Fuzzy querying of incomplete, imprecise and heterogeneously structured data in the relational model using ontologies and rules

P. Buche, C. Dervin, O. Haemmerlé and R. Thomopoulos

UMR BIA INA P-G/INRA

16, rue Claude Bernard,

F-75231 Paris Cedex 5, France

Patrice.Buche@inapg.inra.fr, tel (+33) 1 44 08 16 75

February 3, 2004

Abstract

In this paper, we present a new method, called multi-view fuzzy querying, which permits to query incomplete, imprecise and heterogeneously structured data stored in a relational database. This method has been implemented in the MIEL software. MIEL is used to query the Sym'Previus database which gathers information about the behaviour of pathogenic germs in food products. In this database, data are incomplete because information about all possible food products and all possible germs is not available; data are heterogeneous because they come from various sources (scientific bibliography, industrial data, etc); data may be imprecise because of the complexity of the underlying biological processes that are involved. To deal with heterogeneity, MIEL queries the database through several views simultaneously. To query incomplete data, MIEL proposes to use a fuzzy set, expressing the query preferences of the user. Fuzzy sets may be defined on a hierarchized domain of values, called an ontology (values of the domain are connected using the *a kind of* semantic link). MIEL also proposes two optional functionalities to help the user query the database: (i) MIEL can use the ontology to enlarge the querying in order to retrieve the nearest data corresponding to the selection criteria, (ii) MIEL proposes fuzzy completion rules to help the user formulate his/her query. To query imprecise data stored in the database with fuzzy selection criteria, MIEL uses fuzzy pattern matching.

Keywords: fuzzy querying, fuzzy database, ontologies, query completion rules, heterogeneous data.

1 Introduction

Our team is involved in a national project, called Sym'Previus, which aims at building a tool to help experts in the field of predictive microbiology. This tool is composed of two subsystems: (1) a database which gathers knowledge about the behaviour of pathogenic germs in food products, (2) a simulation tool. This paper is dedicated to the database subsystem. Knowledge stored in this database is characterized by several specific properties: information is incomplete, imprecise and weakly-structured. Information is incomplete by nature because the database will never contain information about all the possible food products and all the possible pathogenic germs. It is imprecise because of the complexity of the biological processes involved. It is weakly-structured because the information comes from heterogeneous sources (bibliographical sources, industrial data, ...) and because the knowledge about predictive microbiology, which is still a field of research, is evolving rapidly. Together with our partners of the Sym'Previus project, we have designed a relational database to store the data. Due to the heterogeneity of the data, the database schema is complex.

In this paper, we propose a new system to query a relational database which retrieves the most relevant heterogeneously structured information according to the selection criteria of the query. We have made the choice first to query the database using fuzzy values expressing the user's preferences and on the other hand to represent imprecise data stored in the database as possibility distributions (see section 2). In this context, the central point of our work is to propose a way of helping any user to make a fuzzy query to be performed on a complex database schema. This is achieved by the concept of multi-view fuzzy query. This notion is an extension of the classical view concept in databases, e.g. a virtual table (1) which hides the complexity of the schema from the user, (2) in which all the information needed by the user is brought together. The query is multi-view because the fuzzy query is processed simultaneously in several views in order to take into account the heterogeneity of the database structure. This query is fuzzy because the selection criteria are expressed as fuzzy predicates as in [24] and [14]. *An important specificity of our fuzzy query language is that the selection criteria may be defined on domains where values are partially ordered according to the "a kind of" relation. We call such a domain, an ontology.*

Five original advantages are provided by this multi-view fuzzy query system and are listed in the following. As we said previously, the engine is able to :

- process the fuzzy query in several views to retrieve information heterogeneously structured in the database;
- define precisely preferences expressed as fuzzy sets by the user on ontologies. To do that, we propose an unambiguous semantics and definitions of a fuzzy set defined on an ontology;

The engine provides additional optional assistance to the user to make a fuzzy query:

- a query completion with pertinent values thanks to a rule base;
- an automatic enlargement of the user's preferences using ontologies to retrieve additional pertinent information;

The fifth advantage is that the engine is able to scan all existing RDBMS which implement SQL, unlike previous systems which depend on a given RDBMS (see section 4).

Section 2 presents the querying language and the semantics of fuzzy sets defined on an ontology. Section 3 describes the fuzzy query processing. Section 4 gives some details about the implementation of the engine. We conclude this paper and give the perspectives of this work in section 5.

2 Fuzzy querying language

Our work can be related to two types of works. In a first category of study, the fuzzy set framework has been shown to be a sound scientific choice to model flexible queries [3].

