC" 2404
            %PRV 6 "Cinebase"/"conceptual-12"."MUSIC"."Number" 2401
            %CAR 1-1
            %TYP A
            %SET S
            %LEN 50
            %MET "corresp" "" "VU"
            %MET "line" "397:107" "V"
        %END
        *MOD SIA
        %BEG
            *OLD SIA
            %BEG
                %OID 2419
                %NAM "CD"
                %OWN 3 "Cinebase"/"conceptual-12"."MUSIC" 2404
            %END
            %OID 2419
            %NAM "CD"
            %OWN 3 "Cinebase"/"conceptual-12"."MUSIC" 2404
            %PRV 6 "Cinebase"/"conceptual-12"."MUSIC"."Composer" 2415
            %CAR 0-1
            %TYP N
            %SET S
            %LEN 10
            %DEC 0
            %MET "corresp" "" "VU"
            %MET "line" "397:109" "V"
        %END
        &MOD SIA
        %BEG
            *OLD SIA
            %BEG
                %OID 2415
                %NAM "Composer"
                %OWN 3 "Cinebase"/"conceptual-12"."MUSIC" 2404
            %END
            %OID 2415
            %NAM "Composer"
        %END
%END
*DEL GRP
%BEG
        %OID 2405
        %NAM "IDMUSIQUE"
```

```
        %OWN 3 "Cinebase"/"conceptual-12"."MUSIC" 2404
        %COM 6 "Cinebase"/"conceptual-12"."MUSIC"."Number" 2401
        %TYP A
        %FLA "P"
        %MET "line" "397:110" "V"
        &MOD GRP
        %BEG
                *OLD GRP
                %BEG
                        %OID 2405
                        %NAM "IDMUSIQUE"
                        %OWN 3 "Cinebase"/"conceptual-12"."MUSIC" 2404
                %END
                %OID 2405
                %NAM "IDMUSIQUE"
        %END
%END
*DEL SIA
%BEG
        %OID 2419
        %NAM "CD"
        %OWN 3 "Cinebase"/"conceptual-12"."MUSIC" 2404
        %PRV 6 "Cinebase"/"conceptual-12"."MUSIC"."Title" 2389
        %CAR 0-1
        %TYP N
        %SET S
        %LEN 10
        %DEC 0
        %MET "corresp" "" "VU"
        %MET "line" "397:109" "V"
%END
*DEL SIA
%BEG
        %OID 2389
        %NAM "Title"
        %OWN 3 "Cinebase"/"conceptual-12"."MUSIC" 2404
        %PRV 6 "Cinebase"/"conceptual-12"."MUSIC"."Number" 2401
        %CAR 1-1
        %TYP A
        %SET S
        %LEN 50
        %MET "corresp" "" "VU"
        %MET "line" "397:107" "V"
%END
*DEL SIA
%BEG
        %OID 2401
        %NAM "Number"
        %OWN 3 "Cinebase"/"conceptual-12"."MUSIC" 2404
        %CAR 1-1
        %TYP N
        %SET S
        %LEN 5
        %DEC 0
        %MET "corresp" "" "VU"
        %MET "line" "397:106" "V"
%END
*DEL ROL
%BEG
```

```
        %OID 2433
        %POX 73193
        %POY 83137
        %OWN 2 "Cinebase"/"conceptual-12"."from" 2407
        %ETR 3 "Cinebase"/"conceptual-12"."FILM" 2408
        %CAR 1-1
        &MOD ROL
        %BEG
                *OLD ROL
                %BEG
                        %OID 2433
                        %OWN 2 "Cinebase"/"conceptual-12"."from" 2407
                        %ETR 3 "Cinebase"/"conceptual-12"."FILM" 2408
                %END
                %OID 2433
                %OWN 2 "Cinebase"/"conceptual-12"."from" 2407
        %END
%END
*DEL ROL
%BEG
        %OID 2435
        %POX 48117
        %POY 95507
        %OWN 2 "Cinebase"/"conceptual-12"."from" 2407
        %ETR 3 "Cinebase"/"conceptual-12"."MUSIC" 2404
        %CAR 1-1
        &MOD ROL
        %BEG
                *OLD ROL
                %BEG
                        %OID 2435
                        %OWN 2 "Cinebase"/"conceptual-12"."from" 2407
                        %ETR 3 "Cinebase"/"conceptual-12"."MUSIC" 2404
                %END
                %OID 2435
                %OWN 2 "Cinebase"/"conceptual-12"."from" 2407
        %END
%END
*DEL REL
%BEG
        %OID 2407
        %NAM "from"
        %POX 60656
        %POY 89322
        %OWN 1 "Cinebase"/"conceptual-12" 2379
        %MET "corresp" "" "VU"
%END
*DEL ENT
%BEG
        %OID 2404
        %NAM "MUSIC"
        %POX 31484
        %POY 103715
        %OWN 1 "Cinebase"/"conceptual-12" 2379
        %MET "corresp" "" "VU"
        %MET "line" "397:105" "V"
%END
*TRF desagre_att
%BEG
```

```
        %OWN 6 "Cinebase"/"conceptual-12"."DISTRIBUTOR"."Address" 2421
        &MOD SIA
        %BEG
                *OLD SIA
                %BEG
                        %OID 2395
                        %NAM "Street"
                        %OWN 6 "Cinebase"/"conceptual-12"."DISTRIBUTOR"."Address" 2421
                        %CAR 1-1
                %END
                %OID 2395
                %NAM "Street"
                %CAR 1-1
        %END
        &MOD SIA
        %BEG
                *OLD SIA
                %BEG
                        %OID 2387
                        %NAM "Zip-code"
                        %OWN 6 "Cinebase"/"conceptual-12"."DISTRIBUTOR"."Address" 2421
                        %CAR 1-1
                %END
                %OID 2387
                %NAM "Zip-code"
                %CAR 1-1
        %END
        &MOD SIA
        %BEG
                *OLD SIA
                %BEG
                        %OID 2417
                        %NAM "City"
                        %OWN 6 "Cinebase"/"conceptual-12"."DISTRIBUTOR"."Address" 2421
                        %CAR 1-1
                %END
                %OID 2417
                %NAM "City"
                %CAR 1-1
        %END
        &DEL COA
        %BEG
                %OID 2421
                %NAM "Address"
                %OWN 3 "Cinebase"/"conceptual-12"."DISTRIBUTOR" 2411
                %PRV 6 "Cinebase"/"conceptual-12"."DISTRIBUTOR"."Name" 2402
                %CAR 1-1
                %SET S
                %MET "corresp" "" "VU"
        %END
%END
*POT "end-file"
```

Appendix J : dbtodbconverter program

