Modeling information sources for information integration[1]

François Goasdoué, Chantal Reynaud

LRI – Univ. De Paris-Sud – Bât. 490 – 91 405 Orsay cedex – France
{goasdoue, cr}@lri.fr
Tel: 33 (0)1 69 15 66 45 – 33 (0)1 69 15 56 48
Fax: 33 (0)1 69 15 65 86

Abstract: The aim of this paper is to present an approach and automated tools for designing knowledge bases describing the contents of information sources in PICSEL[2] knowledge-based mediators. We address two problems: the abstraction problem and the representation problem, when information sources are relational databases. In a first part, we present an architectural overview of the PICSEL mediator showing its main knowledge components. Then, we describe our approach and tools that we have implemented (1) to identify, by means of an abstraction process, the main relevant concepts, called *semantic concepts*, in an Entity Relationship model and (2) to help representing these concepts using CARIN, a logical language combining description logics and Datalog Rules, and using specific terms in the application domain model.

Key words: information sources modeling, identification of concepts, entity-relationship model, description logics, knowledge abstraction, support to knowledge representation.

Introduction

Our research works are developed in the context of the PICSEL project [Rousset & al., 98]. The aim is to build information servers over existing information sources that are distributed and possibly heterogeneous. The approach which has been chosen in PICSEL is to define an information server as a *knowledge-based* mediator between users and several information sources relative to a *same application domain*.

The idea in the knowledge-based mediator approach is to manage multiple heterogeneous information sources thanks to knowledge bases describing their contents in a logical formalism and using the same vocabulary. This provides shared access to multiple data and preserves the autonomy of each information source. The mediator plays the role of an interface between the user and the sources giving the illusion of querying a centralized and homogeneous system. The aim of this paper is to present an approach and automated tools for designing the knowledge bases (KB) describing the contents of information sources in PICSEL knowledge-based mediators.

Designing such KB refers first to an abstraction problem, second to a representation problem. The aim of the abstraction problem is to define concepts which capture abstractions in information sources, usable to describe their contents. The representation problem which arises is the choice of the representation language and also the problem to represent knowledge with a formal language which may be difficult to use and with specific terms.

[1] Submitted to EKAW'99, 11th European Workshop on Knowledge Acquisition, Modeling and Management, Dagstuhl Castle, Germany, May 26-29 1999.
[2] Granted by CNET (Centre National d'Études des Télécommunications) under contract number 97 1B 378.

In the paper, we address these two problems when information sources are relational databases (DB). Our approach to solve the abstraction problem is based on Entity Relationship (ER) models used to model the schema of database applications, a technique which has been proven to be very effective for database design. ER models are interesting because they are abstract representations of data. Yet, they are flat models with all concepts at the same level. Moreover, ER models are built according to modeling rules and don't necessarily represent concepts relevant for users of databases. We need abstraction mechanisms to make sets of objects really perceptible and relevant to users emerge. In our approach, we propose *automated techniques* to identify the main relevant concepts, called *semantic concepts*, in ER models. These techniques are based on the mechanism of aggregation to create higher level concepts from primitive ones.

Once the concepts to be described in the knowledge base of the mediator are identified, the problem is then to write their description. Statements in the PICSEL mediator knowledge bases must be represented in CARIN [Levy & al, 98], a logical language combining description logics and Datalog rules. Moreover, the descriptions of the contents of all databases must be represented using the same vocabulary. Only terms of the application domain represented in the domain model of the PICSEL mediator are usable. So, we must obtain descriptions of the contents of a database, represented in CARIN and using terms in the domain model, from descriptions represented with the ER modeling language and using terms particular to a database. The problem is therefore to obtain a matching between semantically equivalent concepts represented with different terms and different formalisms. The idea to solve the representation problem is to exploit capabilities of database administrators (DBA). Administrators know the contents of the databases they manage, they exactly know the meaning of their conceptual schema. In our approach, each administrator will have to design the knowledge base referring to its own database. The identification of semantic concepts allows to organize the description of a whole conceptual schema of a database, such a schema being sometimes enormous. We guide then the DBA in the description of each semantic concept i.e. we have implemented automated tools to help them (1) to understand the meaning of the vocabulary of the domain model, (2) to write CARIN sentences, (3) to characterize concepts represented in a database in comparison with those represented in the domain model and using terms in the domain model.

The paper is organized as follows. In a first part, we present an architectural overview of the PICSEL mediator showing its main knowledge components. In a second part, we present the notion of semantic concept. Section 3 deals with their identification in an ER model and section 4 describes automated techniques to help DBA to describe semantic concepts in CARIN.

I. Architectural Overview

I.1. General presentation

In PICSEL, a mediator has been designed according to a knowledge-based approach. It has two main parts: a generic query engine and knowledge bases specific to information servers. The knowledge bases contain both the model of the application domain of a server and abstract descriptions of the contents of the information sources accessible from this server. Given an information server, there are one KB to model the domain and one KB per information source to describe its contents as shown on figure 1. The domain model contains all the basic vocabulary used to ask queries. The query engine takes in charge the access to

the sources in order to obtain the answers to user queries. Abstract descriptions of the contents of the information sources help it to localize relevant sources. They are represented in the same logical formalism as the user queries and as the sentences in the domain model. Wrappers are specialized modules in respect to data models. When the source is a relational DB, wrappers take in charge the translation of a query expressed in terms of source relations to relational form.

We give a description of the main knowledge components in a PICSEL mediator in the following figure.

