The MIEL System: Uniform Interrogation of Structured and Weakly-Structured Imprecise Data

Ollivier Haemmerlé, Patrice Buche, Rallou Thomopoulos

▶ **To cite this version:**

Ollivier Haemmerlé, Patrice Buche, Rallou Thomopoulos. The MIEL System: Uniform Interrogation of Structured and Weakly-Structured Imprecise Data. Journal of Intelligent Information Systems, 2007, 29 (3), pp.279-304. 10.1007/s10844-006-0014-z . lirmm-00195268

HAL Id: lirmm-00195268

https://hal-lirmm.ccsd.cnrs.fr/lirmm-00195268

Submitted on 31 May 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

J Intell Inf Syst (2007) 29:279–304
DOI 10.1007/s10844-006-0014-z

The MIEL system: Uniform interrogation of structured and weakly-structured imprecise data

Ollivier Haemmerlé · Patrice Buche · Rallou Thomopoulos

Received: 16 July 2003 / Revised: 4 October 2005 /
Accepted: 30 January 2006 / Published online: 24 January 2007
© Springer Science + Business Media, LLC 2007

Abstract We present an information system developed to help assessing the microbiological risk in food. That information system contains experimental results in microbiology, mainly extracted from scientific publications. The increasing amount of the experimental results available and the difficulty to integrate them into a classic relational database schema led us to design a system composed of two distinct subsystems queried through a common interface. The first subsystem is a classic relational database. The second subsystem is a database containing weakly-structured pieces of information expressed in terms of conceptual graphs. The data stored in both bases can be fuzzy ones in order to take into account the specificities of the biological information. The uniform query language used on both relational database and conceptual graph database allows the users to express preferences by using fuzzy sets in their queries. The MIEL system is now operational and used by the microbiologists involved in the Sym'Previus French project.

Keywords Information integration · Fuzzy sets · Conceptual graphs

O. Haemmerlé (✉)
GRIMM-ISYCOM, Université de Toulouse le Mirail, Département de Mathématiques-Informatique, 5 allées Antonio Machado, F-31058 Toulouse Cedex, France
e-mail: ollivier.haemmerle@univ-tlse2.fr

P. Buche
INRA, Département Mathématiques et Informatique Appliquées, Unité Mét@risk, 16 rue Claude Bernard, F-75231 Paris Cedex 5, France e-mail: patrice.buche@inapg.inra.fr

R. Thomopoulos
INRA, UMR IATE (bâtiment 31), 2 place Viala, 34060 Montpellier Cedex 1, France
e-mail: rallou.thomopoulos@ensam.inra.fr

R. Thomopoulos
LIRMM, UMR 5506, 161 rue Ada, F-34392 Montpellier Cedex 5, France

1 Introduction

As a result of several food safety problems, the Marrakech Agreement was signed in 1994 during the creation of the World Trade Organization. Included in this agreement, the SPS Agreement (Sanitary and Phytosanitary measures) concerns the international trade of food, and targets the safety and protection of human health. An important principle of the SPS Agreement is the study of risk analysis.

Our research project is part of the Sym'Previus project which is a French programme involving research and technical centers, food companies, professional and governmental organizations. The Sym'Previus project aims at building a tool for microbiological risk analysis in food products. This tool is based on a database containing data extracted from scientific publications in microbiology.

Our project has to deal with several important specificities which impact on the one hand the kind of data we have to store, and on the other hand the kind of processings we want to run on these data.

As for the kind of data we want to store, their first specificity is that they concern a scientific field of intense activity. It is really difficult to propose the microbiologists a classic database schema which remains up to date for a long time, since their needs are constantly evolving. Their second characteristic is that they can be numerical (values of the different experimental parameters such as the temperature, the pH, ...) or symbolic. The symbolic information is often hierarchized (taxonomy of bacteria (Ballows, Truper, Dworkin, Harder & Schleifer, 1992), of food products (Ireland & Moller, 2000), ...). Finally, the data can be imprecise since they concern complex biological processes; moreover the measurement tools are limited by their internal imprecision.

As for the processing on these data, it is important to note that the end-users are non-specialists in computer science. Our tool is dedicated to help microbiologists in their search for information in order to prevent contaminations in food products. Another specific point is that a database containing biological experiments is incomplete as it will never cover all the possible experimental conditions. The risk of empty answers to a query is significant.

All these specificities led us to design a data model which presents several original aspects.

The first choice we made is to dispatch the data into two distinct bases. The first one is a classic relational database which contains the stable part of our information. The choice of a relational database was made for efficiency and standardization reasons and the schema of that relational database has been designed in close collaboration with microbiologists. But, as modifying the schema of such a database is quite an expensive operation, we decided to use an additional base in order to store information that was not expected when the schema of the database was designed, but appears to be useful nevertheless. We chose to represent this part of the data—which we call "weakly-structured"—in the conceptual graph model (Sowa, 1984) for many reasons: (i) its graph structure appeared as a flexible way of representing complementary information; (ii) its readability seemed to us an asset since we had to work with non-specialists; (iii) its interpretation in first order logic provided a robust theoretical framework; (iv) a development platform providing efficient algorithms was available; (v) the distinction between the terminological part and the assertional

part of the knowledge (as in Description Logics, for example) was valuable to implement query relaxation mechanisms.

The necessity of representing imprecise values in our database concerns the relational database as well as the conceptual graph database. Previous works studied the use of the fuzzy set framework to represent imprecise values by means of possibility distributions (Umano, 1982; Prade & Testemale, 1984). We propose a representation of imprecise values in the relational database based on the fuzzy set theory, close to the representation used in FSQL (Galindo, Cubero, Pons & Medina, 1998). Unfortunately, the conceptual graph model was not well-designed for the representation of numerical and imprecise values; that is why we proposed an extension of that model, in order to allow the storage of data with the same expressivity as the data stored in the relational database. The combination of a knowledge representation model with a way of introducing imprecision has been proposed in other previous works. In particular, we can cite formalisms that describe ontologies like the object model (Dubois, Prade & Rossazza, 1991), or information retrieval using description logics (Sebastiani, 1994). The latter are part of the semantic networks, just as the conceptual graph model. The introduction of the fuzzy set theory into the conceptual graph model has been studied in Morton (1987) and extended by several works such as Cao (1999). Compared to the previous approaches, we propose a more homogeneous and integrated way of combining conceptual graphs and fuzzy sets in Thomopoulos, Buche and Haemmerlé (2003b). Moreover, since we use taxonomies, we propose a way of representing fuzzy sets defined on such taxonomies.

In addition to the originalities of our data model, we propose some specificities concerning the query mechanism.

The first one is that the end-users express their queries in a uniform query language which allows one to query the relational database as well as the conceptual graph database. The answers are returned to the users by means of the graphical user interface in a uniform way.

The users express their queries through a set of pre-written queries which we call *views*. These views can be complemented by the users through the simple graphical user interface of the MIEL system. That interface allows the users to specify their projection attributes and their selection criteria. The ontology of the MIEL system can also be browsed by the end-users in order to express their selection criteria.

Finally we propose the user the possibility of expressing large selection criteria which consist of disjunctions of values. The goal is to return a maximal amount of answers in order to avoid empty answers. The user can express different levels of preferences on these values so as to allow flexible querying. Such an expression of preferences is represented in terms of fuzzy sets. The use of fuzzy sets allows us to propose a homogeneous way of representing the values of the attributes in the data as well as the values of the selection criteria in the queries. The expression of queries using fuzzy values has already been studied in the framework of the relational database model. Theoretical studies have been proposed to extend the SQL language by introducing fuzzy predicates processed on crisp information (Bosc & Pivert, 1995) and implementations such as the FQUERY97 system (Zadrozny & Kacprzyk, 1998) have been proposed under the QBE-like Microsoft Access graphical environment. Our work proposes a fuzzy query mechanism in the relational model and an analogous mechanism in the conceptual graph model.

This article presents the solutions we propose to the specificities listed above. Our work has been implemented in the MIEL[1] system. Section 2 presents the data model of the MIEL system: Section 2.1 presents the ontology used, Sections 2.2 and 2.3 present respectively the relational database schema and the conceptual graph database schema. Section 3 presents the queries in the MIEL system: Section 3.1 defines the query language, Sections 3.2 and 3.3 present the definition and the processing of the queries, respectively in the relational and in the conceptual graph subsystem.

2 The MIEL data model

The following sections present the choices we made in the design of our data model in order to take the specificities of the data into account. Section 2.1 presents the ontology of the MIEL system. The attributes are introduced as well as their associated variation domains, which can be hierarchized in order to allow the representation of taxonomies. We show that in the actual data, the attribute values are "fuzzy": numbers are fuzzy numbers and every symbol is augmented with a grade. These fuzzy values express possibility distributions. They are stored into two distinct databases: (i) a relational database with a stable schema; (ii) an additional database expressed in terms of conceptual graphs, containing the data which do not fit the relational database schema. These two distinct bases constitute the actual schema of the MIEL system. The relational database schema is presented in Section 2.2, the conceptual graph database schema is presented in Section 2.3.

2.1 The ontology of the MIEL data model

In this section we introduce the *ontology* of the MIEL data model, which contains all the knowledge of the domain used in the MIEL system. That ontology is duplicated in the relational and in the conceptual graph databases, since it is necessary for both subsystems to work.