```
import java.io.*;
import java.lang.*;
import java.sql.*;
import java.util.Vector;
public class dbtodbconverter
{
public static void main (String args[])
        {
// Connection au serveur local
try {
Class.forName("jdbc.interDB.cql");
} catch (java.lang.ClassNotFoundException e) {
 System.err.print ("ClassNotFoundException: ");
 System.err.println (e.getMessage());
}
ResultSet rsDISTRIBUTOR = null;
ResultSet rsMOVIE = null;
ResultSet rsPRODUCER = null;
ResultSet rsDISTRIBUTORMOVIE = null;
ResultSet rsMOVIEDISTRIBUTOR = null;
ResultSet rsMOVIEPRODUCER = null;
ResultSet rsPRODUCERMOVIE = null;
try {
Connection con = DriverManager.getConnection("df","df","df");

// Extraction des données par TE
Statement stmt = con.createStatement();
rsDISTRIBUTOR = stmt.ExecuteQuery("SELECT DISTRIBUTOR.Name ,
DISTRIBUTOR.Address.Street , DISTRIBUTOR.Address.Zip-code , DISTRIBUTOR.Address.City
FROM DISTRIBUTOR ;");
rsMOVIE = stmt.ExecuteQuery("SELECT MOVIE.Title , MOVIE.Director , MOVIE.Actor ,
MOVIE.Duration , MOVIE.Abstract , MUSIC.Composer FROM MOVIE , MUSIC WHERE MUSIC
from MOVIE ;");
rsPRODUCER = stmt.ExecuteQuery("SELECT PRODUCER.Name FROM PRODUCER ;");
rsDISTRIBUTORMOVIE = stmt.ExecuteQuery("SELECT DISTRIBUTOR.Name , MOVIE.Title
FROM DISTRIBUTOR , MOVIE WHERE DISTRIBUTOR distributes MOVIE ;");
rsMOVIEPRODUCER = stmt.ExecuteQuery("SELECT MOVIE.Title , PRODUCER.Name FROM
MOVIE , PRODUCER WHERE MOVIE produces PRODUCER ;");
// deconnection au serveur local

con.close();

}
catch(SQLException ex) {
System.err.println("SQLException: " + ex.getMessage());}
// Connection au serveur local
try {
Class.forName("jdbc.interDB.cql");
} catch (java.lang.ClassNotFoundException e) {
```

```
 System.err.print ("ClassNotFoundException: ");
 System.err.println (e.getMessage());
}
try {
Connection con = DriverManager.getConnection("er","er","er");

stmt = con.createStatement();
String ident2 = new String ("");
String ident1 = new String ("");
String tmp = new String ("");
boolean bool;
int id2,id1;
PRODUCER PRODUCER1;
while (rsPRODUCER.next())
        {
        PRODUCER1=(PRODUCER)rsPRODUCER.getAll();
        stmt.ExecuteQuery("insert object PRODUCER1 into PRODUCER");
        }

bool = rsPRODUCER.First();
DISTRIBUTOR DISTRIBUTOR1;
while (rsDISTRIBUTOR.next())
        {
        DISTRIBUTOR1=(DISTRIBUTOR)rsDISTRIBUTOR.getAll();
        stmt.ExecuteQuery("insert object DISTRIBUTOR1 into DISTRIBUTOR");
        }

bool = rsDISTRIBUTOR.First();
MOVIE MOVIE1;
String MOVIE2;
while (rsMOVIEPRODUCER.next())
{
        ident1 =rsMOVIEPRODUCER.getString(MOVIE.Title);
        while (ident1.compareTo(rsMOVIE.getString(MOVIE.Title))!=0)
        {
        bool =rsMOVIE.next();
        }
        MOVIE1 =rsMOVIE.getAll();
        bool = rsMOVIE.First();
        ident2 =rsMOVIEPRODUCER.getString(PRODUCER.Name);
        while (ident2.compareTo(rsPRODUCER.getString(PRODUCER.Name))!=0)
        {
        bool =rsPRODUCER.next();
        }
        PRODUCER1 =rsPRODUCER.getAll();
        bool = rsPRODUCER.First();
        MOVIE2 = "INSERT OBJECT MOVIE1 INTO MOVIE LINKED WITH PRODUCER1 VIA
produces";
        while (ident1.compareTo(rsDISTRIBUTORMOVIE.getString(MOVIE.Title))!=0)
        {
        bool =rsDISTRIBUTORMOVIE.next();
        }
        ident2 =rsDISTRIBUTORMOVIE.getString(DISTRIBUTOR.Name);
        while (ident2.compareTo(rsDISTRIBUTOR.getString(DISTRIBUTOR.Name))!=0)
        {
        bool =rsDISTRIBUTOR.next();
        }
        DISTRIBUTOR1 =rsDISTRIBUTOR.getAll();
        bool = rsDISTRIBUTOR.First()
```

```
        MOVIE2.concat (" AND DISTRIBUTOR1 VIA distributes");
        bool = rsDISTRIBUTORMOVIE.First();
        stmt.executeQuery(MOVIE2);
}
// deconnection au serveur local

con.close();

}
catch(SQLException ex) {
System.err.println("SQLException: " + ex.getMessage());}
        }
}
```

Appendix K : DB-to-XML log file

```
*POT "begin-file"
*DEL GRP
%BEG
        %OID 2467
        %NAM "IDMUSIQUE"
        %OWN 3 "Cinebase"/"conceptual-13"."MUSIC" 2466
        %COM 6 "Cinebase"/"conceptual-13"."MUSIC"."Number" 2463
        %TYP A
        %FLA "P"
        %MET "line" "397:110" "V"
        &MOD GRP
        %BEG
          - -     *OLD GRP
                  %BEG
                          %OID 2467
                          %NAM "IDMUSIQUE"
                          %OWN 3 "Cinebase"/"conceptual-13"."MUSIC" 2466
                  %END
                  %OID 2467
                  %NAM "IDMUSIQUE"
        %END
%END
*DEL SIA
%BEG
        %OID 2481
        %NAM "CD"
        %OWN 3 "Cinebase"/"conceptual-13"."MUSIC" 2466
        %PRV 6 "Cinebase"/"conceptual-13"."MUSIC"."Composer" 2477
        %CAR 0-1
        %TYP N
        %SET S
        %LEN 10
        %DEC 0
        %MET "corresp" "" "VU"
        %MET "line" "397:109" "V"
%END
*DEL SIA
%BEG
        %OID 2477
        %NAM "Composer"
        %OWN 3 "Cinebase"/"conceptual-13"."MUSIC" 2466
        %PRV 6 "Cinebase"/"conceptual-13"."MUSIC"."Title" 2451
        %CAR 1-1
        %TYP A
        %SET S
        %LEN 20
        %MET "corresp" "" "VU"
        %MET "line" "397:108" "V"
%END
```