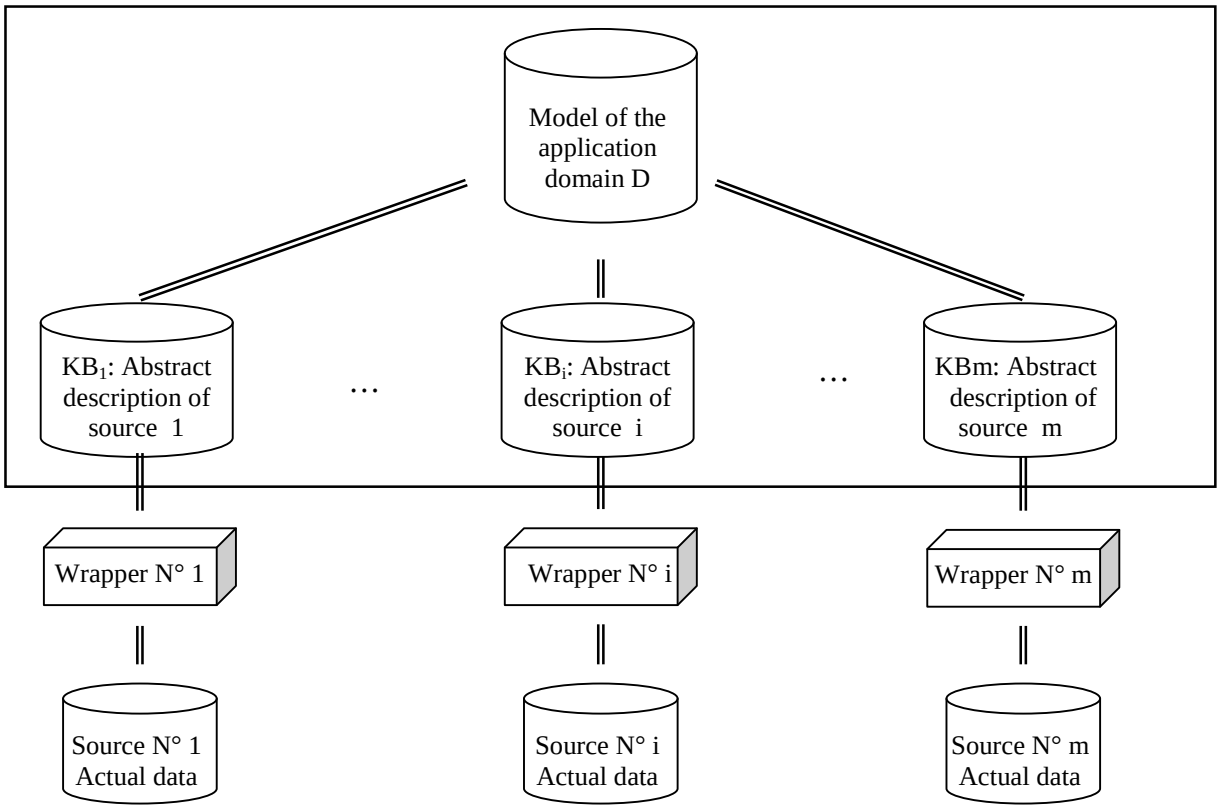

Figure 1: The knowledge base part of the PICSEL mediator in an information server dedicated to the application domain D

I.2. The main knowledge components in a PICSEL mediator

I.2.1. The domain model

The domain model contains all the vocabulary of an application domain used to ask queries. All the categories of objects that may be considered by users of the information server have to be represented. The domain model can be seen as a categorization of domain objects from a user-oriented point of view.

The domain model is represented in CARIN [Levy & al., 98], a logical language combining description logics[3] and Datalog rules. It is a formal language. Its semantics ensures that its

[3] The DL language that we consider in the PICSEL project is referred to as core-CLASSIC. It contains the constructors $\wedge$(conjunction), $\forall$(concept restriction), ($\geq$ n R), ($\leq$ n R)(number restrictions) and $\neg$(negation on basic concepts only).

exploitation at the symbol level by the engine conforms to its meaning at the knowledge level.

A domain model is built as follows. First, a basic vocabulary in terms of base predicates is acquired. New domain relations, significant for the application domain, can then be defined over the base relations using CARIN, either by rules or by concept expressions. Basic and complex relations constitute a taxinomic hierarchy that can be automatically constructed.

For example, the hierarchy represented in figure 2 is computed from the following expressions:
Product $\subseteq$ *(= 1 DepartureDate[4]) and (= 1 ArrivalDate),*
Journey := Product $\wedge$ *(= 1 DeparturePlace)* $\wedge$ *(= 1 ArrivalPlace)* $\wedge$ *(= 1 MeansTransport),*
Stay := Product $\wedge$ *(= 1 AssBuilding),*
Flight := Journey $\wedge$ *(*$\forall$*MeansTransport.Plane),*
TourismFlight := Flight and ($\forall$*MeansTransport.(*$\neg$*SupersonicPlane)),*
VIPFlight := Flight and ($\forall$*MeansTransport.(*$\neg$*TourismPlane)) and (*$\geq$ *1 AssociatedMeal).*

These sentences define the concepts *Journey*, *Stay*, *Flight*, *TourismFlight* and *VIPFlight* from the primitive concept *Product* (base predicate, unary relation) and from the primitive roles (binary relations) *DeparturePlace*, *ArrivalPlace*, *MeansTransport*. The concept *Product* is at least characterized by a single departure date and a single arrival date. The concept *Journey* is defined as a set of products that have exactly one departure place, one arrival place and one means of transport. The concept *Stay* is defined as a set of products that have exactly one associated building. The concept *Flight* is defined as a set of journeys which means of transport are necessarily planes. The concept *TourismFlight* is defined as a set of flights which means of transport are not supersonic planes whereas the concept *VIPFlight* is a set of flights which means of transport are not tourism planes and which have at least one associated meal.

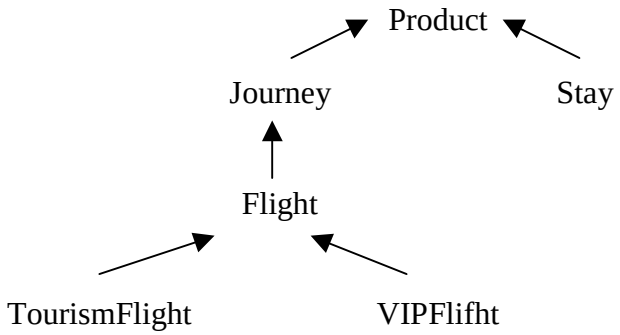

Figure 2: A taxinomic hierarchy

Our work doesn't focus on the design of the domain model. We have considered that the domain model was already built.

I.2.2. Abstract descriptions of the contents of a source

The abstract descriptions of a source consists of a set of source relations Vs1, Vs2, ..., Vsn for which it is specified: (1) a one-to-one mapping with domain relations, (2) a set of constraints that are used to characterize the instances of the domain relations that can be found in a

[4] To simplify, we use the syntax '*(= n R)*' for '*(*$\geq$ *n R) and (*$\leq$ *n R)*'.

source S. For example, for a given source S, the descriptions may say that we can find instances of *housing places* and the constraints may indicate that the housing places that we can find in S are all located in France.

More precisely, each abstract description of a relational database is a knowledge base that contains two sets of assertions: I_S and C_S. I_S represents mappings with domain relations by logical implications.

Example:
$v_{S1}(x) \Rightarrow$ *HousingPlace*(x), $v_{S2}(x,y) \Rightarrow$ *Located*(x,y) are two elements of Is if the source S contains instances of housing places with their location, *HousingPlace* and *Located* being two relations in the domain model.

C_S indicates the constraints that are known to hold on the database relations. They are represented with core-CLASSIC inclusions or incompatibility rules.

Example:
Let us consider that all housing places in S are located in Europe, but not in Germany. This can be described thanks to (1) the inclusion statement: $v_{S1} \subseteq (\forall v_{S2}.$*Europe*) and (2) the incompatibility rule: $v_{S1}(x) \wedge v_{S2}(x,y) \wedge$ *Germany*$(y) \Rightarrow \bot$.