It is a natural way of representing the notion of preference using a gradual scale. In [6], the semantics of a language called SQLf has been proposed to extend the well-known SQL language by introducing fuzzy predicates processed on crisp information. In this framework, two main strategies have been proposed to implement the evaluation of the query [7]: (i) the first one, called derivation, assumes that a threshold (λ) is associated with the fuzzy query in order to retrieve its λ-level cut [5, 10]; (ii) the second strategy rests on two steps: firstly, the breakdown of the query into elementary operations and secondly, the evaluation of elementary operations by means of specific algorithms [7, 16, 15, 24]. We did not focus on works studying the first strategy because they only retrieve a λ-level cut, losing graduality in the response. Using the second strategy, the FQUERY97 system described in [24] has been implemented under the Microsoft Access graphical environment. Answers are ordered according to a matching degree which is a result of the comparison between fuzzy predicates and crisp values. [15] proposes a querying system using a comparison process where fuzzy logic is used to generalise equality to similarity. In a second category of works, the fuzzy set framework has also been proposed to represent imprecise values by means of possibility distributions [23]. Several authors have developed this approach in the context of databases [18, 19, 4, 2, 22, 21]. The FSQL system prototype presented in [14] has been developed in the Oracle 7 relational database management system (RDBMS). The imprecise information represented as possibility distribution is stored in standard tables. The user has to write a fuzzy query using the FSQL language. This query is then translated once only into a standard SQL query that calls on functions provided by the FSQL system to compute the degrees of matching.

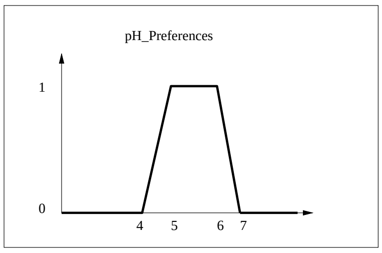
In the previous works which propose fuzzy querying systems such as FSQL or FQUERY97, the user has to build himself the query flexibility; for instance, in FSQL, the user has to specify in his/her query whether he/she is using a fuzzy join or a standard one. Those systems are more or less aimed at computer science specialists. As we mentioned in the introduction, the aim of our system is to help any user to make a fuzzy query against a complex database schema. In the definition of our fuzzy querying language in 2.1, we propose two original specificities:

- in order to manage the search in an heterogeneous schema, the querying language permits the processing of a fuzzy query in several views,
- *in order to obtain an automatic enlargement of the querying, the querying language permits the use of similarity relationships calculated on the domain of values (we compare our approach to that of [15] at the end of section 3.3.2).*

To the best of our knowledge, the works in our bibliography about the use of fuzzy sets in databases do not particularly study fuzzy sets defined on a hierarchized domain of values. Therefore, we propose in 2.2 two original definitions of fuzzy sets (intentional and extensional) defined on a hierarchized domain of values.

2.1 Language definition

The queries are expressed in terms of a set of views and a set of conjunctive selection criteria of the form attribute/value. A list of projection attributes is expressed in each view (which is a classic concept in databases, e.g. a virtual table in which all the information needed by the user is brought together).



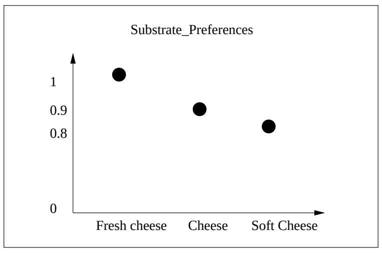


Figure 1: Fuzzy sets pH_Preferences and Substrate_Preferences

Definition 1 *A query* Q *is a set* $\{< V_1, a_{11}, \ldots, a_{1n} >, \ldots, < V_m, a_{m1}, \ldots, a_{mn'} >, < a_1, v_1, b_1 >, \ldots, < a_l, v_l, b_l >, nb, t\}$ *where* $V_1, \ldots, V_m$ *are the names of the views in which the query is asked;* a_{ij} *are the projection attributes in* V_i*,* $< a_1, v_1, b_1 >, \ldots, < a_l, v_l, b_l >$ *are triples defining the selection criteria common to* $V_1, \ldots, V_m$*;* nb*,* t *respectively define the maximum number of tuples and the minimum degree of matching of each tuple in the result (*$t \in [0, 1]$*). The triples defining the selection criteria have the following meaning:*

$\forall i \in [1, l]$

- a_i *is a selection attribute common to* $V_1, \ldots, V_m$ *and* v_i *is the value associated with the selection attribute* a_i*.* v_i *is a fuzzy set on the underlying domain* D*. Its membership function* μ *is defined by* $\mu_{v_i} : D \rightarrow [0, 1]$*. We distinguish two kinds of fuzzy sets depending on the underlying domain of the attribute (numeric or symbolic). Two examples of fuzzy sets are given in fig. 1.*

- b_i *is a Boolean value specifying to the engine whether the fuzzy set may be enlarged or not (see section 3.3).*

Here is an example of a query $Q = \{<$ *OneFactorExperience, Substrate, Germ,* pH_{min}*,* pH_{max}*, Factor, ResponseType* $>$, $<$ *TwoFactorsExperience, Substrate, Germ,* pH_{min}*,* pH_{max}*,* *Factor1, Factor2, ResponseType>, <Substrate, Substrate_Preferences, false>, <pH, pH_Preferences, false>, 10, 0.2*}.

The answer corresponding to a given query is composed of a set of elementary answers as defined below.