```
*DEL SIA
%BEG
        %OID 2451
        %NAM "Title"
        %OWN 3 "Cinebase"/"conceptual-13"."MUSIC" 2466
        %PRV 6 "Cinebase"/"conceptual-13"."MUSIC"."Number" 2463
        %CAR 1-1
        %TYP A
        %SET S
        %LEN 50
        %MET "corresp" "" "VU"
        %MET "line" "397:107" "V"
%END
*DEL SIA
%BEG
        %OID 2463
        %NAM "Number"
        %OWN 3 "Cinebase"/"conceptual-13"."MUSIC" 2466
        %CAR 1-1
        %TYP N
        %SET S
        %LEN 5
        %DEC 0
        %MET "corresp" "" "VU"
        %MET "line" "397:106" "V"
%END
*DEL ROL
%BEG
        %OID 2495
        %POX 73193
        %POY 83137
        %OWN 2 "Cinebase"/"conceptual-13"."from" 2469
        %ETR 3 "Cinebase"/"conceptual-13"."FILM" 2470
        %CAR 1-1
        &MOD ROL
        %BEG
                *OLD ROL
                %BEG
                        %OID 2495
                        %OWN 2 "Cinebase"/"conceptual-13"."from" 2469
                        %ETR 3 "Cinebase"/"conceptual-13"."FILM" 2470
                %END
                %OID 2495
                %OWN 2 "Cinebase"/"conceptual-13"."from" 2469
        %END
%END
*DEL ROL
%BEG
        %OID 2497
        %POX 48117
        %POY 95507
        %OWN 2 "Cinebase"/"conceptual-13"."from" 2469
        %ETR 3 "Cinebase"/"conceptual-13"."MUSIC" 2466
        %CAR 1-1
        &MOD ROL
        %BEG
                *OLD ROL
                %BEG
                        %OID 2497
```

```
                    %OWN 2 "Cinebase"/"conceptual-13"."from" 2469
                    %ETR 3 "Cinebase"/"conceptual-13"."MUSIC" 2466
            %END
            %OID 2497
            %OWN 2 "Cinebase"/"conceptual-13"."from" 2469
        %END
%END
*DEL REL
%BEG
        %OID 2469
        %NAM "from"
        %POX 60656
        %POY 89322
        %OWN 1 "Cinebase"/"conceptual-13" 2441
        %MET "corresp" "" "VU"
%END
*DEL ENT
%BEG
        %OID 2466
        %NAM "MUSIC"
        %POX 31484
        %POY 103715
        %OWN 1 "Cinebase"/"conceptual-13" 2441
        %MET "corresp" "" "VU"
        %MET "line" "397:105" "V"
%END
*DEL SIA
%BEG
        %OID 2455
        %NAM "Technique"
        %OWN 3 "Cinebase"/"conceptual-13"."FILM" 2470
        %PRV 6 "Cinebase"/"conceptual-13"."FILM"."Color" 2478
        %CAR 0-1
        %TYP A
        %SET S
        %LEN 15
        %MET "corresp" "" "VU"
        %MET "line" "397:102" "V"
%END
*DEL SIA
%BEG
        %OID 2478
        %NAM "Color"
        %OWN 3 "Cinebase"/"conceptual-13"."FILM" 2470
        %PRV 6 "Cinebase"/"conceptual-13"."FILM"."Year" 2450
        %CAR 0-1
        %TYP B
        %SET S
        %MET "corresp" "" "VU"
        %MET "line" "397:101" "V"
%END
*DEL SIA
%BEG
        %OID 2450
        %NAM "Year"
        %OWN 3 "Cinebase"/"conceptual-13"."FILM" 2470
        %PRV 6 "Cinebase"/"conceptual-13"."FILM"."Category" 2482
        %CAR 1-1
        %TYP N
```

```
        %SET S
        %LEN 4
        %DEC 0
        %MET "corresp" "" "VU"
        %MET "line" "397:100" "V"
%END
*DEL SIA
%BEG
        %OID 2482
        %NAM "Category"
        %OWN 3 "Cinebase"/"conceptual-13"."FILM" 2470
        %PRV 6 "Cinebase"/"conceptual-13"."FILM"."Abstract" 2486
        %CAR 1-1
        %TYP A
        %SET S
        %LEN 20
        %MET "corresp" "" "VU"
        %MET "line" "397:99" "V"
%END
*TRF et_to_att
%BEG
        %NAM "TELEPHONE"
        %OWN 1 "Cinebase"/"conceptual-13" 2441
        %OID 2453
        %POX 158342
        %POY 24604
        %MET "corresp" "" "VU"
        %MET "line" "" "V"
        *MOD SIA
        %BEG
                *OLD SIA
                %BEG
                        %OID 2462
                        %NAM "Number"
                        %OWN 3 "Cinebase"/"conceptual-13"."TELEPHONE" 2453
                        %OWN 3 "Cinebase"/"conceptual-13"."DISTRIBUTOR" 2473
                        %CAR 1-1
                        %SET S
                        %TYP N
                        %FLA "r"
                        %LEN 12
                        %DEC 0
                %END
                %OID 2462
                %NAM "Phone"
                %CAR 0-5
                %TYP N
                %SET S
                %LEN 12
                %DEC 0
                %FLA "r"
        %END
        &MOD GRP
        %BEG
                *OLD GRP
                %BEG
                        %OID 2454
                        %NAM "IDTELEPHONE"
                        %OWN 3 "Cinebase"/"conceptual-13"."TELEPHONE" 2453
```