I.3. The information sources (relational databases) accessible from an information server

Relational DB applications play an important role today. So, in this paper, we focus on information sources which are relational DB. Relational DB developments are usually decomposed in several steps. One main step is the construction of a conceptual model. The aim is to facilitate the communication between designers and end users by providing them with a conceptual representation of an application that does not include many of the details of how the data is physically stored. One of the most popular and prominent conceptual model is the Entity Relationship (ER) model introduced by Chen [Chen 76]. An ER model indicates the classes which objects can be found in the database. They are abstract representations of data. Instances useful in an application are grouped into classes or concepts and ER models represent classes rather than actual instances.

The approach that we propose to model databases relies on the analysis of ER models. It does not exploit the data of the database at all. Given a query, the aim is to identify relevant information sources which *may* give an answer. It is not to identify the sources which, at the moment, given their data, are able to give an answer to the query. Yet, an ER model doesn't provide a conceptualization adequate to the description of the contents of a database in PICSEL mediator. ER models are quite flat. All concepts are represented at the same level. Moreover the construction of an ER model is guided by modelisation rules and the concepts that are represented are not necessarily relevant for a user of the database application. We need abstraction mechanisms to make concepts really perceptible and relevant to users emerge. (cf. section III)

The basic primitives of an ER model are: entities, relationships, attributes. The classes of objects that are contained in the database are called entities and the objects within an entity are called instances. The language allows relationships between instances of differents entities to be represented. Each relationship has a name which is used to describe, in a litteral

form, links between instances. Attributes describe characteristics of instances of an entity or characteristics of related instances.

Multiplicity constraints on the relationships are given. A multiplicity constraint describes a restriction on the minimum and maximum number of instances from an entity that may be associated with any one instance from the other entity. In an ER Model, multiplicity constraints on the relationships define an application from the cardinal product ExR (E being the set of entities and R the set of relationships) to a set of cardinalities. This gives rise to different forms of binary relationships: one-to-one, one-to-many or many-to-many depending on whether the maximum number of the two pairs of cardinality corresponding to a binary relationship is 1, or only one of them, or none of them.

Furthermore, our approach needs to characterize the relationships of an ER Model according to the minimum number of the cardinalities. We will speak abouk a *weak* (resp. *strong*) relationship according to a given entity e if the minimum number of the pair of cardinality of (e,r) is 0 (resp. 1).

II. The notion of semantic concepts

In order to model relevant concepts, we defined the notion of *semantic concept*. This notion has been adapted from the *natural object* model used in the Dialog module of the CASE TRAMIS [Brès, 93]. One of the aim of this model is to aggregate entities and relationships of an ER model to allow objects really perceptible to the users to emerge.

A semantic concept (SC) can be seen as a grouping of entities and relationships. Such a grouping brings to the fore one particular entity, called the *root entity* of the SC, while the other ones only characterize it.

For example, in the ER model in figure 3, only two objects are perceptible to users: Regions (including data on their departments and data on the towns of these departments) and Regional settings (including guided tours). That means that, in the context of the database, regions are not meaningful for a user without information on their departments and on their towns. A department is not perceptible to users apart from regions. Any department always belongs to only one region (the pair of cardinalities of (department, belongs-1) is (1,1)). Data on a department can then be seen as a characterization of the region to which it belongs. Furthermore, in the database (always according to the ER model below), some towns are not close to any setting and conversely, some setting are not close to any town. That means that the concept of town and setting are not dependent of each other.

We shall say that an entity e', related to another entity e by the relationship r, characterizes e if the maximum number of the pair of cardinalities of (e',r) is 1. Given such a definition, we studied, in our work, all the ways an entity e' may be related to another one e depending on the numbers of the cardinalities of (e', r) and (e, r), r being a relationship relating e and e'. This led us to define several degrees of characterization of a relationship: *none* < *weak* < *strong* < *pairable*. In the following, relationships are similar to weak, strong or pairable characterization links.

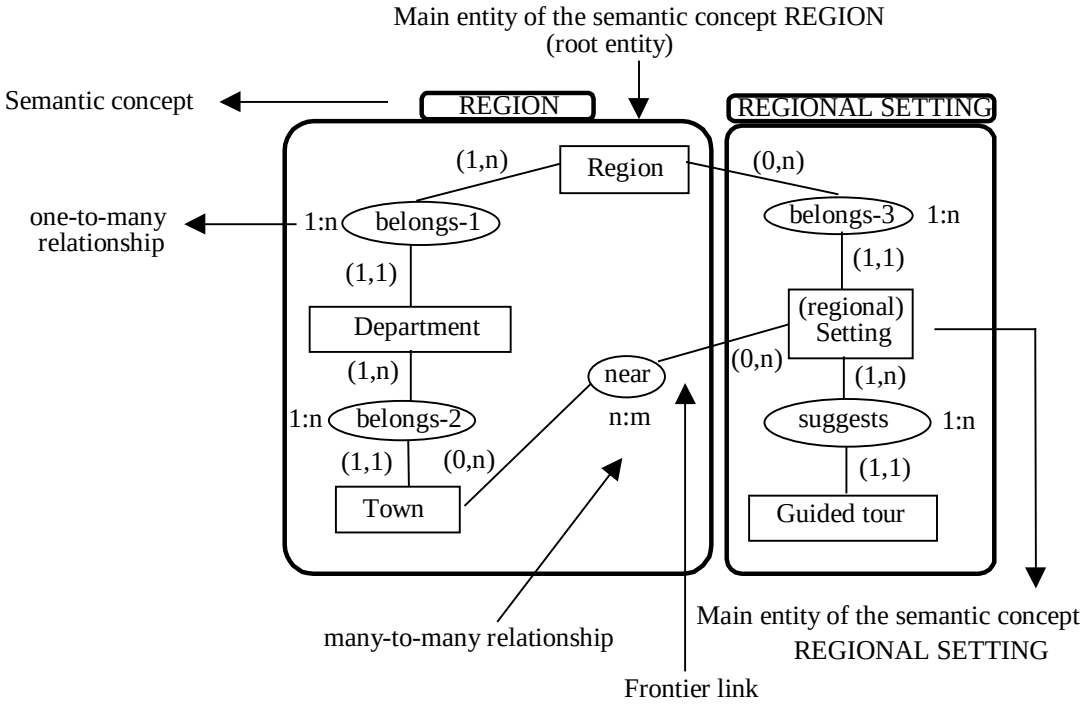

Figure 3: An ER Model split into two semantic concepts

To define a SC, the ER model is seen as a connected graph, where entities are vertices and relationships (characterization links) are edges or directed edges. An edge is directed only if it is a strong or weak characterization link.