Definition 2 *An answer A to a query Q is a set of elementary answers $A_{V_1}, \ldots, A_{V_m}$. Each elementary answer A_{V_i}, corresponding to the view V_i, is a set of tuples, each of them of the form $\{rd, <a_{i1}, v_{i1}>, \ldots, <a_{in}, v_{in}>\}$ with $a_{i1}, \ldots, a_{in}$ the attributes of the projection in V_i, $v_{i1}, \ldots, v_{in}$ the values (precise or imprecise) resulting from the execution of the query associated with each attribute of the projection in V_i and rd the relevance degree associated with each tuple.*

The result of the query is a set of fuzzy sets. Each of those fuzzy sets is defined on the Cartesian product of the attributes of a given view. In each fuzzy set, each element is a tuple composed of couples $< attribute, value >$. Tuples are ordered according to a relevance degree rd which will be defined in section 3.1.2. A part of an answer corresponding to the view *OneFactorExperience*, associated with the example of query Q defined above, is given in table 1. Note that the attribute pH is an imprecise value stored in the database and represented as an interval in two columns *min* and *max*. For each selection attribute, the user may authorize the system to enlarge his/her preferences. This enlargement corresponds to a generalization of his/her preferences. For instance, in the previous example Q, if the triplet <*Substrate, Substrate_Preferences, false*> is replaced by <*Substrate, Substrate_Preferences, true*>, the system enlarges the fuzzy set $Substrate\_Preferences$ by using a fuzzy tolerance coefficient stored in the knowledge base (see section 3.3.1). The three data in table 1 still

Table 1: Part of the answer corresponding to the view *OneFactorExperience*

rd	Substrate	Germ	pH min	pH max	Factor	Response type
1.0	Pasteurized fresh cheese	Bacillus Cereus	5.1	5.2	Temperature	Temporal cinetic
0.9	Melted cheese	Listeria	5.0	5.4	Temperature	Level of contamination
0.8	Camembert	Listeria	6.0	6.0	Temperature	Level of contamination

appear in the result of the query with enlargement. In addition, a new tuple appears in the answer: {*rd=0.5, <Substrate, Milk for consumption>, <Germ, Listeria>, <*pH_{min}*, 5.1>, <*pH_{max}*, 5.2>, <Factor, Temperature>, <ResponseType, Temporal cinetic >* } . Detail about the computation of the enlargement is given in section 3.3.

2.2 Semantic definition of a fuzzy set defined on an ontology

In our system, a fuzzy set may be interpreted, either, with a semantics of preference when it is associated with a selection attribute in a query, or, with a semantics of an imprecise datum when it is stored in the database. *We suppose that attributes of the database may be defined on two kinds of domains: symbolic domains which are called ontologies in the following, and numerical domains. An ontology represents the* **a kind of** *semantic relationship between values (see definition 3).* In this section we define the semantics of a fuzzy set in intention and then its membership function on the entire ontology which we call fuzzy set in extension.

Definition 3 *An ontology* Ω *is defined as a set of values, partially ordered by* **a kind of** *relation and checking the properties of a lattice (each couple of values has a unique smallest generalization and a unique greatest specialization).*

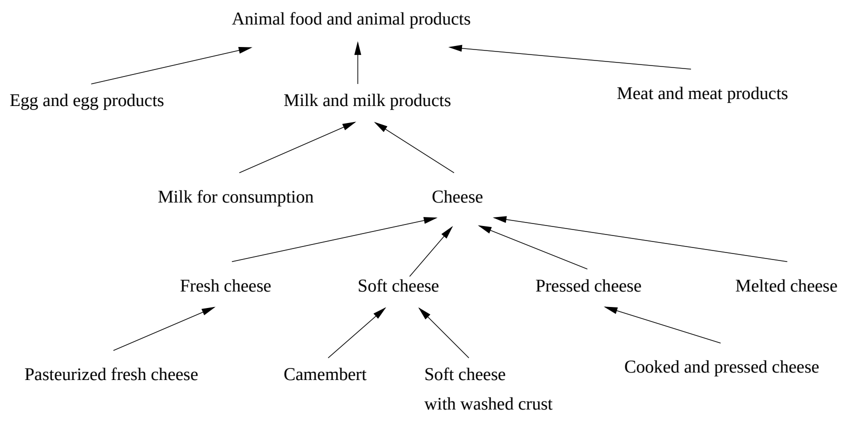


Figure 2: A part of the ontology defined for the attribute Substrate

Figure 2 presents a part of the ontology defined for the values taken by the attribute *Substrate* in our database.

Remark: "flat" symbolic domains, which are composed of unordered values are considered as a particular case of ontology where no value is comparable to another one (according to the **a kind of** relation).

Notation: For any fuzzy set f defined on Ω, and μ, the membership function of f, we note $support(f) = \{a \in \Omega \mid \mu_f(a) > 0\}$ and $kernel(f) = \{a \in \Omega \mid \mu_f(a) = 1\}$.