```
                %FLA "P"
            %END
            %OID 2454
            %NAM "IDTELEPHONE"
            %OWN 3 "Cinebase"/"conceptual-13"."TELEPHONE" 2453
            %FLA "-"
        %END
        &DEL REL
        %BEG
            %OID 2468
            %NAM "has"
            %SNA "D_T"
            %POX 158870
            %POY 46763
            %OWN 1 "Cinebase"/"conceptual-13" 2441
            %MET "corresp" "" "VU"
            &DEL ROL
            %BEG
                %OID 2493
                %POX 159060
                %POY 54734
                %OWN 2 "Cinebase"/"conceptual-13"."has" 2468
                %ETR 3 "Cinebase"/"conceptual-13"."DISTRIBUTOR" 2473
                %CAR 0-5
                &MOD GRP
                %BEG
                    *OLD GRP
                    %BEG
                        %OID 2454
                        %NAM "IDTELEPHONE"
                        %OWN 3 "Cinebase"/"conceptual-13"."TELEPHONE" 2453
                        %COM 8 "Cinebase"/"conceptual-
13"."has"."DISTRIBUTOR" 2493
                        %COM 6 "Cinebase"/"conceptual-
13"."DISTRIBUTOR"."Phone" 2462
                    %END
                    %OID 2454
                    %NAM "IDTELEPHONE"
                    %OWN 3 "Cinebase"/"conceptual-13"."TELEPHONE" 2453
                    %COM 6 "Cinebase"/"conceptual-13"."DISTRIBUTOR"."Phone"
2462
                %END
                &MOD ROL
                %BEG
                    *OLD ROL
                    %BEG
                        %OID 2493
                        %OWN 2 "Cinebase"/"conceptual-13"."has" 2468
                        %ETR 3 "Cinebase"/"conceptual-13"."DISTRIBUTOR"
2473
                    %END
                    %OID 2493
                    %OWN 2 "Cinebase"/"conceptual-13"."has" 2468
                %END
            %END
            &DEL ROL
            %BEG
                %OID 2491
                %POX 158679
```

```
                %POY 38793
                %OWN 2 "Cinebase"/"conceptual-13"."has" 2468
                %ETR 3 "Cinebase"/"conceptual-13"."TELEPHONE" 2453
                %CAR 1-1
                &MOD ROL
                %BEG
                        *OLD ROL
                        %BEG
                                %OID 2491
                                %OWN 2 "Cinebase"/"conceptual-13"."has" 2468
                                %ETR 3 "Cinebase"/"conceptual-13"."TELEPHONE" 2453
                        %END
                        %OID 2491
                        %OWN 2 "Cinebase"/"conceptual-13"."has" 2468
                %END
        %END
    %END
    &DEL ENT
    %BEG
        %OID 2453
        %NAM "TELEPHONE"
        %POX 158342
        %POY 24604
        %OWN 1 "Cinebase"/"conceptual-13" 2441
        %MET "corresp" "" "VU"
        %MET "line" "" "V"
        &DEL GRP
        %BEG
                %OID 2454
                %NAM "IDTELEPHONE"
                %OWN 3 "Cinebase"/"conceptual-13"."TELEPHONE" 2453
                %COM 6 "Cinebase"/"conceptual-13"."DISTRIBUTOR"."Phone" 2462
                %TYP A
                %FLA "-"
                %MET "line" "" "V"
                &MOD GRP
                %BEG
                        *OLD GRP
                        %BEG
                                %OID 2454
                                %NAM "IDTELEPHONE"
                                %OWN 3 "Cinebase"/"conceptual-13"."TELEPHONE" 2453
                        %END
                        %OID 2454
                        %NAM "IDTELEPHONE"
                %END
        %END
    %END
%END
*MOD ENT
%BEG
    *OLD ENT
    %BEG
        %OID 2470
        %NAM "FILM"
        %OWN 1 "Cinebase"/"conceptual-13" 2441
    %END
    %OID 2470
    %NAM "MOVIE"
```

```
        %OWN 1 "Cinebase"/"conceptual-13" 2441
%END
*POT "end-file"
```

Appendix L : dbtoxmlconverter program

```
import java.io.*;
import java.lang.*;
import java.sql.*;
import java.util.Vector;
public class dbtoxmlconverter
{
//Declaration de la procédure pour écrire dans un fichier


public static void writln ( String file)
{
        try
        { - -
        FileOutputStream fos = new FileOutputStream (file,true);
        fos.write(10);
        }
        catch (IOException e)
        {
        System.err.println (e);
        }
}

// Procedure pour passer une ligne dans un fichier

public static void writ (String args, String file)
{
        try
        {
        FileOutputStream fos = new FileOutputStream (file,true);
        int length = args.length();
        char[] temp = new char[length+1];
        args.getChars (0,length,temp,0);
        int i = 0;
        byte[] b = new byte [length+1];
        while (length>=0)
                {
                b[i] = (byte) temp[i];
                i = i+1;
                length=length-1;
                }
        fos.write (b);
        }
        catch (IOException e)
        {
        System.err.println (e);
        }
}
```

```
public static void main (String args[])
        {
// Connection au serveur local
try {
Class.forName("jdbc.interDB.cql");
} catch (java.lang.ClassNotFoundException e) {
 System.err.print ("ClassNotFoundException: ");
 System.err.println (e.getMessage());
}
ResultSet rsDISTRIBUTOR = null;
ResultSet rsFILM = null;
ResultSet rsPRODUCER = null;
ResultSet rsDISTRIBUTORFILM = null;
ResultSet rsFILMDISTRIBUTOR = null;
ResultSet rsFILMPRODUCER = null;
ResultSet rsPRODUCERFILM = null;
try {
Connection con = DriverManager.getConnection("tof","fuque","zeworld");