Definition of a Semantic Concept: *Given a connected subgraph G of an ER model, G is a Semantic Concept if the graph S obtained further to two operations applied on G (a grouping operation and an elimination operation) is a skeleton.*

Definition of a semantic concept' Skeleton: *Given a semantic concept G, S is its skeleton if:*

- *S is a connected directed graph.*
- *S has a single source vertex V_0 that represents the root entity of the G.*
- *All the vertices of S are reachable from V_0 by following directed edges (characterization links). Reachable vertices represent entities that characterize the root entity of G (V_0).*
- *Any entity of G appears in one of the skeleton's vertices. This implies that all entities of G are either the root entity, either characterizations of the root entity in S.*

For example, in figure 4, S is obtained in two stages. First, vertices of G linked by edges that are pairable characterization links (i.e. linking indissociable entities) are grouped: that's the *grouping operation*. Second, all the edges that are not directed are eliminated: that's the *elimination operation*. The aim is to keep only edges which represent *characterization links between well distinguishable entities*.

In [Goasdoué, 98], we have shown that any ER model can always be split into a partition of semantic concepts (proposition 1). This first proposition led us to find an automated method to construct semantic concepts of an ER model. The method that we propose is based on the

notion of skeleton. In section III , we explain how the notion of skeleton is used and we detail the process of construction.

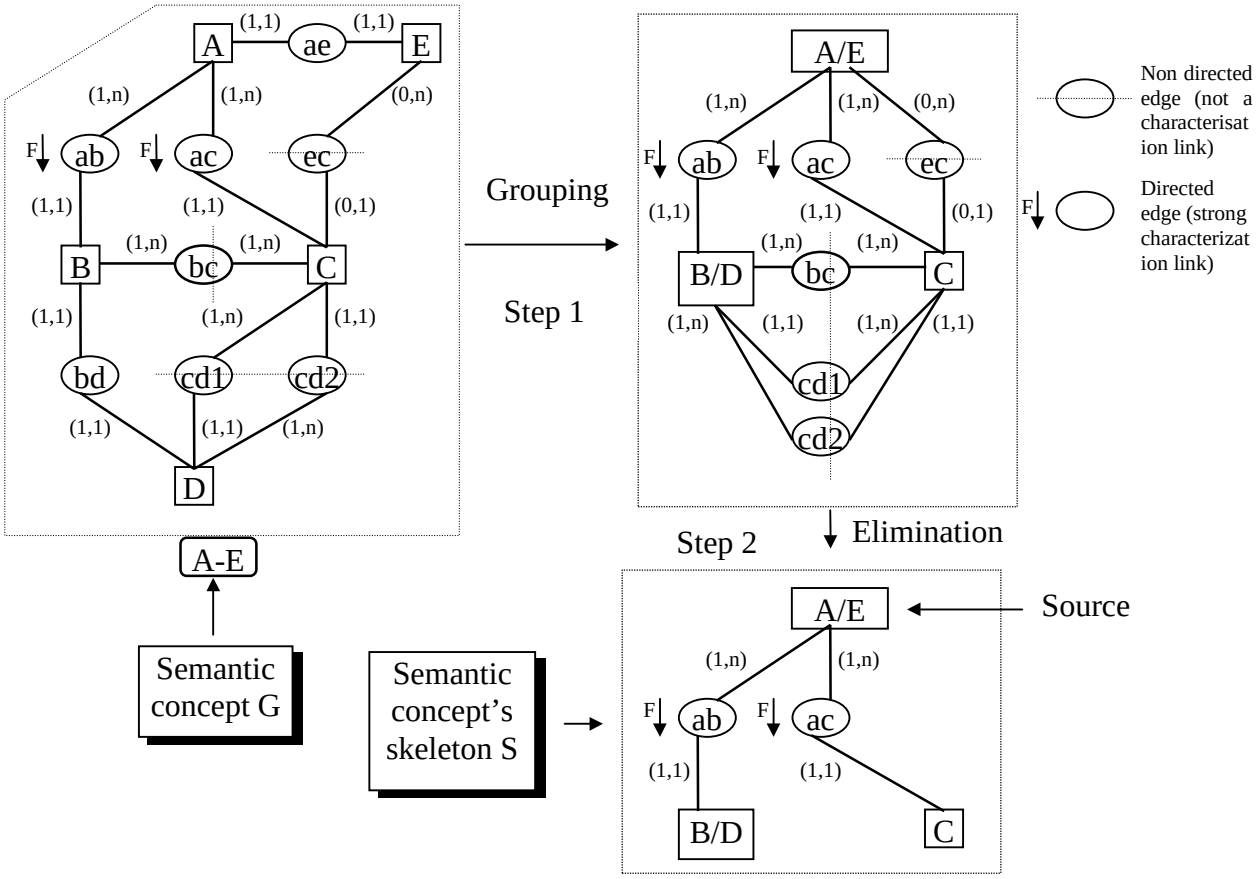

Figure 4: The construction process of the skeleton of the SC "A/E"

III. Identifying semantic concepts

To split an ER model into SCs, we use a method based on the research of SCs' skeletons. We have shown in [Goasdoué, 98] that, given a SC' skeleton, we can find the corresponding SC on the ER model (proposition 2). Furthermore, an ER model may be enormous and complex. In such cases, it might be hard to find relevant groupings of entities and relationships directly on the ER model. It might also be difficult to work directly on the graph representing the whole model. On the opposite, skeletons are simpler graphs than those representing an ER model or even than SCs thanks to the grouping and elimination operations.

So, to identify SCs of an ER model represented by a graph G, we have three stages. In a first preliminary step, we build the skeleton S_G of G. Second, we split S_G into different skeletons in an incremental way. Finally, according to proposition 2, we build the SCs corresponding to each different skeleton of S_G. The first two steps are described in the next sections.

In our approach, we are always interested in discovering the biggest semantic concepts. We would like to describe an ER model by means of a minimum number of concepts.

III.1. The preliminary step: building the skeleton of a graph representing a whole ER model

The aim of this step is to compute the ER model skeleton. To do this, both operations previously introduced (grouping and elimination) are performed on the whole ER model. By analysing the ER model skeleton, we determine which vertices will be sources of skeletons (cf. figure 6). We have shown in [Goasdoué, 98] that given an ER model skeleton's vertex, it can belong to one and only one biggest skeleton (proposition 3). We showed also that, given an ER model skeleton, we can decide for each of its vertices if it will be or not a source of a biggest skeleton (proposition 4). For illustration, figure 6 represents a skeleton corresponding to the ER model in figure 5.