Definition 4 *A fuzzy set f_i is called fuzzy set in intention if it is defined on a domain of values Ω_i which is a subset of values belonging to the ontology Ω. For all couples of values a and b belonging to $support(f_i)$, with b more specific than a, the underlying semantics is defined as follows :*

given μ, the membership function of f_i : $\Omega_i \rightarrow [0, 1]$

- $\mu_{f_i}(a) \geq \mu_{f_i}(b)$ *represents a semantics of restriction for b compared to a,*

- *the opposite case represents a semantics of reinforcement for b compared to a.*

The fuzzy set *Substrate_Preferences* in figure 1 is an example of fuzzy set in intention for the attribute Substrate. It is also noted : *Substrate_Preferences* = 1.0/Fresh Cheese + 0.9/Cheese + 0.8/Soft Cheese. If the fuzzy set *Substrate_Preferences* is interpreted with a semantics of preferences, it means that the user is interested in cheese and implicitly in all subtypes of cheese found in the ontology. Moreover, the user is first interested in fresh cheese, and among the other cheeses, he/she is less interested in soft cheese. The same kind of semantic interpretation, concerning the comparison of membership degrees between values which are comparable[1], may be done if the fuzzy set stands for an imprecise value stored in the database.

The transformation of the fuzzy set in intention into a fuzzy set in extension is realised using the following definition.

Definition 5 *f_e, fuzzy set in extension associated with f_i, is defined on the entire set of values belonging to the ontology Ω. Given a value $\mathrm{c} \in \Omega$, the membership function of f_e is deduced from that of f_i by the following rules :*

- *we call $E = \{a_1, a_2, \ldots, a_n\}$ the set of the smallest values belonging to support(f_i) more general than c and not comparable[2]. If E is not empty and :*
  - *if the fuzzy set expresses preferences, then the degree of preference associated with c must be at least equal to the degree of preference associated with each element of the list $a_1, a_2, \ldots, a_n$, and we define $\mu_{f_e}(\mathrm{c}) = max_{a_1,a_2,\ldots,a_n}\mu_{f_i}(a_i)$*

[1]according to the **a kind of relation**

[2]using the partial order induced by the relation **a kind of**

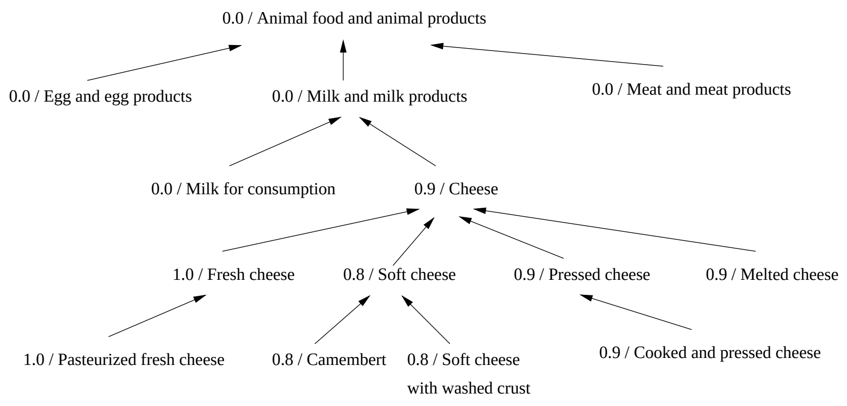

0.0 / Animal food and animal products
0.0 / Egg and egg products
0.0 / Milk and milk products
0.0 / Meat and meat products
0.0 / Milk for consumption
0.9 / Cheese
1.0 / Fresh cheese
0.8 / Soft cheese
0.9 / Pressed cheese
0.9 / Melted cheese
1.0 / Pasteurized fresh cheese
0.8 / Camembert
0.8 / Soft cheese with washed crust
0.9 / Cooked and pressed cheese

Figure 3: The membership function of the fuzzy set in extension corresponding to $Substrate\_Preferences$

  – *if the fuzzy set expresses an imprecise datum, then* c *cannot have a degree of possibility greater than each of those associated with* $a_1, a_2, \ldots, a_n$*, and we define* $\mu_{f_e}(\mathrm{c}) = min_{a_1,a_2,\ldots,a_n} \mu_{f_i}(a_i)$

- *otherwise* $\mu_{f_e}(\mathrm{c}) = 0$

According to this definition, the fuzzy set in extension corresponding to Substrate_Preferences (see figure 1) is shown on the ontology in figure 3.

Remark: The querying language defined in section 2.1 handles the new concept of fuzzy set on ontologies.

3 Fuzzy query processing

As already mentioned in the introduction, the central point of our work is to propose ways of helping any user to query incomplete, heterogeneously structured and imprecise information. Cooperative behaviours to assist the user in querying systems have already been widely studied in the field of databases and artificial intelligence (see [17, 1]). Different classes of cooperative answering have been proposed to be incorporated in a querying system (completive, suggestive, intentional, incomplete, conditional, etc). In the field of fuzzy querying, we propose two cooperative behaviours according to the order of system processing:

- firstly, the engine may use rules (see 3.2) which allow the user to complement the query with pertinent values;
- secondly, the engine may use ontologies (see section 3.3) to enlarge the user's preferences in order to retrieve additional answers which correspond to a generalization of his/her preferences.