The basic notion of the data model of the MIEL system is the concept of *attribute* which must be understood in its classic database meaning. We propose to use, instead of crisp values, fuzzy values used in two different ways: (i) in the data, in order to represent imprecise values expressed in terms of possibility distributions; (ii) in the queries, in order to express preferences on the domain of a selection criterion.

In this paper, we use the representation of fuzzy sets proposed in Zadeh (1965, 1978). We remind that a *fuzzy set* f on a domain X (which can be continuous or discrete) is defined by a membership function μ_f from X to $[0, 1]$ that associates the degree to which x belongs to f with each element x of X. We call respectively *support* and *kernel* of f the following subsets of X: $support(f) = \{a \in X \mid \mu_f(a) > 0\}$; $kernel(f) = \{a \in X \mid \mu_f(a) = 1\}$.

The adequation of an imprecise data B to a fuzzy criterion A is classically measured in the fuzzy set theory by means of two degrees: (i) the *possibility degree of matching* between A and B (Zadeh, 1978), denoted $\Pi(A\ ;\ B)$, which is an overlap

[1] Moteur d'Interrogation ELargie in French, for Extended Query Processor.

measure; (ii) the *necessity degree of matching* between B and A (Dubois & Prade, 1988), denoted $N(A\ ;\ B)$, which is an inclusion measure.

A variation domain and a definition domain are associated with each attribute. The variation domain corresponds to the universe of discourse; the definition domain is the set of fuzzy sets which can be defined on the variation domain: it corresponds to the actual domain in the classical database meaning.

Definition 1 $\mathcal{A}$ is the finite *set of attributes* of the MIEL data model. Each attribute $a \in \mathcal{A}$ is characterized by its type $Type(a)$, its variation domain $Dom_v(a)$ and its definition domain $Dom(a)$. The type $Type(a)$ of an attribute a can be *numerical*, *symbolic* or *hierarchized*. Depending on its type, the variation domain $Dom_v(a)$ of an attribute a is:

- If $Type(a)$ is numerical, $Dom_v(a)$ is defined as a subset of $\mathbb{R}$;
- If $Type(a)$ is symbolic, $Dom_v(a)$ is defined as a set of symbolic constants;
- If $Type(a)$ is hierarchized, $Dom_v(a)$ is defined as a set of symbolic constants and a partial order defined on it.

In all the cases, $Dom(a)$ is defined as the set of all the possible fuzzy sets on $Dom_v(a)$.

A hierarchized variation domain is close to the notion of general partial order on an attribute domain used by Ginsburg and Hull (1983) in the study of order dependencies.

Remark 1 A *value* of an attribute a belongs to $Dom(a)$. In other words, it is a map π of $Dom_v(a)$ to [0,1]. We denote by $\pi(x)$ the degree of possibility that the effective value of a is x.

For example, the variation domain of the numerical attribute pH is the interval [0,14] on $\mathbb{R}$. The variation domain of the symbolic attribute *Author* could be the

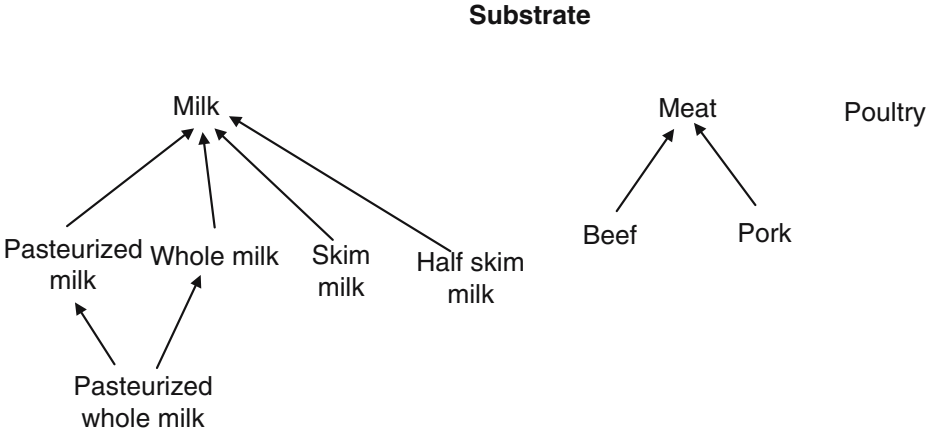

Fig. 1 A part of the variation domain of the attribute "Substrate"

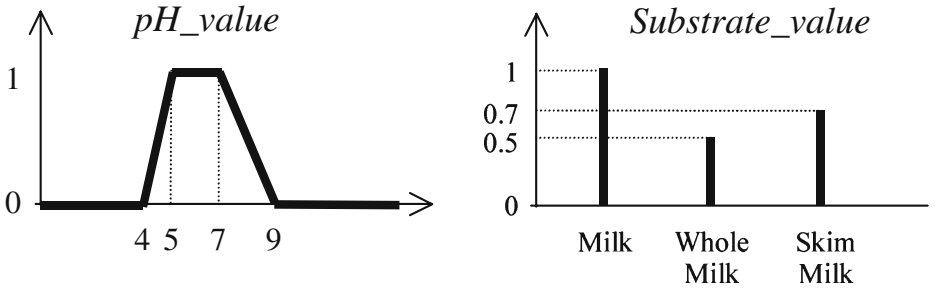

Fig. 2 Two examples of imprecise values

set {*S.Ajjarapu*, *C.P.Rivituso*, *M.Zwietering*}. A part of the variation domain of the hierarchized attribute *Substrate* is represented in Fig. 1.

The value *pH_value* of Fig. 2 schematizes an example of value for the attribute *pH* (that value belongs to $Dom(pH)$: it is a map of $Dom_v(pH)$ into [0,1]). The value *Substrate_value* of Fig. 2 schematizes an example of value for the attribute *Substrate* (that value belongs to $Dom(Substrate)$: it is a map of $Dom_v(Substrate)$ into [0,1]; the elements of $Dom_v(Substrate)$ having a degree equal to 0 are not represented).

Remark 2 For simplicity, we consider that if a same element e appears in two distinct hierarchized symbolic variation domains $Dom_v(a)$ and $Dom_v(b)$, then all the specializations of that element must appear in both $Dom_v(a)$ and $Dom_v(b)$ with the same partial order.

Note that in our data model, we consider that all the values are imprecise values. The case of a crisp value for an attribute a is a particular case of the case of an imprecise value, such that $\exists x \in Dom_v(a)[\pi(x) = 1$, and $\forall y \neq x, \pi(y) = 0]$.

For simplicity, and since it corresponds to the application needs, we chose to limit the representation of numerical values to trapezoidal functions in the actual database. These trapezoidal functions are stored by means of 4 characteristic points defining the limits of the support and the kernel of the fuzzy set. In the example of Fig. 2, these four characteristic points are {4, 5, 7, 9}.

2.2 The schema of the relational database

In this section, we focus on the choices we have made in order to map the ontology of the MIEL data model presented previously onto the RDB schema. We present how the attributes and their variation domains are represented in the RDB. We successively consider the way of representing an attribute belonging to the ontology of the MIEL data model, when that attribute is respectively of type *numerical*, *symbolic* or *hierarchized*.

2.2.1 Representation of a numerical attribute

The representation of a value of a numerical attribute of $\mathcal{A}$ in the relational schema is done by means of a row of an additional table which contains the unique identifier

ExpeId	Substrate	FuzzyPHId	FuzzySetId	MinSupp	MinKer	MaxKer	MaxSupp
27	Pork	221	221	4	5	6	7
39	SkimMilk	223	223	9	10	12	12.5

Fig. 3 The *left* table presents an example of relation referencing numerical fuzzy values. The *right* table contains a part of the relation *FuzzyPH* which contains the actual numerical fuzzy sets (the second row corresponds to a crisp value)

of the fuzzy set and four attributes which correspond to the four characteristic points of the trapezoidal function. The tables presented in Fig. 3 present an example of numerical attribute represented in the relational database.

2.2.2 Representation of a symbolic attribute

The representation of a value of a symbolic attribute a of $\mathcal{A}$ in the relational schema is done by means of one or several rows of an additional table which contains three columns: the unique identifier of the fuzzy set, an element of $Dom_v(a)$ and its associated membership degree in that fuzzy set. We remind that a fuzzy set on a symbolic variation domain is defined as a set of pairs $(element, degree)$. The tables presented in Fig. 4 present an example of the value of a symbolic attribute represented in the relational database.

In addition, the variation domain of each attribute a of $\mathcal{A}$ of type *symbolic* used in the relational schema is stored in a reference table which contains all the possible values for the attribute a.

2.2.3 Representation of a hierarchized attribute

The representation of a value of a hierarchized attribute of $\mathcal{A}$ in the relational schema is done in exactly the same way as the representation of a symbolic attribute (see Section 2.2.2).

The variation domain of each attribute a of $\mathcal{A}$ of type *hierarchized* used in the relational schema is stored in two specific tables: a table which contains all the possible values for the attribute a and a table which contains all the pairs $\{v_i, v_j\}$ of the cover relation of the partial order of $Dom_v(a)$.

Substrate	FuzzyOriginId
Pork	100

FuzzyOriginId	Country	Degree
100	USA	1.0
100	Germany	1.0
100	Italy	0.7

Fig. 4 The *left* table presents an example of relation referencing symbolic fuzzy values. The *right* table contains a part of the relation *SubstrateOrigin* which contains symbolic fuzzy sets (only one fuzzy set is represented in that example)

Substrate	SubstrateSup	SubstrateInf
Milk	Milk	FullMilk
FullMilk	Milk	PasteurizedMilk
PasteurizedMilk	PasteurizedMilk	PasteurizedFullMilk
PasteurizedFullMilk	FullMilk	PasteurizedFullMilk

Fig. 5 The *left* table presents a part of relation $Dom_{Substrate}$. The *right* table presents a part of relation $Ord_{Substrate}$

Tables presented in Fig. 5 are partial instances of relations $Dom_{Substrate}$ and $Ord_{Substrate}$ describing the hierarchized domain for substrates in the relational schema.