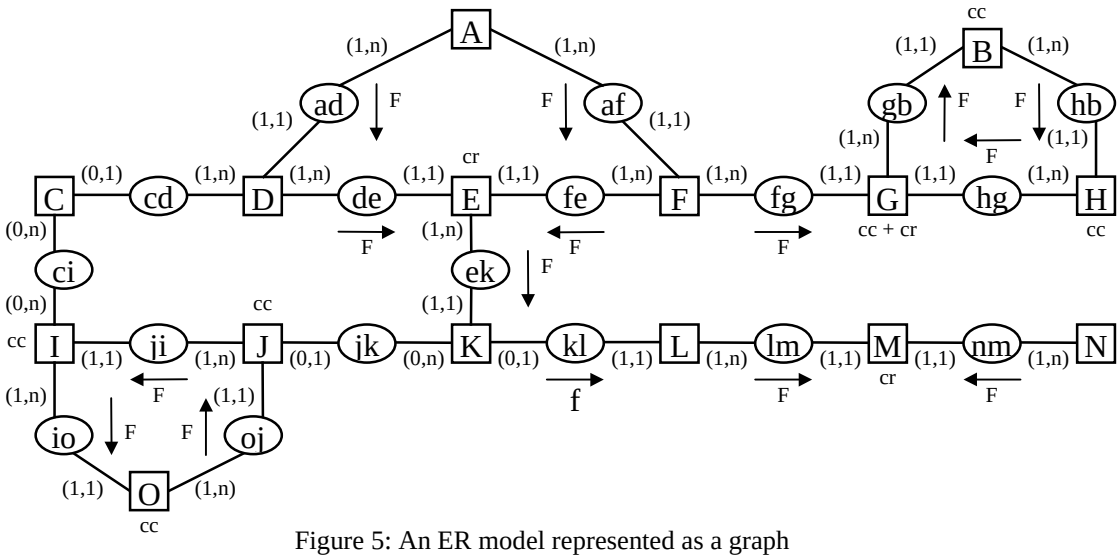

Figure 5: An ER model represented as a graph

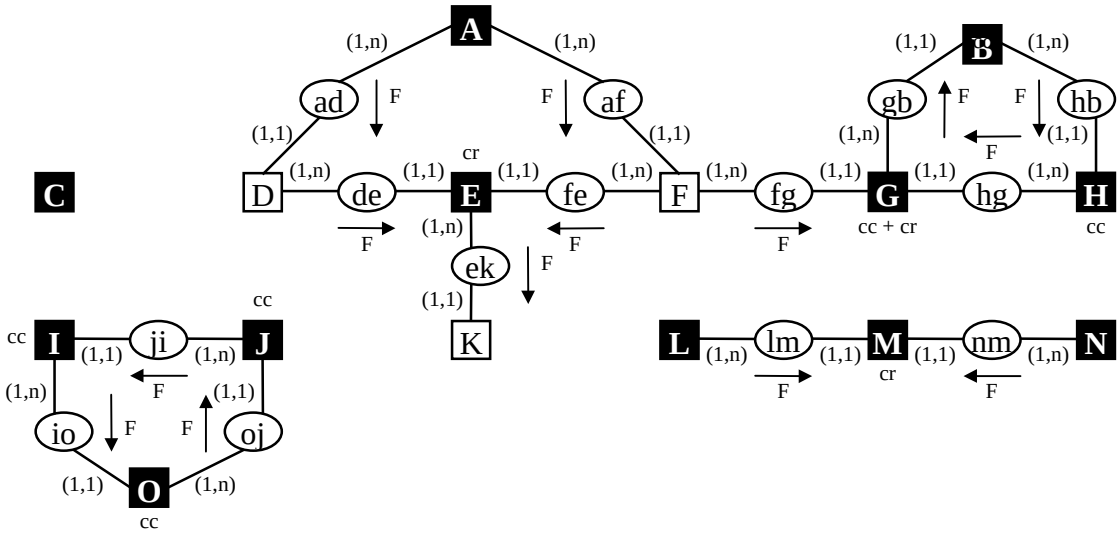

Figure 6: The ER skeleton corresponding to the graph in figure 5 (black vertices are sources of biggest skeletons)

III.2. The identification of skeletons of semantic concepts

Our aim is to automate as much as possible this identification process. Yet, an ER model may often be split into different ways. The administrator of the database (DBA) corresponding to an ER model is the only person who can decide on the best partition. So, we propose to build at the beginning a first one in a fully-automated way. This first partition only proposes groupings which are sure in respect to our construction rules and thus, which don't need the intervention of the DBA. Then, it is shown to the DBA who can decide on further groupings.

The identification of the most relevant concepts of a database is obviously a process which can't be performed without the contribution of a human being, the DBA. The approach that we propose is interesting because it clearly separates the process in two parts, one which can totally be automated and another more little one which needs the DBA to make choices.

To build a partition of SCs in a deterministic way, we showed that, given an ER model skeleton, there is only one partition of it into biggest skeletons (proposition 5). So, we developed algorithms to split an ER model skeleton into *the* partition of its biggest skeletons. The process which is performed is incremental. (Small) skeletons are built and afterwards one can decide to merge several of them.

A variant of the depth-first search algorithm is performed to build the first partition. It allows all vertices that are reachable from a source vertex and that are not sources to be grouped. A source vertex and the vertices reachable from it compose a skeleton (cf. figure 7).

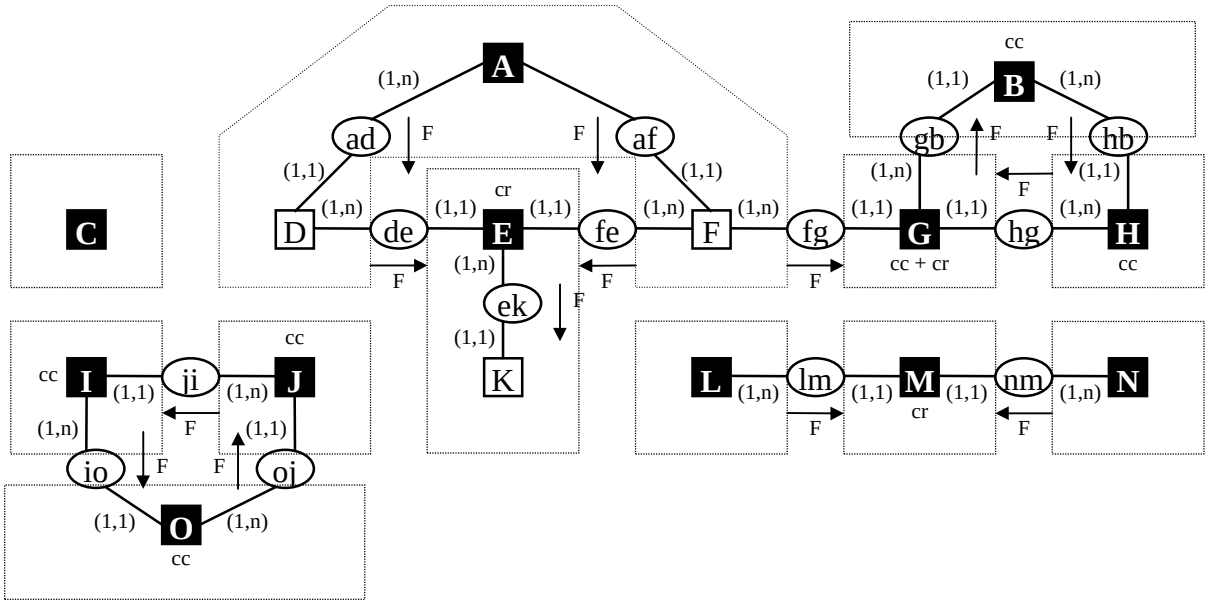

Figure 7: A first partition of SCs obtained in an automated way (black vertices are sources of skeletons)

Then a merging process is performed. We illustrate it on the example below. The final partition which is obtained from figure 7 is represented on figure 8.