We first explain in section 3.1 the standard fuzzy query processing without using those two cooperative behaviours which will be presented in the following two sections.

3.1 Processing

Starting from the MIEL query, the engine has to translate it into a standard SQL query to be able to execute it on a standard RDBMS. In the following we shortly present the 5 steps processed to do that:

- step 1: defuzzification of the fuzzy values associated with the selection attributes (determination of the list of values which belong to the support of the fuzzy set);
- step 2: prewritten SQL query completion with the values generated by the previous step;

- step 3: SQL query submission to a standard relational database management system (ORACLE in the present version);
- step 4: calculation of the relevance degree associated with each tuple retrieved by the relational database management system;
- step 5: tuples sorting according to the relevance degree.

In the calculation of the relevance degree step, firstly the engine has to compare selection attribute values and attribute values stored in the database which may be both represented as fuzzy values (see section 3.1.1); secondly the engine aggregates those elementary comparisons to provide the relevance degree associated with each tuple (see section 3.1.2).

3.1.1 Comparison between fuzzy values

The engine has to compare fuzzy values in two different steps of its processing: (step 1) for the comparison between selection attribute values and attribute values stored in the database to determine the closest tuples (imprecise values stored in the database are considered as possibility distributions); (step 2) for the comparison between selection attributes values of the query and head of rules if the user asked for a query completion (as we will see in section 3.2.3). We can distinguish, at least, two families of approaches to compare fuzzy values. In the first one [8, 16, 15], a measure is proposed to evaluate how close to each other two fuzzy representations are, taking into account similarity relations. In the second one, a measure is used to evaluate the possibility and necessity degrees of equality between a fuzzy value and a requirement (which may also be fuzzy) [11, 19]. [12] proposed an extension of fuzzy pattern matching which integrates similarity relations. In our engine, we have chosen the measure proposed in [12] belonging to the second family of approaches because in the comparison steps of our processing, we need to compare a fuzzy value (attribute values stored in the tuples of the database in step 1, selection attribute values of the query in step 2) to a requirement (selection attribute values of the query in step 1, head of rules in step 2).

Remark: Comparing fuzzy values defined on a symbolic domain, the engine must transform the fuzzy values defined on a part of the ontology (which are given in intention) into fuzzy sets defined in extension.

Definition 6 *Given a fuzzy set f which represents a requirement and a fuzzy set f' which represents a fuzzy value, we consider f_e and f'_e which are the corresponding fuzzy sets in extension. Given Π and N, respectively the degrees of possibility and necessity as defined in [12], we call RM, the ressemblance measure defined as follows :*

$$RM(f, f') = \frac{\Pi(f_e, f'_e) + N(f_e, f'_e)}{2}$$

3.1.2 Computation of the relevance degree

The engine aggregates elementary comparisons, presented in the previous section, to provide the relevance degree associated with each tuple, as follows.

Definition 7 *$a_i(t)$ is defined as the value associated with the attribute a_i corresponding to the tuple t stored in the database. This value may be precise or imprecise.*

As Q is a conjunctive query, the ressemblance measures obtained for each selection attribute $< a_i, v_i >$ compared to tuple t, noted $RM(a_i(t), v_i)$ and defined in section 3.1.1, are aggregated using a *min* operator.

Definition 8 $\{< a_i, v_i >, \forall i \in 1, \ldots, l\}$ *being the set of selection attributes of query* Q*, tuples* t *are ordered according to a relevance degree* rd *defined as follows:*

$$rd(t) = \min_{a_i \in 1,\ldots,l} (RM(a_i(t), v_i))$$

3.2 Rules

We present in this section the query completion mechanism using rules. We first define the semantics provided by a rule in section 3.2.1. Rule validation and rule completion are presented in the two following sections.

3.2.1 Rule definition

Rules, which are application dependent, have to be defined by an expert of the domain. In our microbiological application, they have been determined thanks to bibliographical sources independent from the database. Each rule is composed of (i) a head which determines which category of data is concerned by the rule, (ii) a body which permits to complement a query for a given selection criterium : the body contains a list of pertinent values to be scanned in the database for this selection criterium.

Here is one example of a rule used by our system in the field of microbiology : R_{Cheese} has for $Head$ = <Substrate, 1.0/Cheese>, and for $Body$ = <Germ, 1.0/Listeria monocytogenes + 1.0/Salmonella + 1.0/Escherichia coli + 0.5/Clostridium botulinum + 0.5/Staphylococus aureus + 0.2/Shigella + 0.2/Brucella spp>. Since an ontology of the domain is available for both attributes Substrate and Germ, it means that the fuzzy sets are defined here in intention. This rule has been extracted from a bibliographical source (see [20]). The order relation provided by the fuzzy set structure is used here to represent decreasing degrees of importance (frequency of appearance in the context of the application) and it means that: Listeria monocytogenes, Salmonella and Escherichia coli are frequent hazards in cheese; Clostridium botulinum and Staphylococus aureus have difficulties surviving but have been encountered; cases of contamination involving Shigella and Brucella spp have been encountered but are very rare.