Remark 3 When an attribute is known to be a "crisp" value (for example the substrate in Fig. 3), the database designers have used classic database attributes of type real, integer or string instead of fuzzy values.

As presented in the two preceding sections, the ontology of the MIEL data model is stored in specific tables of the relational database schema. This is due to the fact that we have to proceed to a referential integrity control in the data we store.

2.3 The schema of the conceptual graph database

The conceptual graph model is a knowledge representation model based on labelled graphs. A lot of research has been done on this model in different kinds of applications such as Natural Language processing, information retrieval... The model we use is based on the formalization presented in Mugnier and Chein (1996). A database built on the conceptual graph model is composed of (i) a *support* which contains the terminological knowledge (mainly a partially ordered set of concept types and a partially ordered set of relation types); (ii) a set of *conceptual graphs* which conform to the support and which contain the assertional knowledge. Fig. 7 is a part of the support of the MIEL system, Fig. 10 is an example of conceptual graph built on that support.

The *specialization relation* (noted $\leq$) partially pre-orders the set of conceptual graphs. That specialization relation can be computed by means of the *projection operation* (a graph morphism allowing a restriction of the vertex labels): $G' \leq G$ if and only if there is a projection of G into G'.

Definition 2 A projection Π from a conceptual graph G into a conceptual graph G' is a pair (f, g) of mappings, f (resp. g) from the set of relation vertices (resp. concept vertices) of G to the set of relation vertices (resp. concept vertices) of G' such that: (i) the edges and their labels are preserved; (ii) the vertex labels may be restricted.

An example is given in Fig. 6. The projection is a ground operation in the conceptual graph model since it allows the search for answers, which can be viewed as specializations of a query.

The flexibility of the conceptual graph model played an important part in the choice of that knowledge representation model in the MIEL system: no static schema

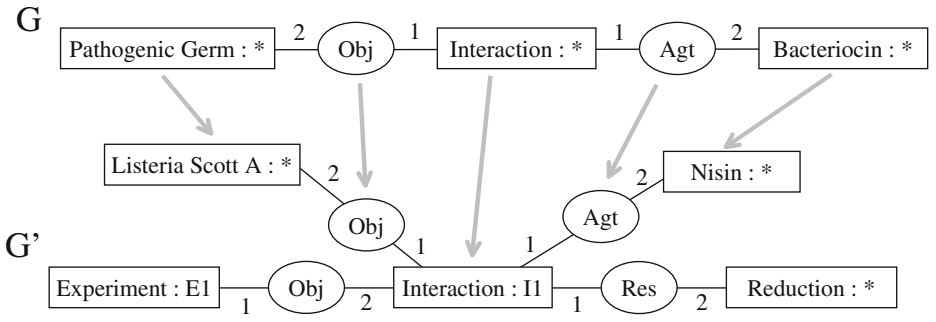

Fig. 6 There is a projection from G into G', $G' \leq G$ (G' is a specialization of G)

is used and we can build pieces of information which have different shapes by adding or removing some characteristics easily (by adding or removing graph vertices).

We present in Section 2.3.1 the way we have built the support from the MIEL ontology. Then we present respectively in Sections 2.3.2 and 2.3.3 the representation of values and data in the conceptual graph database.

2.3.1 Representation of the MIEL ontology in the conceptual graph model

2.3.1.1 The concept type set of the conceptual graph model is used to represent the main part of the ontology of the MIEL system, since it is a partially ordered set, designed to contain the concepts of a given application. The concept type set of the MIEL system is built as follows:

1. A concept type t_a is associated with each attribute a of $\mathcal{A}$;
2. If $Type(a)$ is *hierarchized*:
   a. A concept type t_{v_i} is associated with each element v_i of $Dom_v(a)$;
   b. The set of concept types composed of the t_{v_i} is partially ordered by following the partial order of $Dom_v(a)$;
   c. t_a is inserted on top of the set of the t_{v_i};
3. All distinct concept types built in step 2(a) having the same label are merged (in case the same value belongs to two distinct hierarchized variation domains);
4. All the concept types built in the previous steps are brought together into a single concept type set by adding a common super-type (*Universal*) and a common sub-type (*Absurd*).

Figure 7 represents a part of the concept type set of the MIEL conceptual graph database. The name of the attribute *Substrate* and its hierarchized variation domain presented in Fig. 1 of Section 2.1 appear as a partial subgraph of that concept type set.

2.3.1.2 The set of individual markers is used to store the variation domain of each attribute a of type *symbolic* or *numerical*. More precisely, all the values of the variation domains of the *symbolic* attributes as well as the values of $\mathbb{R}$ are inserted

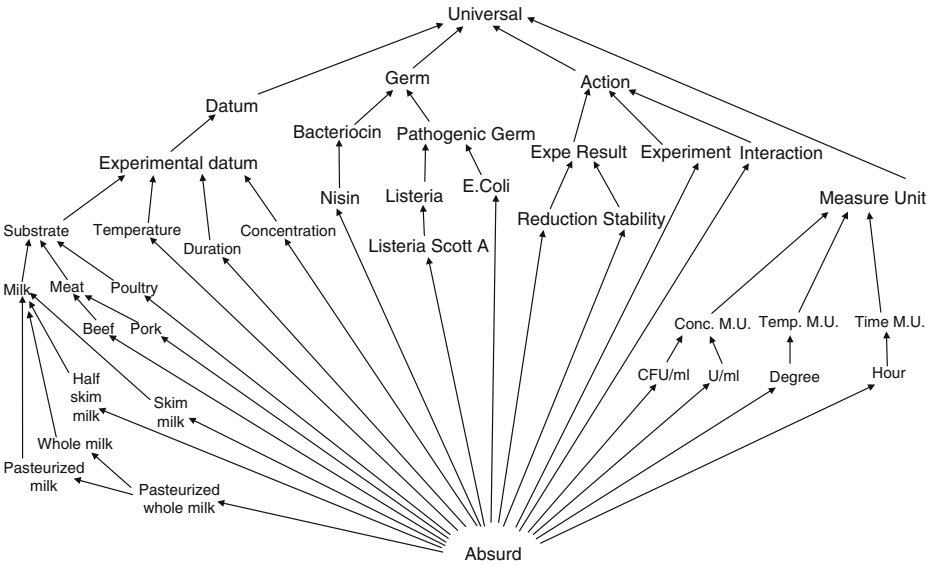

Fig. 7 A part of the concept type set of the MIEL conceptual graph database

into the set of individual markers—we proposed an extension of the conceptual graph model to the representation of numerical values in Thomopoulos et al. (2003b).

2.3.1.3 The set of relation types does not play an important part in our conceptual graph database since the semantics is mainly contained in the concept vertices. Then the set of relation types we use is composed of relation types such as "agent," "object," "characteristics." These relations correspond more or less to the grammatical function of a concept when one tries to translate a conceptual graph into natural language.

2.3.2 Representation of the values in the conceptual graph database

The values of the attributes are represented in different ways in the conceptual graph database, depending on the type of the attribute.

2.3.2.1 Representation of a numerical value If the attribute a we consider is *numerical*, and if v is its value, the pair (a, v) is represented in a conceptual graph composed of a concept vertex of type a, a concept vertex labelled by the type *NumericalValue* and the marker v, these two vertices being linked by a relation vertex labelled by *NumVal*. For example, the fact that the value 7 is associated with the attibute pH is represented by the graph in Fig. 8.

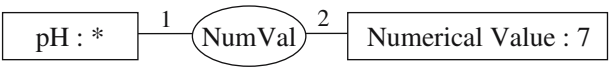

Fig. 8 A conceptual graph corresponding to the information $pH = 7$

2.3.2.2 Representation of a symbolic value If the attribute a we consider is *symbolic*, and if v is its value, the pair (a, v) is represented by a concept vertex labelled by the type a and the marker v. For example the fact that the value $M.Zwietering$ is associated with the attribute *Author* is represented by the concept vertex [Author : M.Zwietering].

2.3.2.3 Representation of a hierarchized value If the attribute a we consider is *hierarchized*, and if v is its value, the pair (a, v) is represented by a generic concept vertex labelled by the type v. For example, the fact that the value *Milk* is associated with the attribute *Substrate* is simply represented by the concept vertex [Milk : *], since *Milk* is a subtype of *Substrate* in the concept type set.

2.3.2.4 Extension of the conceptual graph model for the representation of fuzzy values In order to allow an homogeneous expressivity between the relational database and the conceptual graph database, we proposed an extension of the conceptual graph model to the representation of fuzzy values presented in Thomopoulos et al. (2003b). We only remind that extension through an example. A fuzzy set can appear in two ways in a concept vertex: (i) as a *concept with a fuzzy type*. The type is a fuzzy set defined on a subset of the concept type set; (ii) as a *concept with a fuzzy marker*. The marker is a fuzzy set defined on a subset of the set of individual markers. c_1 and c_2 presented in Fig. 9 are respective examples of these two cases.