- Let S_A and S_E be the two skeletons which source vertex is respectively A and E. We can notice that (1) E characterizes D (because of the strong characterization link "de"), (2) E characterizes F (because of the strong characterization link "fe"), (3) D and F are both characterizations of A (because of the strong characterization links "ad" and "af"). So, E and

its characterizations are also characterizations of A. Since we want to build the biggest skeletons, we merge S_A and S_E.

- Let S_A, S_B, S_G and S_H be the skeletons which source vertex is respectively A, E, G and H. We can notice that (1) B, G and H characterize themselves, (2) G is a characterization of F, (3) F is a characterization of A. So, we can deduce that B, G and H characterize A. Since we want to build the biggest skeletons, we merge S_A, S_B, S_G and S_H.

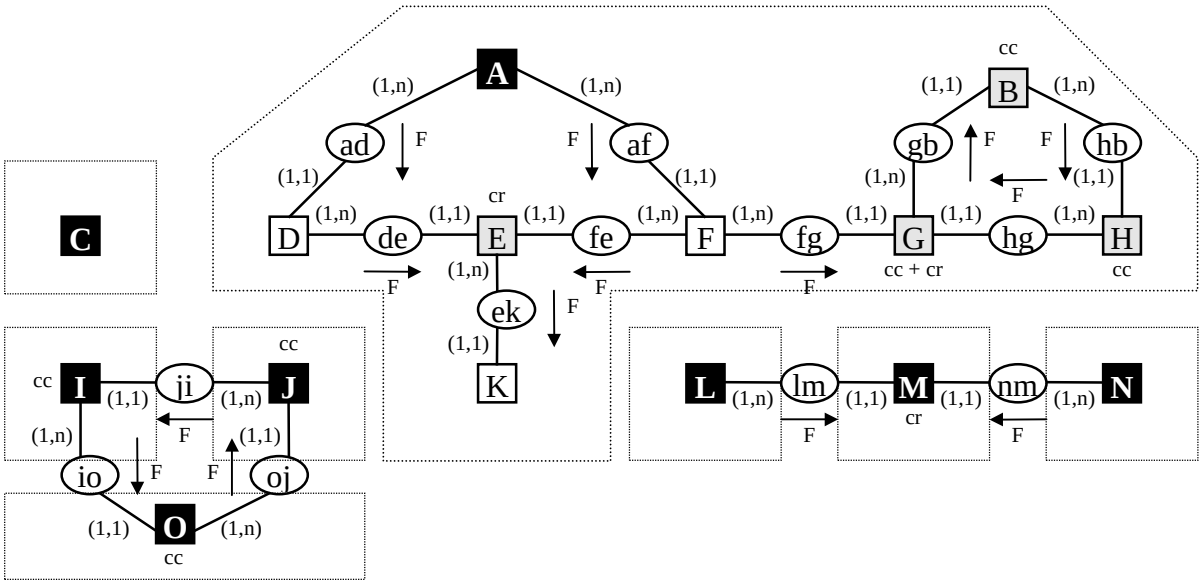

Figure 8: The final partition of SCs obtained in an automated way (black vertices are sources of skeletons)

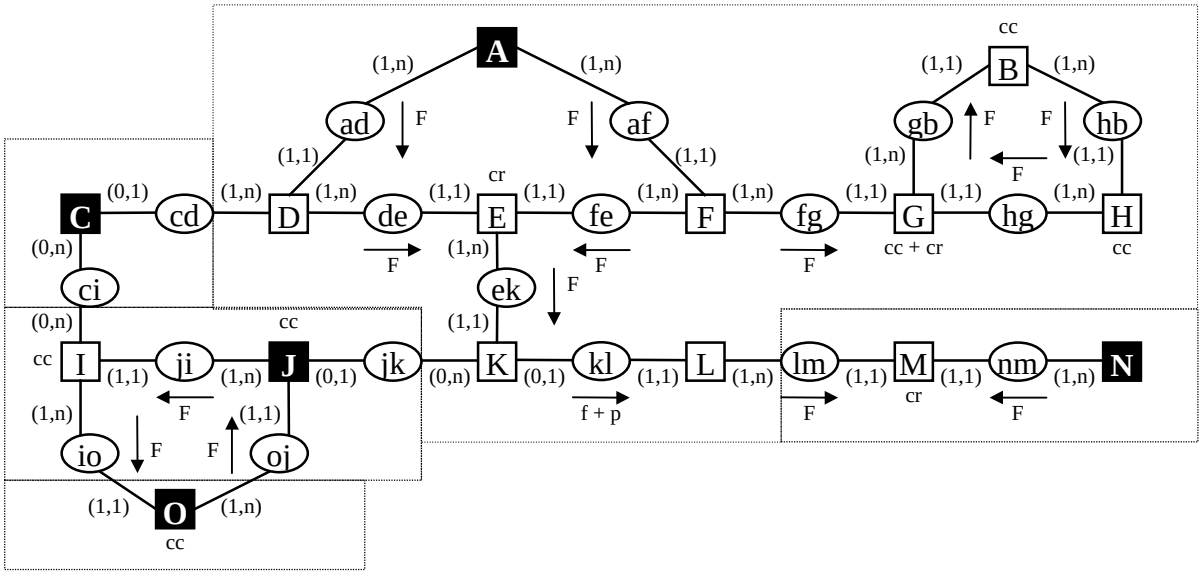

Figure 9: A SCs partition of the ER model of figure 5 (black vertices are roots of produced SCs)

At the end of the identification step, we have a partition of an ER model into SCs (cf. figure 9 for example). Each one represents a semantic concept which significant entity is the root entity of the SC. Now, to build the knowledge base that will reflect the semantic contents of an ER model, expressed in CARIN, we have to describe each of the identified SCs, in the terms of the domain model.

IV. Describing semantic concepts in CARIN

To build the knowledge base that describes the semantic contents of a database, we have developped three tools: a semantic concepts explorer, a domain model explorer and a CARIN sentences composer. All of them are automated supports in the description of a whole relational DB. The idea behind the approach is that the space of choices of concepts to describe can, to some extend, be controlled by the introduction of the notion of semantic concepts. That way, a DBA will have to describe its DB only part by part, each part corresponding to a SC, an abstraction representation of semantically related and indissociable data.

IV.1. The semantic concepts explorer

This tool allows the administrator of a given database to browse the previously identified semantic concepts of his ER model. It is a way to recall to him the significant notions to describe.