A query, which satisfies the selection criteria of the head of a rule, may be complemented with the body of that rule. The complemented query is therefore more constrained than before. It provides three advantages : (i) there is a great chance that the user will retrieve fewer answers to analyze, (ii) answers are meaningful because the additional constraint provided by the rule is the expression of expert knowledge and (iii) the user can focus on the first answers because they are sorted according to decreasing degrees of importance as specified by the expert who wrote the rule.

Definition 9 *A rule, denoted R, is defined as a couple $<Head, Body>$ where:*

- *$Head$ is defined as a set of conjunctive atomic fuzzy predicates P_k defined as couples $<a_k, v_k>$. a_k is a selection attribute. v_k, defined as a fuzzy set, is the value associated with a_k.*
- *Body is the list of pertinent values for selection criteria. It is a set of couples $<a'_k, v'_k>$. a'_k is a selection attribute. v'_k is a fuzzy set representing pertinent values.*

In the knowledge base of the engine, rules are associated with the views available in the querying system. For example, the rule R_{Cheese} is associated with the two views $OneFactorExperience$ and $TwoFactorExperience$.

Definition 10 *Given a view V, LR_V is the list of rules $R_1, \ldots, R_n$ associated with V.*

Before being used for query completion (see section 3.2.3), those rules are first validated (see section 3.2.2).

3.2.2 Rule validation

As the knowledge included in rules is independent from that stored in the database, rule validation by the database is useful. When a rule is proposed by the system to the user to complement his/her query, it is important to inform him/her if data corresponding to this rule are available in the database.

This validation is, of course, not computed every time a query is processed but every time the rule base or the database is updated.

Definition 11 *A rule R is validated by the database through the view V if there exist n data ($n > 0$) which, with a degree strictly positive, simultaneously match the head and the body of R. This rule R is called n-validated through the view V.*

For example, the rule R_{Cheese} given previously is 1-validated by the following datum obtained through the view *OneFactorExperience*: <Substrate, 1.0/Pressed cheese>, <Germ, 1.0/Listeria monocytogenes>.

Remark 1: The n-validation calculus of rule R through a view V can be considered as a particular case of query we defined in section 2.1. $< a_1, v_1 >$

$, \ldots, < a_l, v_l >$ being the head of R and $< a'_1, v'_1 >, \ldots, < a'_m, v'_m >$ being the body of R, the query Q_R, processing the n-validation of rule R through a view V, can be written: $\{< V, a_1, \ldots, a_l, a'_1, \ldots, a'_m >, < a_1, v_1, false >, \ldots,$ $< a_l, v_l, false >, < a'_1, v'_1, false >, \ldots, < a'_m, v'_m, false >, NULL, 0\}$.

Remark 2: As the database is supposed to be incomplete by nature, rule validation by the database is an interesting tool for the database manager to determine the lack of information in it.

3.2.3 Query completion

When a user has begun to associate values with one (or more) selection attribute(s) of his/her query, it can be useful to propose pertinent values for the other selection attributes using rules presented in the previous section. As already mentionned in section 3.1.1, to determine if a rule R is applicable to complement a query Q, the head of R is considered as a requirement which must be checked by the selection criteria of Q. As the selection criteria of query Q are conjunctive and the head of rule R are all together composed of couples (attribute, value) also conjunctive, the relevance degree defined in section 3.1.2 may also be used to check this requirement.

Definition 12 *A rule R is applicable to complement a query Q if:*

- $R \in LR_{V_1} \cap \ldots \cap LR_{V_m}$ *($V_1, \ldots, V_m$ being the list of views specified in Q, LR_{V_i} being the list of rules associated with V_i),*
- *the relevance degree $rd(R, Q)$ between the head of R and the list of selection attributes of Q is strictly positive.*

The calculus of the list of applicable rules is available, as an optional function, in the user graphical interface of the querying system. When the user decides to apply a rule R, the system complements the query Q, under construction, with the body part of R.

For example, we consider the query Q, under construction, which is the following: $Q = \{$<*OneFactorExperience, Substrate, Germ,* pH_{min}*,* pH_{max}*, Factor, ResponseType* >, <*TwoFactorsExperience, Substrate, Germ,* pH_{min}*,* pH_{max}*, Factor1, Factor2, ResponseType*>, < *Substrate, Substrate_Preferences, false* >, *10, 0.2*$\}$. If the user asks for the query completion processing, the rule R_{Cheese} (given in section 3.2.1) will be considered applicable. If the user decides to apply it, the query Q will be complemented with the following selection attribute : <Germ, 1.0/Listeria monocytogenes + 1.0/Salmonella + 1.0/Escherichia coli + 0.5/Clostridium botulinum + 0.5/Staphylococus aureus + 0.2/Shigella + 0.2/Brucella spp>.

Rules allow to control the query completion using expert knowledge to focus the scanning on "well-known" combinations of values for selection criteria (Substrate and Germ for example). It is useful for users who are not experts of the application domain.