2.3.3 Representation of data in the conceptual graph database

The conceptual graph database is composed of a set of conceptual graphs, each of them representing an elementary data. For example, Fig. 10 is a part of a conceptual graph extracted from the MIEL conceptual graph database.

At the moment, the conceptual graph database contains about 150 conceptual graphs corresponding to scientific publications which do not fit the relational database subsystem schema. These conceptual graphs have been built manually by analyzing the relevant sentences of these publications.

3 The queries in the MIEL system

An overview of the MIEL system and its global query processing is presented in Fig. 11.

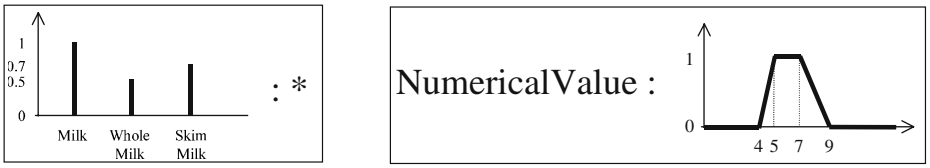

Fig. 9 Two examples of fuzzy concept vertices

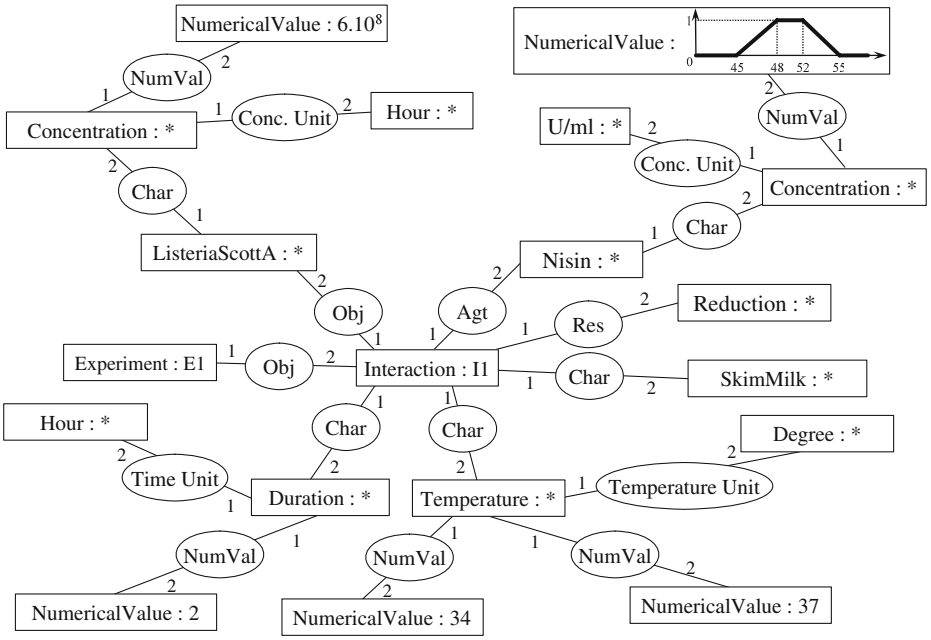

Fig. 10 An example of conceptual graph with a fuzzy concept vertex

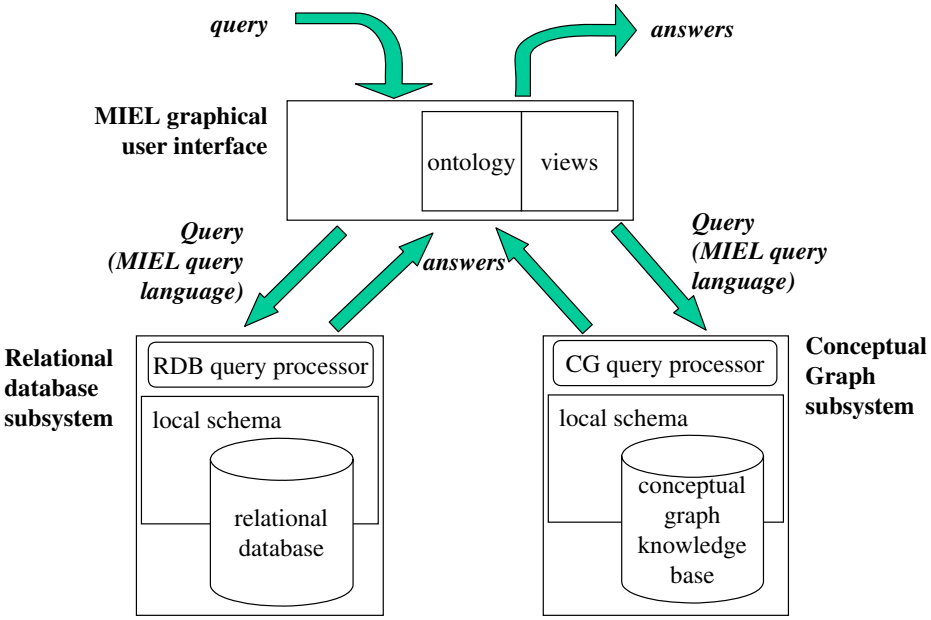

Fig. 11 Overview of the MIEL system

The MIEL graphical user interface allows the user to specify his/her query. Such a query is expressed in a view (selected by the user from a list of available views). The user also specifies in his/her query a set of projection attributes and a set of selection criteria. Then the MIEL user interface sends the query to two "subsystems," the *relational subsystem* and the *conceptual graph subsystem*. Each of these subsystems adapts the query to the formalism it uses (an SQL query in the relational subsystem, a conceptual graph query in the conceptual graph subsystem), then uses its own *query processor* in order to execute the query. Finally, the answers to the query are returned to the MIEL interface which presents them to the user in a homogeneous way.

Section 3.1 presents the MIEL uniform query language. Then Sections 3.2 and 3.3 present respectively the query definition and processing in the relational subsystem and in the conceptual graph subsystem.

3.1 The query language

A query asked on the MIEL system is expressed in the *MIEL query language* through the *MIEL graphical user interface*.

In the following, we present the notions we use in a way close to domain relational calculus (Ullman, 1988): a query Q is characterized by a set of attributes and a predicate on these attributes. We denote $Q = \{a_1, \ldots, a_n | P_Q(a_1, \ldots, a_n)\}$ the query Q characterized by the predicate P_Q on the set of attributes $(a_1, \ldots, a_n)$. We consider that the tuple $\tau = (v_1, \ldots v_n)$ is an answer to Q iff $P_Q(a_1, \ldots, a_n)$ is true when we substitute a_i for v_i, $1 \leq i \leq n$. We note $\tau[a_i]$ the value v_i of the attribute a_i in the tuple τ.

3.1.1 The notion of view

A view is a usual notion in relational databases: it is a virtual table built from the actual tables of the relational database schema by means of a query. In the MIEL system, a set of views (which are pre-written queries) is proposed to the user in order to hide the complexity of the database schema.

Definition 3 A *view* V on n $(n > 0)$ queryable attributes $a_1, \ldots, a_n$ is defined by $V = \{a_1, \ldots, a_n | P_V(a_1, \ldots, a_n)\}$ where P_V is a predicate which characterizes the construction of the view.

Example 1 The view BacteriocinInteraction is defined on 6 attributes: BacteriocinInteraction = {*PathogenicGerm*, *Bacteriocin*, *ExpeResult*, *Substrate*, *Duration*, *Temperature* | $P_{Bact.Inter.}$(*PathogenicGerm*, *Bacteriocin*, *ExpeResult*, *Substrate*, *Duration*, *Temperature*)}. The predicate $P_{Bact.Inter.}$ defines the way the attributes involved in the view are linked together. That view characterizes the experimental result of the interaction of bacteriocins (which are toxins) with pathogenic germs.

Sections 3.2.1 and 3.3.1 present respectively the mapping of that general notion of view based on a predicate into an actual implementation in the relational database and the conceptual graph database. Note that it is possible to have the same view defined in both the relational and the conceptual graph databases. Such a case means that data of a same nature are dispatched into both MIEL subsystems.

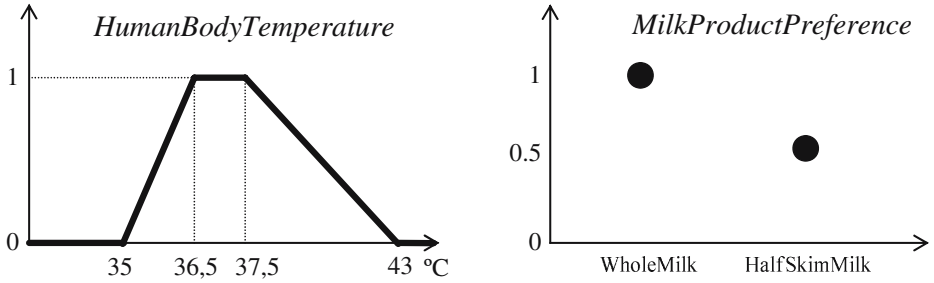

Fig. 12 Two fuzzy sets which can be used as values for selection criteria

3.1.2 Expression of a query

A query in the MIEL system is a specialization of a given view by the end-user, who specifies a set of projection attributes as a subset of the queryable attributes of the view and a set of selection criteria on some other attributes (the selection criteria and the $\approx$ operator should be defined in Section 3.1.3).