IV.2. The domain model explorer

This tool displays all the hierarchies that can be computed from concept inclusions and concept declaration statements in the domain model (cf. the hierarchy presented in *I.2.1*)

For each node of a hierarchy that is a base concept, a description in natural language is available. Moreover, for each node, we can retrieve all the roles that have the node type as type of one of their arguments. So, the DBA can browse the different hirarchies to learn the vocabulary defined by the domain model, or find the concept that represents the best a notion he wants to put in the knowledge base.

We can also list all the roles of the terminology, with, for each of them, their meaning in natural language. Moreover, if information is also available from domain model, we can display for each role the type of concepts that it links.
For example, we can deduce that the role *DepartureDate* needs a concept of type *Date* as its second argument from the following expression: $DepartureDate(X,Y) \wedge \neg Date(Y) \Rightarrow \bot$.

IV.3. A CARIN sentences composer

When a DBA decides to describe a significant notion encountered in a semantic concept of his ER model (thanks to the semantic concepts explorer), he can choose the concept of a hierarchy that represents the best that notion (thanks to the domain model explorer). The result of such an action is to produce automatically a new source relation declaration: $v_i(x) \Rightarrow Cj(x)$, were C_j is the concept that has just been selected in the domain model. The purpose of our tool is to help the DBA to characterize that source relation.

First, we try to characterize v_i using roles $R_{k\ (1\le k\le n)}$ that are associated with objects of C_j. For example, the roles *DepartureDate, ArrivalDate, DeparturePlace, ArrivalPlace* and *MeansTransport* can be used to characterize the concept *Flight*, according to the piece of domain model presented in I.2.1.
Possible characterizations of v_i are expressed thanks to source relation inclusions like: $v_i \subseteq C_1$ *and* C_2 *and …* *and* C_m, where each $C_{l\ (1\le l\le m)}$ is of the form $(\le n\ R_k)$, $(\ge n\ R_k)$ or $(\forall R_k.C_{accepted})$.

On the one hand, for a $C_{l\ (1\le l\le m)}$ like $(\le n\ R_k)$ or $(\ge n\ R_k)$, there is no particular problem. The DBA has only to select a role $R_{k\ (1\le k\le n)}$ and to give the cardinality *n*. On the other hand, a $C_{l\ (1\le l\le m)}$ of the form $(\forall R_k.C_{accepted})$ implies that the concept $C_{accepted}$ is compatible with R_k. To be sure that $C_{accepted}$ is compatible with a R_k selected by the DBA, let's consider the following process:

Case 1: if $C_{accepted}$ is a concept name *CN*, it must appear in the domain model and *CN* must have the same type as the one of R_k' second argument. To do this, our tool uses the domain model explorer, pointing at the node of the hierarchy in which *CN* appears. Then, the DBA has to choose *CN* or one of its specializations for $C_{accepted}$.

Case 2: if $C_{accepted}$ is a concept like $(\le n\ R_l)$ or $(\ge n\ R_l)$, R_l must have as first argument's type, the same type as the R_k' second argument. To do this, our tool retrieves from the domain model all the roles R_l which satisfy this property. Then, the DBA will have to choose one of them (R_l), and to give the cardinality *n*.

Case 3: if $C_{accepted}$ is a concept like $(\forall R_l.C_{accepted'})$, the choice of R_l is done as R_k's choice is. The concept $C_{accepted'}$ must be of the same type as the type of R_l' second argument. Thus, $C_{accepted'}$ is chosen like $C_{accepted}$ was (i.e. case 1,2,3 or 4).

Case 4: if $C_{accepted}$ is a concept like C_1 *and* C_2 *and …* *and* C_p, each of the $C_1 \ldots C_p$ is defined like $C_{accepted}$ is (i.e. case 1,2,3 or 4).

This way, if we want to express that we have a source relation v_i over flights which arrival places are located in Europe and which type is Tourism, we can generate the following expressions, according to the piece of domain model presented in I.2.1.: $v_i(x) \Rightarrow Flight(x)$, $v_i \subseteq (\forall ArrivalPlace.Europe)^5 \wedge (\forall MeansTransport.TourismFlight)$.

Second, we try to characterize v_i using roles $R_{k'\ (1\le k'\le n)}$ that are known not to associate objects of C_j but objects of classes subsumed by C_j. These roles are those of the domain model that are different of the R_k above, but that accept the same concept type as the one of C_j as first argument.
Again, possible characterizations of v_i are expressed thanks to source relation inclusions like: $v_i \subseteq C_1$ *and* C_2 *and …* *and* C_m, where each $C_{l\ (1\le l\le m)}$ is a $(\le n\ R_{k'})$, $(\ge n\ R_{k'})$ or $(\forall R_{k'}.C_{accepted})$. The same process of characterization as the one describes above is used.

For example, if we want to express that we have the source relation v_i over flights that propose to have diner on board, we can generate the following expression, according to the piece of domain model presented in I.2.1.: $v_i \subseteq (\ge 1\ AssociatedMeal) \wedge (\forall AssociatedMeal.Diner)^6$.

Related work and conclusion

Our aim was to obtain descriptions using terms in the domain model from representations using terms particular to a database schema. The fundamental problem which arises is

[5] Here, we consider that the concept *Europe* appears in the domain model.
[6] Here, we consider that the concept *Diner* appears in the domain model.

semantic heterogeneity – the fact that the same concepts are represented differently in a database schema and in the domain model.
Some issues raised by semantic heterogeneity have been studied in the database community. When two or more databases need to work together, in many cases the same data is replicated. Different database schemas and different conceptualizations are typically used to represent the replicated data.
So, in the database schema integration field which aim is to construct a global, unified schema from existing or proposed databases, the semantic heterogeneity problems to be dealt with are structural and naming conflicts [Batini & al., 96]. Structural conflicts arise as a result of a different choice of modeling constructs or integrity constraints. In such cases, the same concepts are represented differently although the same modeling language is generally used. Some constructs used in the different schemata may be equivalent. Concepts may also be related among themselves with different dependencies because the contexts of the DB applications are not similar. Whatever the reason is, the concepts which modelization is problematic always *appear* in all schemata which are compared, but not in the same fashion. Naming conflicts arise because people from different application areas refer to the same data using their own terminology and names. Thus, to make schemas compatible, one must replace terms by other ones, the new terms belonging to the same level of discourse. In both cases, conflict discovery and restructuring are in general aided by a strong interaction between the different DB designers.
Other works have focused on heterogeneity including different data models (ER, object, functional, and so on). [Spaccapietra & al., 92] proposes a language to define the correspondence between constructs of heterogeneous schemata. They allow some kinds of structural conflicts to be solved automatically.
To summarize, in most research works on the integration DB schema field, techniques to solve the semantic heterogeneity problem are used prior to the integration step. It is very often a manual process, except for some kinds of structural conflicts. In any cases, the preintegration step is considered the responsability of the DB designers. Furthermore, in much work, mappings between database schemata are assumed to be provided. A solution to the semantic heterogeneity problem would be to enhance the semantic description of each schema. For that, Bonjour in [Bonjour & al., 1994] proposes to introduce concept bases on top of a set of schemes to integrate. These bases could help to compare concepts represented in different systems.
More recently, other database works have focused on importing and integrating selected portions of DB schemata as in federated [Sheth & al., 90] or knowledge-based mediator or data warehouses architectures [Widom, 95]. A lot of problems of semantic heterogeneity which arise are the same nature as in data integration field. But they have been addressed a little. Nevertheless, in the knowledge-based mediator approaches, we notice two trends. Mediator approaches have in common the use of knowledge bases which describe both the domain model and the contents of information sources. So, they don't need correspondences directly between the information sources but, instead, they need correspondences between the domain model and the descriptions of the contents of each information source. A way to make such correspondences easier is to capture the intended meaning of DB schemata using ontologies [Gruber, 93]. It relies on manipulation techniques coming from the fields of artificial intelligence. In Observer [Mena & al., 96] for example, the objects in the sources are represented as intensional descriptions by pre-existing ontologies. The query engine rewrites user queries by using interontology relationships to obtain semantics-preserving translations across the ontologies. The approach is interesting but new problems arise : how to build the ontologies ? how to acquire the terminological relationships represented between terms across the ontologies ? An other approach is to use the same vocabulary to describe both the