// Extraction des données par TE
Statement stmt = con.createStatement();
ISTRIBUTOR = stmt.executeQuery("SELECT DISTRIBUTOR.Name  , DISTRIBUTOR.Address.Street
, DISTRIBUTOR.Address.Zip-code , DISTRIBUTOR.Address.City  , TELEPHONE.Number  FROM
DISTRIBUTOR , TELEPHONE  WHERE TELEPHONE has DISTRIBUTOR ;");
rsMOVIE = stmt.executeQuery("SELECT FILM.Title  , FILM.Director  , FILM.Screenwriter  ,
FILM.Actor  , FILM.Duration  , FILM.Abstract  FROM FILM ;");
rsPRODUCER = stmt.executeQuery("SELECT PRODUCER.Name  , PRODUCER.Address.Street ,
PRODUCER.Address.Zip-code , PRODUCER.Address.City  FROM PRODUCER;");
rsDISTRIBUTORMOVIE = stmt.executeQuery("SELECT DISTRIBUTOR.Name  , FILM.Title  FROM
DISTRIBUTOR , FILM  WHERE DISTRIBUTOR distributes FILM;");
rsMOVIEPRODUCER = stmt.executeQuery("SELECT FILM.Title  , PRODUCER.Name  FROM FILM
, PRODUCER WHERE FILM produces PRODUCER;");
// deconnection au serveur local

con.close();

}
catch(SQLException ex) {
System.err.println("SQLException: " + ex.getMessage());}

//insertion des données dans le fichier XML

String s = new String ("");
String valatt= new String ("");
String valid= new String ("");
String valid2= new String ("");
String validtemp= new String ("");
String sidref= new String ("");
String idx = new String ("");
String idx2 = new String ("");
Boolean b;
String fout = "Movie.xml";

//Initialisation du vecteur tidxml qui contient les identifiants XML(entre autre)

tabididref tidxml = new tabididref();
tidxml.addelem( "DISTRIBUTOR","DISTRIBUTOR.Name ",rsDISTRIBUTOR);
tidxml.addelem( "MOVIE","FILM.Title ",rsMOVIE);
```

```
tidxml.addelem( "PRODUCER","PRODUCER.Name ",rsPRODUCER);

//Ecriture de l'entete du fichier xml : reference au DTD

s = "<?xml version = \"1.0\"?>";
writ(s,fout);
writln(fout);
s = "<!DOCTYPE Movie SYSTEM \"Movie.dtd\">";
writ(s,fout);
writln(fout);
s = "<Movie>";
writ(s,fout);
writln(fout);

//Ecriture des entités

try {
b = rsDISTRIBUTOR.first();
while (rsDISTRIBUTOR.next())
        {
        valid = (String) rsDISTRIBUTOR.getString("DISTRIBUTOR.Name ");
        tidxml.getidxml("DISTRIBUTOR",valid,idx);
        // s= l'entite avec son identifiant
        s = "<DISTRIBUTOR id="+idx+" ";

        sidref = "";
        b = rsDISTRIBUTORMOVIE.first();

        // while qui donne la liste des idref
        while (rsDISTRIBUTORMOVIE.next())
                {
                valid2 = (String) rsDISTRIBUTORMOVIE.getString("MOVIE.Title ");
                validtemp = (String) rsDISTRIBUTORMOVIE.getString("DISTRIBUTOR.Name ");
                if (validtemp == valid)
                        {
                        tidxml.getidxml("MOVIE",valid2,idx2);
                        sidref=sidref.concat(" \""+idx2+"\" ");
                        }
                }
        s = s.concat("distributes.MOVIE = "+sidref);
        writ(s,fout);
        writ(">",fout);
        writln(fout);

        // ecriture de chaque attribut simple de l'entite

        valatt = (String) rsDISTRIBUTOR.getString("DISTRIBUTOR.Name ");
        s = "<DISTRIBUTOR.Name>"+valatt+"</DISTRIBUTOR.Name>";
        writ(s,fout);
        writln(fout);
        valatt = (String) rsDISTRIBUTOR.getString("TELEPHONE.Number");
        s = "<DISTRIBUTOR.Phone>"+valatt+"</DISTRIBUTOR.Phone>";
        writ(s,fout);
        writln(fout);

        // ecriture de chaque attribut composé de l'entite

        s ="<DISTRIBUTOR.Address>";
        writ(s,fout);
```

```
        valatt = (String) rsDISTRIBUTOR.getString("DISTRIBUTOR.Address.Street");
        s = "<DISTRIBUTOR.Address.Street>"+valatt+"</DISTRIBUTOR.Address.Street>";
        writ(s,fout);
        writln(fout);
        valatt = (String) rsDISTRIBUTOR.getString("DISTRIBUTOR.Address.Zip-code");
        s = "<DISTRIBUTOR.Address.Zip-code>"+valatt+"</DISTRIBUTOR.Address.Zip-code>";
        writ(s,fout);
        writln(fout);
        valatt = (String) rsDISTRIBUTOR.getString("DISTRIBUTOR.Address.City");
        s = "<DISTRIBUTOR.Address.City>"+valatt+"</DISTRIBUTOR.Address.City>";
        writ(s,fout);
        writln(fout);
        s ="</DISTRIBUTOR.Address>";
        writ(s,fout);
        writln(fout);
        s = "</DISTRIBUTOR>";
        writ(s,fout);
        writln(fout);
        }
b = rsMOVIE.first();
while (rsMOVIE.next())
        {
        valid = (String) rsMOVIE.getString("FILM.Title ");
        tidxml.getidxml("MOVIE",valid,idx);
        // s= l'entite avec son identifiant
        s = "<MOVIE id="+idx+" ";

        sidref = "";
        b = rsDISTRIBUTORMOVIE.first();

        // while qui donne la liste des idref
        while (rsDISTRIBUTORMOVIE.next())
                {
                valid2 = (String) rsDISTRIBUTORMOVIE.getString("DISTRIBUTOR.Name ");
                validtemp = (String) rsDISTRIBUTORMOVIE.getString("FILM.Title ");
                if (validtemp == valid)
                        {
                        tidxml.getidxml("DISTRIBUTOR",valid2,idx2);
                        sidref=sidref.concat(" \""+idx2+"\" ");
                        }
                }
        s = s.concat("distributes.DISTRIBUTOR = "+sidref);

        sidref = "";
        b = rsMOVIEPRODUCER.first();

        // while qui donne la liste des idref
        while (rsMOVIEPRODUCER.next())
                {
                valid2 = (String) rsMOVIEPRODUCER.getString("PRODUCER.Name ");
                validtemp = (String) rsMOVIEPRODUCER.getString("FILM.Title ");
                if (validtemp == valid)
                        {
                        tidxml.getidxml("PRODUCER",valid2,idx2);
                        sidref=sidref.concat(" \""+idx2+"\" ");
                        }
                }
        s = s.concat("produces.PRODUCER = "+sidref);
        writ(s,fout);
```

```
        writ(">",fout);
        writln(fout);