Definition 4 A query Q asked on a view V is defined by $Q = \{a_1, \ldots, a_l | \exists a_{l+1}, \ldots, a_n (P_V(a_1, \ldots, a_n) \wedge (a_{l+1} \approx v_{l+1}) \wedge \ldots \wedge (a_m \approx v_m))\}_{1 \leq l \leq m \leq n}$, where P_V is the predicate which characterizes the view V, $a_1, \ldots, a_l$ are the projection attributes, $a_{l+1}, \ldots, a_m$ are the selection attributes and their respective values $v_{l+1}, \ldots, v_m$ given as selection values by the user. The attributes $a_{m+1}, \ldots, a_n$ are the queryable attributes of the view which are not used in that query.

Example 2 $Q = \{PathogenicGerm,\ ExpeResult \mid \exists Substrate,\ Duration\ (P_{Bact.Inter.}(PathogenicGerm,\ Bacteriocin,\ ExpeResult,\ Substrate,\ Duration,\ Temperature) \wedge (Temperature \approx HumanBodyTemperature) \wedge (Bacteriocin \approx \text{“}Nisin''))\}$

The query Q expresses that the user wants to obtain the *PathogenicGerm* and the *ExpeResult* from the view *BacteriocinInteraction* when the *Temperature* is a *HumanBody - Temperature* (see Fig. 12) and the *Bacteriocin* is *Nisin*.

3.1.3 The selection criteria

As we saw previously, a database in microbiology is naturally incomplete since the number of all the possible experiments is infinite. That incompleteness may often result in empty answers to the queries. We propose to allow the user to use large selection criteria corresponding to disjunctions of searched values, which can be weighted by preferences. We propose to use the fuzzy set theory in order to express such expression of preferences. The result of the execution of a query is not a classic tuple but a tuple associated with an adequation degree to the query, corresponding to a measure of the satisfaction of the preferences expressed by the user.

Example 3 Based on the fuzzy sets presented in Fig. 12:

- A selection criterion ($Temperature \approx HumanBodyTemperature$) means that the user is interested by tuples with a Temperature value belonging to the interval

[36.5, 37.5]. He/she also accepts values from 36 to 36.5 or 37.5 to 40 with a lower preference degree;
- A selection criterion (*Substrate* $\approx$ *Milk Product Preference*) means that the user is interested by tuples with a *Substrate* value equal to *WholeMilk*. He/she also accepts *HalfSkimMilk* with a lower preference degree.

Definition 5 A selection criterion in the MIEL query language is of the form $(a \approx v)$, a being an attribute and v being an attribute value expressed by a fuzzy set.

We consider that the result of the evaluation of a selection criterion $(a \approx v)$ on a tuple τ is composed of two measures:

– The possibility degree of matching $\Pi(v, \tau[a])$ (see Zadeh, 1978). When $\Pi(v, \tau[a]) = 0$ there is no overlap between v and $\tau[a]$;
– The necessity degree of matching $N(v, \tau[a])$ (see Dubois & Prade, 1988). When $N(v, \tau[a]) = 0$ there is no inclusion of $\tau[a]$ into v ;

The selection criterion $(a \approx v)$ is *satisfied* iff $\Pi(v, \tau[a]) > 0$ (we always have $\Pi \geq N$).

In other words, a selection criterion is satisfied if there is an overlap between the selection value of the criterion and the corresponding value of the tuple.

Remark 4 The expressions of preferences allow the users to prioritize the values they searched for in their queries. In the actual MIEL system, the users simply set a partial order on the attribute values they are interested in by means of a Graphical User Interface (see Fig. 17). The MIEL system associates automatically decreasing weights with the searched values.

In the following paragraph, we present how the comparisons between fuzzy sets used in Definition 5 are computed.

3.1.4 Comparisons on fuzzy sets in the MIEL system

As we saw previously, the fuzzy sets in the MIEL database can be defined on three types of variation domains: numerical, symbolic and hierarchized. The classic comparison operators on fuzzy sets make it necessary that the fuzzy sets to be compared are defined on a same domain.

3.1.4.1 Comparisons on numerical or symbolic fuzzy sets The comparisons on numerical or symbolic fuzzy sets do not present any kind of problem since the fuzzy sets are defined on the same domain (respectively $\mathbb{R}$ or the set of symbolic constants).

3.1.4.2 Comparisons on hierarchized fuzzy sets In the special case of fuzzy sets defined on a subset of a hierarchized variation domain, it is possible to consider that degrees on the whole variation domain of the considered attribute are implicitly defined.

For example, assume the user chooses the selection criterion (*Substrate* $\approx$ *Milk Product Preference*) in a query. Such a criterion can be interpreted as: "the user wants *WholeMilk* as a substrate but he/she also accepts *HalfSkimMilk* with a lower

adequation degree." Since the domain is hierarchized by the "kind-of" relation, it is possible to consider that the user who is interested in *WholeMilk* is also interested in all the kinds of *WholeMilk* such as *WholePasteurizedMilk*.

We propose to associate a *fuzzy set closure* defined on all the variation domain with each fuzzy set defined on a subset of a hierarchized variation domain. That fuzzy set closure takes into account the implicit degrees induced by the "kind-of" relation, by propagating the degree associated with a value to all the specializations of that value.

There are special cases when the user associates different degrees with comparable values in the "kind-of" relation. We consider such a case with semantics of reinforcement when the most specific value has a greater degree than the most general one, with semantics of restriction otherwise. For example, the user can be interested in *Milk* but more particularly in *WholeMilk* (he/she can give a lower degree to *Milk* than to *WholeMilk*). On the contrary, he/she can be interested in *Milk* but with a lower preference for *WholeMilk* (in such a case, he/she gives a lower degree to *WholeMilk*).

Definition 6 Let f be a fuzzy set defined on a hierarchized attribute a on the domain $D \subseteq Dom_v(a)$ with the membership function μ. The *fuzzy set closure* f^* associated with f is defined on the whole variation domain $Dom_v(a)$. Its membership function μ^* is computed as follows:

For each element e of $Dom_v(a)$, let $M = \{u_1, \ldots, u_n\}$ be the set of most specific elements of D such that $u_k \geq e$ in the partial order on $Dom_v(a)$,

- If M is not empty, $\mu^*(e) = max_{1 \leq k \leq n}(\mu(k))$;
- Otherwise $\mu^*(e) = 0$.

For example, let $f = (1/Milk, 0.8/PasteurizedMilk, 0.6/WholeMilk)$ a fuzzy set defined on $D = \{Milk, PasteurizedMilk, WholeMilk\} \subseteq Dom_v(Substrate)$. The fuzzy set closure f^* associated with the fuzzy set f is presented in Fig. 13. The value associated with *PasteurizedWholeMilk* in f^* is computed as follows: (a) the degrees associated with elements belonging to D remain unchanged in

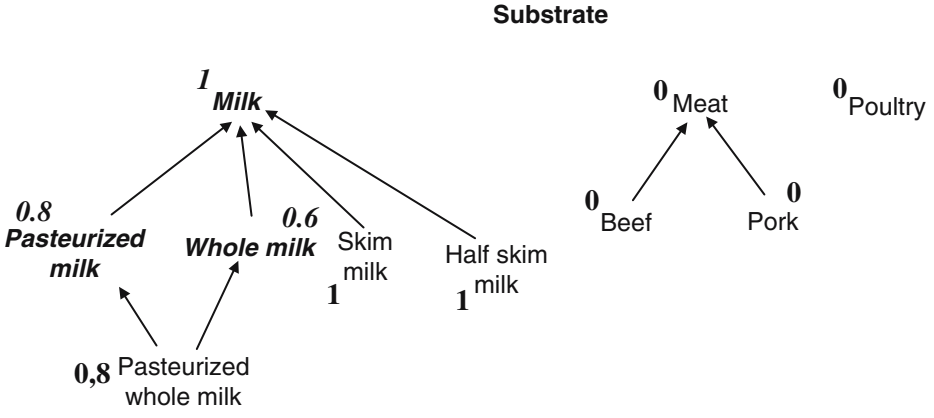

Fig. 13 The fuzzy set closure f^* corresponding to $f = (1/Milk, 0.8/Pasteurized\ milk, 0.6/Whole\ milk)$

f^*, (b) those degrees are propagated to the subtypes when there is no conflict (SkimMilk and HalfSkimMilk get the degree of Milk in f^*), (c) when there is a conflict (for example, 0.6 or 0.8 could be propagated to PasteurizedWholeMilk) the maximum degree of the most specialized generalizations of the considered element is chosen (in our example, $\{PasteurizedMilk, WholeMilk\}$ are the most specialized generalizations of $PasteurizedWholeMilk$ in $Dom_v(a)$, 0.8 is then associated with $PasteurizedWholeMilk$ in f^*), (d) 0 is associated with the other elements of f^*.

When a fuzzy set defined on a hierarchized variation domain is used as a value in a selection criterion of a query, the fuzzy set closure is computed and then used as the actual selection value used to search for satisfying answers in the database. Note that, in order to compare a selection criterion and an imprecise value represented by two fuzzy sets defined on a hierarchized variation domain, we use this mechanism of computation of the fuzzy set closure to both fuzzy sets to be compared.

3.1.4.3 Comparison operators Since after the computation of the fuzzy set closure the domains of the fuzzy sets are homogeneous, the classical comparison operators on fuzzy sets (possibility and necessity degrees of matching) can be applied (see Zadeh, 1978; Dubois & Prade, 1988).

3.1.5 The answers

An answer A to a query Q in the MIEL system is a set of tuples. Each tuple is composed of values (which are fuzzy sets as presented in Def 1). Each tuple satisfies the selection criteria (as defined in Def 5) of the query.