domain model and the contents of the different sources. The problem of different vocabularies disappears but we must be able (1) to describe each source with the vocabulary of the domain model, and (2) to link the descriptions of the contents of a DB to the DB itself. Our paper is relative to the first point of this second approach. As far as we know, no other research works have proposed solutions in this direction. We have faced to a semantic heterogeneity problem because concepts in the domain model and in the DB schemata were not represented at the same level of abstraction and not because of structural or naming conflicts. Our aim was not to find the good structure or the good name corresponding to each concept in a DB schema. It was to identify parts of DB schemata which represent concepts relevant for a user and which will be able to be made in correspondence with concepts in the domain model, corresponding to a user view point too.

On another hand, our problem is relative to knowledge engineering. We want to build a knowledge-based mediator. Thus, we need techniques to construct all the knowledge bases useful to the mediator. Current knowledge engineering works describe the structure of a knowledge-based system (KBS) through highly structured models [Schreiber & al., 1994]. The domain models which describe the specific knowledge of a domain are one of the components of such models. Recently, in the knowledge engineering community, research works have been conducted to characterize domain knowledge and to help building domain models by means of ontologies [Gruber, 1993]. Building ontologies become then a key issue. An ontology is based on the definition of a structured and formalized set of concepts. A great part of it comes from text analysis. So, one trend is to benefit from both knowledge engineering and linguistics approaches. Researchers have studied mutual contributions and this led them to elaborate the concept of Terminological Knowledge Base (TKB), first defined by Ingrid Meyer [Skuce & al., 91]. In France, the works of the TIA research group is centered on this notion too [Bourigault & al., 95]. A TKB is an intermediate model which helps toward the construction of a formal ontology ; it contains conceptual data, represented in a network of domain concepts, but also linguistic data on the terms used to name the concepts. A TKB can enhance communication and be a great help to choose the names of concepts. Such research works address the problem of the identification of concepts. We deal with the same problem but we have to identify concepts from database schemata, not from text analysis. So, the techniques that we propose are specific ones, based on the notion of semantic concepts.

In conclusion, this paper deals with identifying and modeling relevant concepts. First, we have presented a way to identify relevant concepts. Our aim was to automate this process as much as possible although it can't be entirely performed without the contribution of DB designers. We have identified two different parts in the process performed sequentially: one which can be totally automated and another one performed in cooperation with the DB designer. Second, we have presented techniques usable by DB designers to be guided in the description of relevant concepts in CARIN and using terms in the domain model. Much of the techniques described in the paper have been implemented in Java.

References

[Batini & al., 96] Batini, C., Lenzerini M., Navathe S.B., "A comparative Analysis of Methodologies for Database Schema Integration", ACM Computing Surveys, Vol. 18, n°4, pp. 323-364, Dec. 1986.

[Bonjour & al., 94] Bonjour M., Falquet G., "Concept bases: A support to Information Systems Integration", CAISE 94.

[Bourigault & al., 95] Bourigault D., Condamines A., "Réflexions autour du concept de base de connaissances Terminologiques", Dans les actes des journées nationalies du PRC-IA, Nancy, 1995.

[Brès, 93] Brès P.-A., "L'apport de l'approche objet dans la conception de systèmes d'information", AGL'93, Pact-Group, 6-8 Avril 1993.

[Chen, 76] Chen P.S., "The entity-relationship model", ACM Transactions on Database Systems, 1, 166-192, 1976.

[Goasdoué, 98] Goasdoué F., "Assistance à la conception de bases de connaissances dédiées au médiateur PICSEL", Mémoire de D.E.A. d'informatique, Université Paris 11, Sept. 1998.

[Gruber, 93] Gruber T.R., "A translation Approach to Portable Ontology Specifications", Knowledge Acquisition, 5, pp. 199-220, 1993.

[Levy & al., 98] Levy A., Rousset M.-C., "Combining Horn Rules and Description Logics in CARIN", Artificial Intelligence Journal, vol. 14, September 1998.

[Mena & al., 96] Mena E., Kashyap V., Seth A., Illaramendi A., "OBSERVER: An approach for Interoperation accross pre-existing ontologies", Proceedings of the first IFCIS International Conference on Cooperative Information Systems (CoopIS'96), June 96.

[Rousset & al., 98] Rousset M.-C., Lattes V., "The use of CARIN language and algorithms for information Integration: the PICSEL project", Intelligent Information Integration Workshop associated with ECAI'98 Conference, Brighton, August 1998.

[Schreiber & al., 94] Schreiber A.T., Wielinga B.J., De Hoog R., Akkermans J.M., Van de Velde W., "CommonKads: a comprehensive methodology for KBS development", IEEE Expert, 9(6): 28-37, 1994.

[Sheth & al., 90] Sheth A. P., Larson A., "Federated Database Systems for managing Distributed, Heterogeneous and Autonomous Databases", ACM Computing Surveys, Vol. 22, n°3, pp. 183-236, Sept. 90.

[Skuce & al., 91] Skuce D., Meyer I., "Terminology and Knowledge Acquisition: exploring a symbiotic relationship", In Proc. 6th Knowledge Acquisition for Knowledge-based System Workshop, Banff, pp. 29/1-29/21, 1991.

[Spaccapietra & al., 92] Spaccapietra S., Parent C., Dupont Y., "Model independant Assertions for Integration of Heterogeneous Schemas", VLDB Journal, 1, pp. 81-126, Joachim Schidt editor, 1992.

[Widom, 95] Widom J., "Research Problems in Data Warehousing", Proceedings of Fourth International Conference on Information and Knowledge Management (CIKM'95), pp. 25-30, Baltimore, Maryland, Nov. 1995.