3.3 Query enlargement with ontologies

The query enlargement provides the user with a mechanism to evaluate automatically the mass of pertinent information available in the database which is close to his/her interest. In the MIEL query language, (see definition 1 of section 2.1), the user may specify, for each selection attribute, whether the engine is authorized to enlarge his/her preferences using similarity relations. The case of attributes defined on numerical domains has already been studied in [12]. In our system, this mechanism has been especially studied for attributes defined on an ontology. We present in section 3.3.1 the similarity relation we have defined on ontologies and in section 3.3.2 the preferences enlargement calculus realised thanks to this similarity relation. We compare our approach to that of [15] at the end of section 3.3.2.

3.3.1 Similarity relation

The query enlargement calculates a controlled generalization of the fuzzy set associated with a given selection criterium. By controlled generalization, we mean that the values added to the fuzzy set are pertinent values in the application context.

In this processing, the main difficulty is to be able to take into account the degree of generalization corresponding to this enlargement. For example, we want to enlarge the fuzzy set Substrate_Preferences shown in figure 1 using the ontology presented in figure 2. We would like to consider that the degree of generalization between the value *Cheese* and the value *Milk and Milk product* is not the same as that between the value *Milk and Milk product* and the value *Animal food and animal products*: the first one concerns milk products which have a similar behaviour with microorganisms, the second one groups together product families which have unsimilar behaviours with microorganisms (eggs, meat and milk).

In the following we propose a solution to take into account this degree of generalization in our system.

Notation: Given an ontology Ω, $(x, y) \in \Omega^2$, $x < y$ means that x is more specific than y.

Definition 13 *Given an ontology Ω, $(x, y) \in \Omega^2$ with $x < y$, y is called a father of x iff there is no $z \in \Omega$ such that $x < z < y$.*

The degree of generalization between values is defined by an expert and provided to the engine by means of the following function DG.

Definition 14 *Given an ontology Ω, DG is the partial function defined from Ω^2 to $\mathbb{R}^+$ for all couples of values $(x, y) \in \Omega^2$ such that y is a father of x.*

For example, *DG(Cheese, Milk and milk product)= 1* and *DG(Milk and milk product, Animal food and animal products)= 10. The greatest DG(x,y) is, the most general the link between x and y is.*

We drew upon the work of [13] to build a distance function between any couple of values belonging to the ontology Ω. To do that, we need first to introduce the notions of path and path distance between couples of values which are comparable.

Definition 15 *Given an ontology Ω, given two values $(x, y) \in \Omega^2$ with $x \leq y$, we call path from x to y, the set composed of n values $(v_1, \dots, v_n) \in \Omega^n$ such that v_1 is x, v_{i+1} is a father of v_i and v_n is y with $i \in [1, n]$).*

Definition 16 *Given an ontology Ω, given two values $(x, y) \in \Omega^2$ with $x \leq y$, given p a path from x to y, we call $PD_p(x, y)$, the distance between x and y associated with the path p. PD_p is defined as follows:*

$$PD_p(x, y) = \sum_{i=1}^{n-1} DG(v_i, v_{i+1})$$

Definition 17 *Given an ontology Ω, given two values $(x, y) \in \Omega^2$, we call $SD(x, y)$ the semantic distance between x and y defined as follows:*

- *if $x = y$ then $SD(x, y) = 0$,*

- *if x and y are comparable, (we suppose $x \leq y$), given the set of all the paths $p_1, \ldots, p_m$ from x to y, $SD(x,y) = \min_{p_i \in p_1,\ldots,p_m} PD_{p_i}(x,y)$,*

- *otherwise, given $z \in \Omega$, the more specific value which generalises x and y, $SD(x,y) = SD(x,z) + SD(y,z)$.*

We drew upon the work of [12] to build a similarity relation between any couple of values belonging to the ontology Ω.

Definition 18 *Given an ontology Ω and a semantic distance SD on Ω^2, we call SR, the similarity relation defined as follows :*

$SR : \Omega \times \Omega \rightarrow [0,1]$

$\forall (a,b) \in \Omega \times \Omega,\ SR(a,b) = max(0, min(1, \frac{\delta+\epsilon-SD(a,b)}{\epsilon}))$

$$= \begin{cases} 1\ if\ SD(a,b) \leq \delta \\ 0\ if\ SD(a,b) > \delta + \epsilon \\ otherwise\ \frac{\delta+\epsilon-SD(a,b)}{\epsilon} \end{cases}$$

For example, if we suppose that $\delta = 0$ and $\epsilon = 5$, *SR(Cheese, Milk and milk product) = 0.8*, *SR(Milk and milk product, Animal food and animal product) = 0* and *SR(Cheese, Animal food and animal product) = 0.* In the query enlargement process which uses the similarity relation, the example means that if the user has specified a *Milk and milk product* (or any more specific value), then the engine does not enlarge to *Animal food and animal product* which is considered as too general.