Definition 7 Let $Q = \{a_1, \ldots, a_l | \exists a_{l+1}, \ldots, a_n (P_V(a_1, \ldots, a_n) \wedge (a_{l+1} \approx v_{l+1}) \wedge \ldots \wedge (a_m \approx v_m))\}$ be a query. The answer A to the query Q is: $A = \{\tau_1, \ldots, \tau_r\}$, the set of tuples of the form $\{\tau_i[a_1], \ldots, \tau_i[a_l], n_i, \pi_i\}_{1 \leq i \leq r}$, such that every tuple of A satisfies all the selection criteria of Q, with $n_i = min_{i=l+1,\ldots,m} N(v_i, \tau[a_i])$ and $\pi_i = min_{i=l+1,\ldots,m} \Pi(v_i, \tau[a_i])$ – π_i being strictly positive – their respective necessity and possibility degrees of matching.

Note that the *min* operator is classically used as an aggregator when the query expresses a conjunction of elementary requirements (Dubois & Prade, 1995).

In order to be able to sort the answer tuples following the preferences expressed by the users in their selection criteria, we have to characterize the best-matching tuples. As proposed in Dubois and Prade (1995), we can consider that the necessity degree is of greater importance than the possibility degree, because when the necessity degree is positive, we are (more or less) certain that the item matches the requirement. A way of ranking the answer tuples could be to sort the tuples on the values of their n and then on their π in case the n values are equal.

Nevertheless, our system is meant for microbiologists, and we thought that it was valuable to associate a single ranking value with each tuple instead of a pair composed of a necessity and a possibility degrees. That value which we call *adequation degree* of the tuple to the query is defined as follows for a tuple τ_i: $\delta_i = \frac{n_i + \pi_i}{2}$.

Since we always have $N(v_i, \tau[a_i]) \leq \Pi(v_i, \tau[a_i])$ (Dubois & Prade, 1995), we obviously have $n_i \leq \pi_i$ for all the tuples τ_i. Thus we know that δ_i is an upper bound

Fig. 14 A partial instance of relation *LViews*

IdView	FromPart	SelectPart	WherePart
SubstrateList	Publication P, Substrate S, FuzzyTemp F	P.Title, S.Substrate	P.IdPub=S.IdPub and S.FuzzySetId=F.FuzzySetId

for n_i and a lower bound for π_i, and can be viewed as a synthesis of these two values. Moreover, in the classic case of a database containing precise data, n_i and π_i are equal. These remarks led us to use δ_i as a unique ranking value for the tuples of the answer.

The drawback of that unique ranking value is that our system makes no difference between a tuple τ_i with $n_i = 0.6$ and $\pi_i = 1$ and another tuple τ_j with $n_j = \pi_j = 0.8$: $\delta_i = \delta_j = 0.8$.

To summarize, let us consider some special values of δ and their respective meanings. Let τ_i be an answer tuple to a query with its adequation degree δ_i:

- $\delta_i = 1$ means that every selection criterion of the query has been satisfied with a necessity degree (and then a possibility degree) of matching equal to 1. In other words, there is a "strong" inclusion of all the attribute values of the tuple in all the values of the selection criteria, since the supports of the tuple values are included in the kernels of the values of the respective selection criteria;
- $\delta_i > 0.5$ means that $n_i > 0$: the minimal necessity degree computed for the different selection criteria of the tuple implies that there exists an inclusion[2] for all the values of the selection criteria and their respective values in the tuple τ_i;
- $\delta_i < 0.5$ means that $\pi_i < 1$: the minimal possibility degree computed for the different selection criteria of the tuple implies that there is no overlap[3] of the kernels for all the values of the selection criteria and their respective values in the tuple τ_i;
- $\delta_i = 0$ is impossible, since a selection criterion s is considered to be satisfied iff $\pi_s > 0$ (see Definition 5).

3.2 Query definition and processing in the relational subsystem

3.2.1 Views in the relational database

The views in the relational database of the MIEL system are SQL queries. In the actual implementation of the MIEL system, the views are stored in a specific table of the database called *LViews*, in which each tuple represents a view and is composed of four columns: *IdView* is the unique identifier of the view, *SelectPart* is the list of projection attributes, *FromPart* is the list of relations involved in the view, *WherePart* is the list of join predicates between those relations.

[2] In the meaning of the necessity degree of matching.

[3] In the meaning of the possibility degree of matching.

Example 4 The SQL query which corresponds to the view SubstrateList defined in Fig. 14 is : select P.Title, S.Substrate from Publication P, Substrate S, FuzzyTemp F where P.IdPub = S.IdPub and S.FuzzySetId=F.FuzzySetId.

3.2.2 The relational database query processing

The query processing in the database subsystem is processed as follows:

1. Selection of the view corresponding to the query;
2. Transformation of the fuzzy values of the selection criteria into classic SQL conditions (we call that process "defuzzification");
3. Completion of the SQL query corresponding to the view in order to build the actual "defuzzified" SQL query;
4. Subsmission of the SQL query to a standard relational database management system (ORACLE in the present version);
5. Computation of the adequation degree of each tuple of the answer.

For example, assume the query asked through the MIEL graphical user interface is: $\{Title, Substrate | SubstrateList(Title, Substrate) \wedge (Temperature \approx Human\ BodyTemperature)\}$. The selected view is that of Example 4. The defuzzification of the selection criterion leads to the actual selection criterion: $(F.MinSupp < 43)$ *and* $(F.MaxSupp > 35)$: 35 and 43 are respectively the lower and the upper bounds of the support of the fuzzy value HumanBodyTemperature; that "defuzzified" condition ensures there is an overlap between the two fuzzy sets and thus a satisfaction of the selection criterion.

The SQL query submitted to the ORACLE query processor is then: *select P.Title, S.Substrate from Publication P, Substrate S, FuzzyTemp F where P.IdPub = S.IdPub and S.FuzzySetId=F.FuzzySetId and ((F.MinSupp <43) and (F.MaxSupp >35))*.

3.3 Query definition and processing in the conceptual graph subsystem

3.3.1 The views

The conceptual graph subsystem relies on a set of *view graphs* which allow us to define views on the conceptual graph database.

Definition 8 A view graph S associated with a view V on n queryable attributes $\{a_1, \ldots, a_n\}$ is a pair $\{G, C\}$ where G is an acyclic conceptual graph and $C = \{c_1, \ldots, c_n\}$ is a set of distinct concept vertices of G corresponding to the queryable attributes of the view. The type of each c_i is the concept type associated with the attribute a_i (as presented in Sections 2.3.2.1–2.3.2.3).

The graph presented in Fig. 15 is a view graph for the view *BacteriocinInteraction*, the concepts of C are framed in bold.

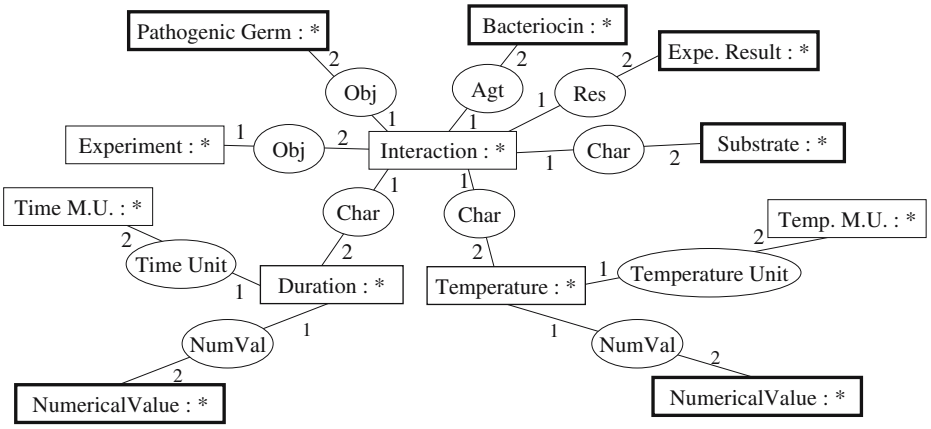

Fig. 15 An example of a view graph for the view BacteriocinInteraction

3.3.2 The queries

When a query is asked in the conceptual graph subsystem, the view graph corresponding to the considered view is specialized by the "instantiation" of concept vertices in order to take into account the selection attributes, resulting in a *query graph*.

Definition 9 Let $(a \approx v)$ be a selection criterion and $S = \{G, C\}$ a view graph to be instantiated in order to build a query graph. Let c be the concept vertex corresponding to the attribute a in C. The specialization of G in order to obtain a query graph is done as follows:

- If $Type(a)$ is symbolic, then the generic marker of c is replaced by the individual marker v;
- If $Type(a)$ is numerical, then the generic marker of the concept vertex of type *NumericalValue* linked to c by a relation vertex *NumVal* is replaced by the individual marker v;
- If $Type(a)$ is hierarchized, then the type of the concept vertex c is restricted to the concept type labelled by v, which has been inserted in the concept type set (Cf. Section 2.3.1).

Example 5 The query graph presented in Fig. 16, which is a specialization of the view graph presented in Fig. 15, corresponds to the query Q of Example 2 presented in Section 3.1.2. Since the type of the selection attribute *Temperature* is not hierarchized, the specialization is done by means of the definition of a marker in the concept vertex [NumericalValue : HumanBodyTemperature] (the marker is a fuzzy one in that example). The query graph is instantiated differently for the selection attribute (*Bacteriocin* $\approx$ *Nisin*): the type of the selection attribute is hierarchized, a restriction of the corresponding concept type to its subtype "Nisin" is done.