        // ecriture de chaque attribut simple de l'entite

        valatt = (String) rsMOVIE.getString("MOVIE.Title ");
        s = "<MOVIE.Title>"+valatt+"</MOVIE.Title>";
        writ(s,fout);
        writln(fout);
        valatt = (String) rsMOVIE.getString("MOVIE.Director ");
        s = "<MOVIE.Director>"+valatt+"</MOVIE.Director>";
        writ(s,fout);
        writln(fout);
        valatt = (String) rsMOVIE.getString("MOVIE.Screenwriter ");
        s = "<MOVIE.Screenwriter>"+valatt+"</MOVIE.Screenwriter>";
        writ(s,fout);
        writln(fout);
        valatt = (String) rsMOVIE.getString("MOVIE.Actor ");
        s = "<MOVIE.Actor>"+valatt+"</MOVIE.Actor>";
        writ(s,fout);
        writln(fout);
        valatt = (String) rsMOVIE.getString("MOVIE.Duration ");
        s = "<MOVIE.Duration>"+valatt+"</MOVIE.Duration>";
        writ(s,fout);
        writln(fout);
        valatt = (String) rsMOVIE.getString("MOVIE.Abstract ");
        s = "<MOVIE.Abstract>"+valatt+"</MOVIE.Abstract>";
        writ(s,fout);
        writln(fout);

        // ecriture de chaque attribut composé de l'entite

        s = "</MOVIE>";
        writ(s,fout);
        writln(fout);
        }
b = rsPRODUCER.first();
while (rsPRODUCER.next())
        {
        valid = (String) rsPRODUCER.getString("PRODUCER.Name ");
        tidxml.getidxml("PRODUCER",valid,idx);
        // s= l'entite avec son identifiant
        s = "<PRODUCER id="+idx+" ";

        sidref = "";
        b = rsMOVIEPRODUCER.first();

        // while qui donne la liste des idref
        while (rsMOVIEPRODUCER.next())
                {
                valid2 = (String) rsMOVIEPRODUCER.getString("MOVIE.Title ");
                validtemp = (String) rsMOVIEPRODUCER.getString("PRODUCER.Name ");
                if (validtemp == valid)
                        {
                        tidxml.getidxml("MOVIE",valid2,idx2);
                        sidref=sidref.concat(" \""+idx2+"\" ");
                        }
                }
        s = s.concat("produces.MOVIE = "+sidref);
```

```
        writ(s,fout);
        writ(">",fout);
        writln(fout);

        // ecriture de chaque attribut simple de l'entite

        valatt = (String) rsPRODUCER.getString("PRODUCER.Name ");
        s = "<PRODUCER.Name>"+valatt+"</PRODUCER.Name>";
        writ(s,fout);
        writln(fout);

        // ecriture de chaque attribut composé de l'entite

        s ="<PRODUCER.Address>";
        writ(s,fout);
        valatt = (String) rsPRODUCER.getString("PRODUCER.Address.Street");
        s = "<PRODUCER.Address.Street>"+valatt+"</PRODUCER.Address.Street>";
        writ(s,fout);
        writln(fout);
        valatt = (String) rsPRODUCER.getString("PRODUCER.Address.Zip-code");
        s = "<PRODUCER.Address.Zip-code>"+valatt+"</PRODUCER.Address.Zip-code>";
        writ(s,fout);
        writln(fout);
        valatt = (String) rsPRODUCER.getString("PRODUCER.Address.City");
        s = "<PRODUCER.Address.City>"+valatt+"</PRODUCER.Address.City>";
        writ(s,fout);
        writln(fout);
        s ="</PRODUCER.Address>";
        writ(s,fout);
        writln(fout);
        s = "</PRODUCER>";
        writ(s,fout);
        writln(fout);
        }
}
catch (Exception e) { }
s = "</Movie>";
writ(s,fout);
writln(fout);
        }
}
/* Class qui contient le nom de l'entité, un vecteur avec les valeurs de son identifiant, un vecteur avec la
valeur de l'identifiant xml*/
class ididref
        {
        public String entite;
        public Vector idDB;
        public Vector idxml;


//constructor

        public ididref ()
                {
                idDB = new Vector ();
                idxml = new Vector ();
                }

        public ididref (String e, Vector d, Vector x)
```

```
                {
                this();
                entite = e;
                idDB = d;
                idxml = x;
                }

//methodes

        public Vector getidDB ()
                {
                return idDB;
                }

        public Vector getidx ()
                {
                return idxml;
                }

        public String getentite ()
                {
                return entite;
                }
        } - -

//Class qui est un vecteur de  ididref

class tabididref
        {
        public Vector tid;

// constructor

        public tabididref ()
                {
                tid = new Vector ();
                }

//methode pour ajouter un element dans le vecteur

        public void addelem (String ent, String query, ResultSet rs)
                {
                int i=0;
                int cpt=0;
                boolean b;

                //Recherche du nombre d'identifiants XMLdeja attribues

                while (i < tid.size())
                        {
                        cpt = cpt + (((ididref)tid.elementAt(i)).getidx()).size();
                        i=i+1;
                        }

                //création de l'ididref à insérer dans tid

                String temp = new String ();
                b=rs.first();
                Vector vid = new Vector ();
```

```
Vector vidx = new Vector ();

//initialisation de vid et vidx

while (rs.next())
        {
        temp = (String) rs.getString(query);
        vid.addElement(temp);
        cpt = cpt +1;
        vidx.addElement( new Integer (cpt).toString());
        }

// creation de idf
ididref idf = new ididref (ent, vid, vidx);

// ajout de idf dans tid
tid.addElement(idf);
}
// methode pour rechercher l'identifiant xml qui correspond à l'entite

public void getidxml(String ent, String ident, String idx)
{
int cmpt=0;
int cmpt2=0;
while (ent.compareTo(((ididref)tid.elementAt(cmpt)).getentite())!=0)
        {
        cmpt=cmpt+1;
        }
while
(ident.compareTo((String)((Vector)(((ididref)tid.elementAt(cmpt)).getidDB())).elementAt(cmpt2))!=0)
        {
        cmpt2=cmpt2+1;
        }
idx = (String)((Vector)((ididref)tid.elementAt(cmpt)).getidx()).elementAt(cmpt2);
}

}
```

Appendix M : XML document

The XML document created by the dbtoxmlconverter Java program in the case study is given below.