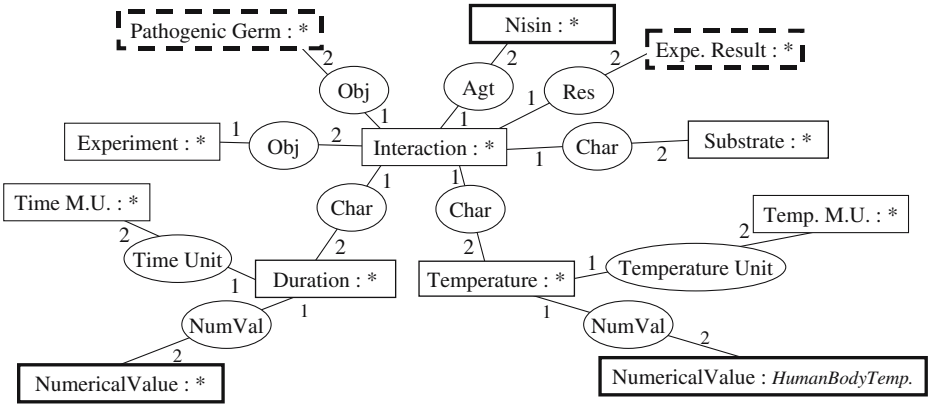

Fig. 16 An example of a query graph. The election attributes are framed in bold, the projection attributes are dashed

3.3.3 Query processing

In Thomopoulos, Buche and Haemmerlé (2003a), we proposed different kinds of comparisons between fuzzy conceptual graphs (like specialization or more flexible comparisons). In the conceptual graph subsystem of the MIEL system, the query processing consists in searching for conceptual graphs which contain a more precise information than the information contained in the query (we search for specializations of the query graph) or, at least, for conceptual graphs which contain "approximate" answers. In order to find such conceptual graphs, we propose to use the δ-projection operation which is a flexible operation of mapping between two conceptual graphs.

Definition 10 A δ-projection Π from a conceptual graph G into a conceptual graph G' is a triple (f, g, δ), f (resp. g) being a mapping from the set of relation (resp. concept) vertices of G to the set of relation (resp. concept) vertices of G' such that: (a) the edges and their labels are preserved; (b) the labels of the relation vertex may be restricted; (c) each concept $c \in G$ labelled $\boxed{t : m}$ is mapped with its image $g(c) \in G'$ labelled $\boxed{t' : m'}$ with a necessity and a possibility degrees of matching $n = min(N(t, t'), N(m, m'))$ and $\pi = min(\Pi(t, t'), \Pi(m, m'))$. The necessity and possibility degrees of matching of G and G' are then computed in the same way as in Definition 7; their adequation degree denoted by δ is computed as presented in Section 3.1.5.

Note that the question of the existence of a classic projection of a graph into another graph is NP-complete. However there are polynomial cases. Our algorithm of δ-projection is based on a polynomial algorithm which questions the existence of a projection of an acyclic graph into a graph (Mugnier & Chein, 1992). The only difference is the comparison operator we use between concept vertices. But we show in Thomopoulos et al. (2003a) that our algorithm remains polynomial. That is the reason why we limit the view graphs (and then the query graphs) to acyclic conceptual graphs (see Definition 8).

The query processing in the conceptual graph subsystem consists in selecting the view graph, building the query graph, and δ-projecting that query graph into all the conceptual graphs of the database. Every time a δ-projection into a fact graph A_G with an adequation degree δ is found, the conceptual graph query processor considers that A_G is an answer graph. A tuple with the adequation degree δ is built from that answer graph by extracting the values of the projection attributes.

For example, if we asked the query in Fig. 16 on a conceptual graph database containing the conceptual graph in Fig. 10, the resulting tuple would be: $('ListeriaScottA','Reduction', \delta = 1)$.

4 Conclusion and perspectives

In this article we have presented some innovative work concerning the extension of the conceptual graph model to the representation of fuzzy values, or concerning the definition of fuzzy sets on hierarchical definition domains. A major contribution of our work is the implementation of the MIEL system, which provides all the mechanisms presented in this paper. The interface of the MIEL system, developed in Java, is meant to be used through an Internet browser. When a query is asked through the MIEL interface, that query is transferred to both subsystems. The answers to the user's query take the opposite way in order to be presented in a uniform manner in the user's browser. The relational subsystem has been implemented as a Java query processor interacting with an Oracle relational database. It is composed of about 90 tables, with data extracted from about 700 microbiological publications. The conceptual graph subsystem has been implemented in C++ under Linux; it is composed of a 5.000-line extension of the CoGITaNT platform (Genest & Salvat, 1998). The conceptual graph database contains about 150 conceptual graphs. The communication between both subsystems and the MIEL interface is done through the TCP protocole via a communication module, written in C.

Figures 17 and 18 show respectively the window allowing the expression of a query in the MIEL system and the window presenting the results to the user.

The MIEL system has been successfully presented to our microbiologist partners and is now operational Buche, Dervin, Haemmerlé, Surleau and Thomopoulos (2003). The technical centers belonging to the Sym'Previus project are currently working on a business plan which aims at commercializing a new expert tool in food predictive microbiology, which includes the MIEL system and a simulation tool (Leporq, Membré, Dervin, Buche & Guyonnet, 2005).

The conceptual graph model appears to be an asset for the MIEL system since it allows non-specialists to add heterogeneous data into our database. The users have been impressed by the simplicity of use of the graphical user interface of the CoGITaNT platform which allows one to acquire conceptual graphs. But it is not always easy to build a conceptual graph out of nothing so we are currently thinking about an interface providing pre-written graph patterns. However that may be, the transparency of the knowledge integration mechanism is really appreciated by the end-users.

An important goal of our system is to provide relevant answers when no exact answer can be found. The MIEL system provides several ways of enlarging the querying mechanism already implemented in the relational database (Buche, Dervin,

Haemmerlé, & Thomopoulos, 2005) and in the conceptual graph subsystem (Buche & Haemmerlé, 2000). The enlargement mechanisms work by generalizing the query. That generalization is done in different ways. It is possible to enlarge the fuzzy sets appearing in the selection criteria (for example, if we search for a temperature between 36 and 38°C, the system can enlarge the interval to 35–39). It is also possible to relax a constraint by taking advantage of the hierarchized variation domains of the attributes (if the user searches for experiments involving *HalfSkimMilk*, it can be useful to return answers concerning all kinds of *Milk* with a decreasing adequation degree in addition to answers concerning *HalfSkimMilk*). Other ways of generalization of the query can be studied, for example by relaxing the structure of the conceptual graph query.

In the MIEL system, both relational and conceptual graph bases are queried simultaneously by a unified querying mechanism. Our approach can be compared to the knowledge integration problematics (Ordille, Levy, & Rajaraman, 1996; Genesereth, Keller & Duschka, 1997; Goasdoué, Lattes, & Rousset, 2000). We have defined a uniform query language (the MIEL query language) which uses an ontology (the set of queryable attributes and their respective variation domains which can be hierarchized by the "kind-of" relation) and a set of views on the database. The ontology and the set of views can be considered as the "mediated schema" of our knowledge integration system. One of the major differences between our system and a classic information integration system based on mediators is that each tuple of an answer to a query in the MIEL system comes from a single data source. There is no declarative description of the different data sources in the mediated schema allowing the system to rewrite a global query into several subqueries intended for the different data sources. In the MIEL system, each query is "totally" asked on each data source, which translates the query expressed in the MIEL query language into a query well-suited for its query processor (an SQL query

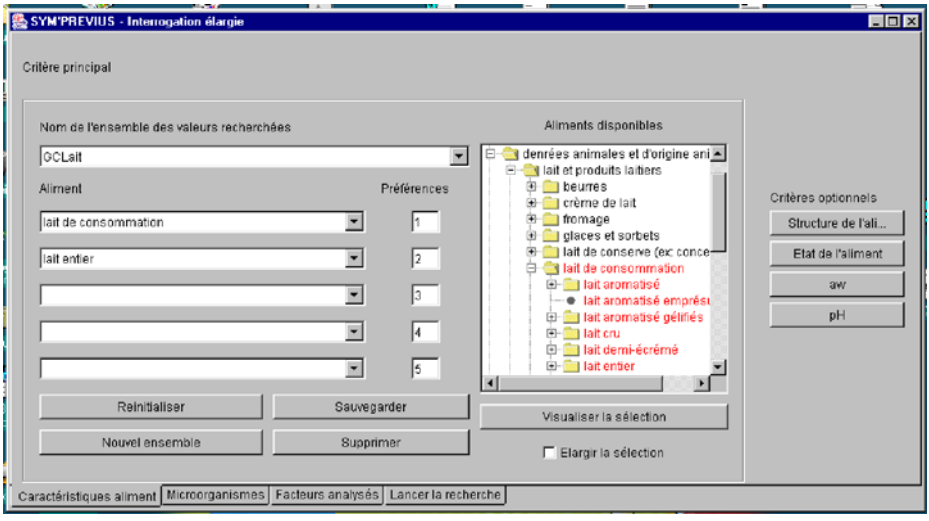

Fig. 17 The graphical user interface used to express a query (in French). The ontology is accessible to the user by the central hierarchy

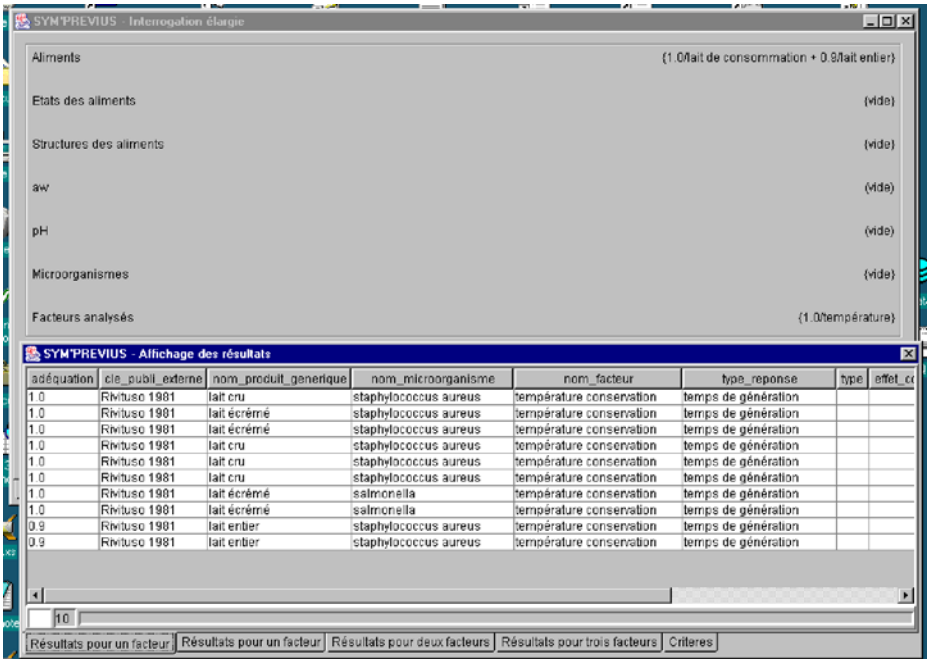

Fig. 18 The graphical user interface used to display the result of a query execution. The results are displayed by decreasing adequation degree

or a conceptual graph query). The global answer to a query in the MIEL system is then the union of the local answers given by the two subsystems. Nevertheless, we can consider that the MIEL system has become a framework allowing us to integrate different kinds of databases. We are now working on the integration of several relational databases existing on our domain of application, but which do not have a schema nor an ontology compatible with ours. We are also working at the integration of a new subsystem based on an XML query processor since we think that XML will become a standard for the representation of data on the Web. We also consider XML as a useful language for the automatic or semi-automatic translation of documents such as spreadsheets. That work is done in the framework of the "e.dot" project, which involves two computer science laboratories and a company.[4] The goal is to build a data warehouse composed of our bases, complemented by data extracted from the Web.

Acknowledgements We would like to thank Marie-Christine Rousset, Juliette Dibie-Barthélemy and Marion Rougier for their careful reading of this paper. This work is partially supported by the French Ministries of Agriculture and Research and by the Sym'Previus consortium.

[4]GEMO Project (INRIA Futurs / Paris XI University), INA P-G and Xyleme.

References

Ballows, A., Truper, H., Dworkin, M., Harder, W., & Schleifer, K. (Eds.) (1992) *The prokaryotes, a handbook on the biology of bacteria: Ecophysiology, isolation, identification, applications* (2nd ed.). Berlin Heidelberg New York: Springer.

Bosc, P., & Pivert, O. (1995) SQLf: A relational database language for fuzzy querying. *IEEE Transactions on Fuzzy Systems*, *3*(1), 1–17.

Buche, P., Dervin, C., Haemmerlé, O., Surleau, E., & Thomopoulos, R. (2003). Carrying out the microbial risk in food products using the MIEL software: A new tool to query incomplete, imprecise and heterogeneously structured experimental data stored in a relational database. In *Proceedings of the International Conference on Predictive Modelling in Food, ICPMF'03* Quimper, France (pp. 58–60). Berlin Heidelberg New York: Springer.

Buche, P., Dervin, C., Haemmerlé, O., & Thomopoulos, R. (2005). Fuzzy querying on incomplete, imprecise and heterogeneously structured data in the relational model using ontologies and rules. *IEEE Transactions on Fuzzy Systems*, *13*(3), 373–383.

Buche, P., & Haemmerlé, O. (2000) Towards a unified querying system of both structured and semi-structured imprecise data using fuzzy views. In *Proceedings of the 8th international conference on conceptual structures, Lecture notes in artificial intelligence #1867* Darmstadt, Germany (pp. 207–220). Berlin Heidelberg New York: Springer.

Cao, T. (1999). Foundations of order-sorted fuzzy set logic programming in predicate logic and conceptual graphs. Ph.D. thesis, University of Queensland, Australia.

Dubois, D., & Prade, H. (1988). *Possibility theory—An approach to computerized processing of uncertainty*. New York: Plenum.

Dubois, D., & Prade, H. (1995). Tolerant fuzzy pattern matching: An introduction. In P. Bosc & J. Kacprzyk (Eds.), *Fuzziness in Database Management Systems* (pp. 42–58). Berlin Heidelberg New York: Springer.

Dubois, D., Prade, H., & Rossazza, J. (1991). Vagueness, typicality and uncertainty in class hierarchies. *International Journal of Intelligent Systems*, *6*, 167–183.

Galindo, J., Cubero, J., Pons, O., & Medina, J. (1998). A server for fuzzy SQL queries. In *Proceedings of the 1998 workshop FQAS'98 (Flexible query-answering systems)*, Roskilde, Denmark (pp. 161–171). Berlin Heidelberg New York: Springer.

Genesereth, M., Keller, A., & Duschka, O. (1997). Infomaster: An information integration system. In *Proceedings of SIGMOD 97* (pp. 539–542). New York: ACM.

Genest, D., & Salvat, E. (1998). A platform allowing typed nested graphs: how CoGITo became CoGITaNT. In *Proceedings of the 6th international conference on conceptual structures (ICCS'1998), Lecture notes in artificial intelligence #1453*, Montpellier, France (pp. 154–161). Berlin Heidelberg New York: Springer.

Ginsburg, S., & Hull, R. (1983). Order dependency in the relational model. *Theoretical Computer Science*, (26), 149–195.

Goasdoué, F., Lattes, V., & Rousset, M.-C. (2000). The use of CARIN language and algorithms for information integration: The PICSEL system. *International Journal of Cooperative Information Systems*, *4*(9), 383–401.

Ireland, J., & Moller, A. (2000) Review of international food classification and description. *Journal of Food Composition and Analysis*, *13*(4), 529–538.

Leporq, B., Membré, J., Dervin, C., Buche, P., & Guyonnet, J. (2005). The "Sym'Previus" software, a tool to support decisions to the foodstuff safety. *International Journal of Food Microbiology*, *100*(1–3), 231–237.

Morton, S. (1987). Conceptual graphs and fuzziness in artificial intelligence. Ph.D. thesis, University of Bristol, UK.

Mugnier, M., & Chein, M. (1992). Polynomial algorithms for projection and matching. In *Proceedings of the 7th annual workshop on conceptual graphs, Lecture notes in artificial intelligence #754*, Las Cruces, New Mexico (pp. 239–251). Berlin Heidelberg New York: Springer.

Mugnier, M., & Chein, M. (1996). Représenter des connaissances et raisonner avec des graphes. *Revue d'intelligence Artificielle*, *10*(1), 7–56.

Ordille, J., Levy, A., & Rajaraman, A. (1996). Querying heterogeneous information sources using source descriptions. In *Proceedings of the international conference on very large data bases* (pp. 251–262). San Francisco, California: Morgan Kaufmann.

Prade, H., & Testemale, C. (1984). Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries. *Information Sciences*, *34*, 115–143.

Sebastiani, F. (1994). A probabilistic terminological logic for modelling information retrieval. In *Proceedings of the 17th annual international ACM-SIGIR conference on research and development in information retrieval*, Dublin, Ireland (pp. 122–130). Berlin Heidelberg New York: Springer.

Sowa, J. (1984). *Conceptual structures—Information processing in mind and machine*. Reading, Massachussetts: Addison-Welsey.

Thomopoulos, R., Buche, P., & Haemmerlé, O. (2003a). Different kinds of comparisons between fuzzy conceptual graphs. In *Proceedings of the 11th international conference on conceptual structures, ICCS'2003, Lecture notes in artificial intelligence #2746*, Dresden, Germany (pp. 54–68). Berlin Heidelberg New York: Springer.

Thomopoulos, R., Buche, P., & Haemmerlé, O. (2003b). Representation of weakly structured imprecise data for fuzzy querying. *Fuzzy Sets and Systems*, *140-1*, 111–128.

Ullman, J. (1988). *Principles of database and knowledge-base systems*. Rockville, Maryland: Computer Science.

Umano, M. (1982). , Chapt. FREEDOM-0: a fuzzy database system. In M. Gupta, & E., Sanchez E. (Eds.), *Fuzzy Information and Decision Processes* (pp. 339–347). Amsterdam, The Netherlands: North-Holland.

Zadeh, L. (1965) Fuzzy sets. *Information and Control*, *8*, 338–353.

Zadeh, L. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, *1*, 3–28.

Zadrozny, S., & Kacprzyk, J. (1998). Implementing fuzzy querying via the internet/WWW: Java applets, activeX controls and cookies. In *Proceedings of the workshop FQAS'98 (Flexible query-answering systems)* Roskilde, Denmark (pp. 358–369). Berlin Heidelberg New York: Springer.