```
<? XML version ="1.0"?>
<!DOCTYPE Movie SYSTEM  "Movie.dtd">

<DISTRIBUTOR id = "1" distributes.MOVIE = "5">
        <DISTRIBUTOR.Name> 20th Century Fox </DISTRIBUTOR.Name>
        <DISTRIBUTOR.Address>
                <DISTRIBUTOR.Address.Street>Hollywood Blvd, 150
                        </DISTRIBUTOR.Address.Street>
                <DISTRIBUTOR.Address.Zip-code>9000
                        </DISTRIBUTOR.Address.Zip-code>
                <DISTRIBUTOR.Address.City> Los Angeles
                        </DISTRIBUTOR.Address.City>
        </DISTRIBUTOR.Address>
        <DISTRIBUTOR.Phone>1230450890</DISTRIBUTOR.Phone>
        <DISTRIBUTOR.Phone>1230450891</DISTRIBUTOR.Phone>
</DISTRIBUTOR>

<DISTRIBUTOR id = "2" distributes.MOVIE = "6" "8" "12">
        <DISTRIBUTOR.Name> Universal</DISTRIBUTOR.Name>
        <DISTRIBUTOR.Address>
                <DISTRIBUTOR.Address.Street>Hollywood Blvd, 151
                        </DISTRIBUTOR.Address.Street>
                <DISTRIBUTOR.Address.Zip-code>9000
                        </DISTRIBUTOR.Address.Zip-code>
                <DISTRIBUTOR.Address.City> Los Angeles
                        </DISTRIBUTOR.Address.City>
        </DISTRIBUTOR.Address>
        <DISTRIBUTOR.Phone>1230756452</DISTRIBUTOR.Phone>
        <DISTRIBUTOR.Phone>1230756453</DISTRIBUTOR.Phone>
</DISTRIBUTOR>

<DISTRIBUTOR id = "3" distributes.MOVIE = "7" "9">
        <DISTRIBUTOR.Name> Warner</DISTRIBUTOR.Name>
        <DISTRIBUTOR.Address>
                <DISTRIBUTOR.Address.Street>Hollywood Blvd, 152
                        </DISTRIBUTOR.Address.Street>
                <DISTRIBUTOR.Address.Zip-code>9000
                        </DISTRIBUTOR.Address.Zip-code>
```

```
            <DISTRIBUTOR.Address.City> Los Angeles
                </DISTRIBUTOR.Address.City>
        </DISTRIBUTOR.Address>
        <DISTRIBUTOR.Phone>1230085236</DISTRIBUTOR.Phone>
        <DISTRIBUTOR.Phone>1230085237</DISTRIBUTOR.Phone>
        <DISTRIBUTOR.Phone>1230085238</DISTRIBUTOR.Phone>
</DISTRIBUTOR>

<DISTRIBUTOR id = "4" distributes.MOVIE = "10" "11">
        <DISTRIBUTOR.Name> Buena Vista</DISTRIBUTOR.Name>
        <DISTRIBUTOR.Address>
            <DISTRIBUTOR.Address.Street>Madison Avenue, 15
                </DISTRIBUTOR.Address.Street>
            <DISTRIBUTOR.Address.Zip-code>8000
                </DISTRIBUTOR.Address.Zip-code>
            <DISTRIBUTOR.Address.City> New York
                </DISTRIBUTOR.Address.City>
        </DISTRIBUTOR.Address>
        <DISTRIBUTOR.Phone> 1500365000</DISTRIBUTOR.Phone>
        <DISTRIBUTOR.Phone>1500365001</DISTRIBUTOR.Phone>
        <DISTRIBUTOR.Phone>1500365002</DISTRIBUTOR.Phone>
</DISTRIBUTOR>

<MOVIE id ="5" produces.PRODUCER = "13" distributes.DISTRIBUTOR = "1">
        <MOVIE.Title>Titanic </MOVIE.Title>
        <MOVIE.Director>Cameron </MOVIE.Director>
        <MOVIE.Screenwriter>Cameron </MOVIE.Screenwriter>
        <MOVIE.Actor>Di Caprio </MOVIE.Actor>
        <MOVIE.Duration>195 </MOVIE.Duration>
        <MOVIE.Abstract>The famous boat sinks </MOVIE.Abstract>
</MOVIE>

<MOVIE id ="6" produces.PRODUCER = "14" distributes.DISTRIBUTOR = "2">
        <MOVIE.Title>E.T. </MOVIE.Title>
        <MOVIE.Director>Spielberg </MOVIE.Director>
        <MOVIE.Screenwriter>Daviau </MOVIE.Screenwriter>
        <MOVIE.Actor>Barrymore </MOVIE.Actor>
        <MOVIE.Duration>115 </MOVIE.Duration>
        <MOVIE.Abstract>A kind E.T. on earth </MOVIE.Abstract>
</MOVIE>

<MOVIE id ="7" produces.PRODUCER = "15" distributes.DISTRIBUTOR = "3">
        <MOVIE.Title>The brides of Madison County </MOVIE.Title>
        <MOVIE.Director>Eastwood </MOVIE.Director>
        <MOVIE.Screenwriter>LaGravenese </MOVIE.Screenwriter>
        <MOVIE.Actor>Eastwood </MOVIE.Actor>
        <MOVIE.Duration>135 </MOVIE.Duration>
        <MOVIE.Abstract>four days for an eternal love </MOVIE.Abstract>
</MOVIE>
```

```
<MOVIE id ="8" produces.PRODUCER = "14" distributes.DISTRIBUTOR = "2">
        <MOVIE.Title>Jurassic Park </MOVIE.Title>
        <MOVIE.Director>Spielberg </MOVIE.Director>
        <MOVIE.Screenwriter>Crichton </MOVIE.Screenwriter>
        <MOVIE.Actor>Goldblum </MOVIE.Actor>
        <MOVIE.Duration>126 </MOVIE.Duration>
        <MOVIE.Abstract>Dinosaurs are alive </MOVIE.Abstract>
</MOVIE>

<MOVIE id ="9" produces.PRODUCER = "16" distributes.DISTRIBUTOR = "3">
        <MOVIE.Title>Mars Attacks </MOVIE.Title>
        <MOVIE.Director>Burton </MOVIE.Director>
        <MOVIE.Screenwriter>Gems </MOVIE.Screenwriter>
        <MOVIE.Actor>Nicholson </MOVIE.Actor>
        <MOVIE.Duration>103 </MOVIE.Duration>
        <MOVIE.Abstract>Aliens attacks earth </MOVIE.Abstract>
</MOVIE>

<MOVIE id ="10" produces.PRODUCER = "17" distributes.DISTRIBUTOR = "4">
        <MOVIE.Title>The nightmare before Christmas </MOVIE.Title>
        <MOVIE.Director>Burton </MOVIE.Director>
        <MOVIE.Screenwriter>McDowell </MOVIE.Screenwriter>
        <MOVIE.Duration>74 </MOVIE.Duration>
        <MOVIE.Abstract>Christmas tale </MOVIE.Abstract>
</MOVIE>

<MOVIE id ="11" produces.PRODUCER = "19" distributes.DISTRIBUTOR = "4">
        <MOVIE.Title>Aladdin </MOVIE.Title>
        <MOVIE.Director>Clements </MOVIE.Director>
        <MOVIE.Screenwriter>Clements </MOVIE.Screenwriter>
        <MOVIE.Duration>90 </MOVIE.Duration>
        <MOVIE.Abstract>Aladdin has some problems </MOVIE.Abstract>
</MOVIE>

<MOVIE id ="12" produces.PRODUCER = "20" distributes.DISTRIBUTOR = "2">
        <MOVIE.Title>Trois couleurs : bleu </MOVIE.Title>
        <MOVIE.Director>Kieslowski </MOVIE.Director>
        <MOVIE.Screenwriter> Kieslowski </MOVIE.Screenwriter>
        <MOVIE.Actor>Binoche</MOVIE.Actor>
        <MOVIE.Duration>98 </MOVIE.Duration>
        <MOVIE.Abstract>Fist part of the trilogy </MOVIE.Abstract>
</MOVIE>

</PRODUCER id="13" produces.MOVIE = "5">
        <PRODUCER.Name>Cameron</PRODUCER.Name>
        <PRODUCER.Address>
                <PRODUCER.Address.Street>rue de Bruxelles, 22
                        </PRODUCER.Address.Street>
                <PRODUCER.Address.Zip-code>5000 </PRODUCER.Address.Zip-
                        code>
```

```
            <PRODUCER.Address.City>Namur </PRODUCER.Address.City>
        </PRODUCER.Address>
</PRODUCER>

</PRODUCER id="14" produces.MOVIE = "6" "8">
        <PRODUCER.Name>Spielberg</PRODUCER.Name>
        <PRODUCER.Address>
            <PRODUCER.Address.Street>chaussée de Nivelles, 56
                </PRODUCER.Address.Street>
            <PRODUCER.Address.Zip-code>5140 </PRODUCER.Address.Zip-
                code>
            <PRODUCER.Address.City>Sombreffe </PRODUCER.Address.City>
        </PRODUCER.Address>
</PRODUCER>

</PRODUCER id="15" produces.MOVIE = "7">
        <PRODUCER.Name>Eastwood</PRODUCER.Name>
        <PRODUCER.Address>
            <PRODUCER.Address.Street>rue du Condroz
                </PRODUCER.Address.Street>
            <PRODUCER.Address.Zip-code>5590 </PRODUCER.Address.Zip-
                code>
            <PRODUCER.Address.City>Ciney </PRODUCER.Address.City>
        </PRODUCER.Address>
</PRODUCER>

</PRODUCER id="16" produces.MOVIE = "9">
        <PRODUCER.Name>Burton</PRODUCER.Name>
        <PRODUCER.Address>
            <PRODUCER.Address.Street>station, 17
                </PRODUCER.Address.Street>
            <PRODUCER.Address.Zip-code>5575 </PRODUCER.Address.Zip-
                code>
            <PRODUCER.Address.City>Gedinne </PRODUCER.Address.City>
        </PRODUCER.Address>
</PRODUCER>

</PRODUCER id="17" produces.MOVIE = "10">
        <PRODUCER.Name>Di Novi</PRODUCER.Name>
        <PRODUCER.Address>
            <PRODUCER.Address.Street>Avenue du Diamant, 96
                </PRODUCER.Address.Street>
            <PRODUCER.Address.Zip-code>1030 </PRODUCER.Address.Zip-
                code>
            <PRODUCER.Address.City>Bruxelles </PRODUCER.Address.City>
        </PRODUCER.Address>
</PRODUCER>

</PRODUCER id="18" >
        <PRODUCER.Name>Godfroid</PRODUCER.Name>
        <PRODUCER.Address>
```

```
            <PRODUCER.Address.Street>rue du Chenois, 186
                </PRODUCER.Address.Street>
            <PRODUCER.Address.Zip-code>6000 </PRODUCER.Address.Zip-
                code>
            <PRODUCER.Address.City>Charleroi </PRODUCER.Address.City>
        </PRODUCER.Address>
</PRODUCER>

</PRODUCER id="19" produces.MOVIE = "11">
        <PRODUCER.Name>Clements</PRODUCER.Name>
        <PRODUCER.Address>
            <PRODUCER.Address.Street>Square de Quinaux, 19
                </PRODUCER.Address.Street>
            <PRODUCER.Address.Zip-code>5100 </PRODUCER.Address.Zip-
                code>
            <PRODUCER.Address.City>Wierde </PRODUCER.Address.City>
        </PRODUCER.Address>
</PRODUCER>

</PRODUCER id="20" produces.MOVIE = "12">
        <PRODUCER.Name>Karmitz</PRODUCER.Name>
        <PRODUCER.Address>
            <PRODUCER.Address.Street>place franco-belge, 5
                </PRODUCER.Address.Street>
            <PRODUCER.Address.Zip-code>6200 </PRODUCER.Address.Zip-
                code>
            <PRODUCER.Address.City>Chatelet </PRODUCER.Address.City>
        </PRODUCER.Address>
</PRODUCER>

</PRODUCER id="21">
        <PRODUCER.Name>Besson</PRODUCER.Name>
        <PRODUCER.Address>
            <PRODUCER.Address.Street>Avenue Bois l'Evêque, 33
                </PRODUCER.Address.Street>
            <PRODUCER.Address.Zip-code>5100 </PRODUCER.Address.Zip-
                code>
            <PRODUCER.Address.City>Wierde </PRODUCER.Address.City>
        </PRODUCER.Address>
</PRODUCER>
```