3.3.2 Enlargement calculus

We also drew upon the work of [12] to compute the enlargement of a fuzzy set using the similarity relation defined above.

Definition 19 *Given a similarity relation SR defined on Ω^2, given a fuzzy set in intention f_i defined on a subset of Ω and its corresponding fuzzy set in extension f_e defined on Ω, we call $gen(f_e)$, the enlarged fuzzy set in extension, defined as follows :*

$$\forall a \in \Omega, \mu_{gen(f_e)}(a) = \max_{b \in \Omega}(\min(\mu_{f_e}(b), SR(a,b)))$$

According to this definition, the enlarged fuzzy set in extension corresponding to *Substrate_Preferences* (showed in figure 1), taking into account the degrees of generalization shown in figure 4 and parameter values $\delta = 0$ and $\epsilon = 5$ for the similarity relation, is : 1.0/Fresh Cheese + 1.0/Pasteurized Fresh Cheese + 0.9/Cheese + 0.9/Pressed Cheese + 0.9/Melted Cheese + 0.9/Cooked and Pressed Cheese + 0.8/Soft Cheese + 0.8/Camembert + 0.8/Soft Cheese with washed crust + **0.8**/Milk and milk products + **0.6**/Milk for consumption + 0/Animal food or animal products + 0/eggs or egg products + 0/Meat or meat products.

We can make two remarks on this way of enlarging a fuzzy set :

- the fuzzy set is enlarged to the nearest more general values (*Milk and milk products* in the example). But it does not guarantee that all the more specific values associated with those nearest more general values are also included in the fuzzy set. It depends on the values chosen for the parameters δ and ϵ of the similarity relation. This first remark

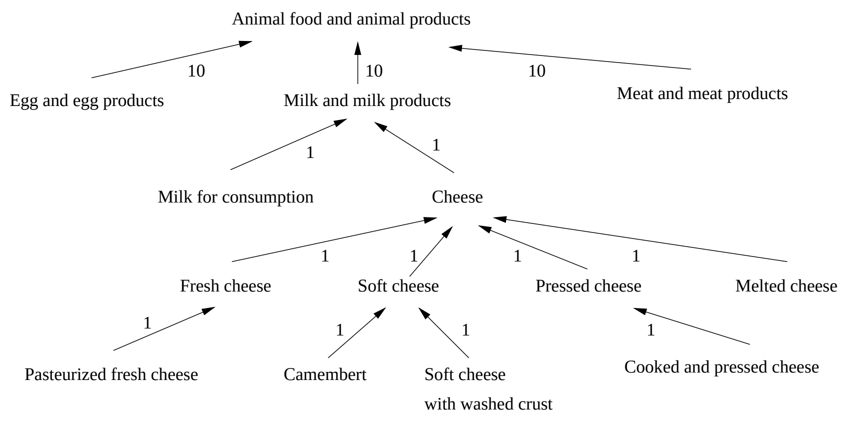


Figure 4: Enlargement of the fuzzy set *My_Substrate* according to definition 19. Values associated with the arrows correspond to the degree of generalization

.

reveals a slight contradiction with our way of defining a fuzzy set on the ontology: when a value belongs to a fuzzy set in intention, all the more specific values are intended to belong to the corresponding fuzzy set in extension.

- given two new values a and b added to the fuzzy set due to the enlargement process, with b more specific than a (*Milk and milk products* and *Milk for consumption* in the example), the membership degree of b is inferior to the one of a by construction. This second remark again reveals a slight contradiction with the semantics we gave to fuzzy sets defined on the ontology (definition 4). If we interpret the fuzzy set with a semantics of preference according to definition 4, the membership degree of b inferior to the one of a means that we are first interested in a (*Milk and milk products* in the example) and then in b (*Milk for consumption* in the example). We do not want to give this semantics to the enlargement process.

Those two remarks show that there are two major drawbacks to a direct application of this fuzzy set enlargement definition. We propose to adapt it in the following way. First, we have to determine the list of values belonging to the ontology which are more general than those of the support of the fuzzy set in extension f_e (called $l\_gen_{f_e}$). Second, we calculate, for each value of $l\_gen_{f_e}$, its membership degree to the enlarged fuzzy set $gen(f_e)$ using the formula of definition 19. Third, for each other value of the ontology which does not belong to $l\_gen_{f_e}$ nor to support(f_e), we determine its membership degree to the enlarged fuzzy set $gen(f_e)$ using similar rules as those given in the definition of a fuzzy set in extension (see definition 5).

Definition 20 *Given a similarity relation SR defined on Ω^2, given a fuzzy set in intention f_i defined on a subset of Ω and its corresponding fuzzy set in extension f_e defined on Ω, we call $l\_gen_{f_e}$, the list of more general values $\in \Omega$ defined as follows:* $\forall$ a $\in \Omega$*,* a $\in l\_gen_{f_e}$ *if* a $\notin$ *support*(f_e) *and* $\exists$ b $\in$ *support*(f_e) *with* a *more general than* b*.*

According to this definition, the list of more general values corresponding to the fuzzy set *Substrate_Preferences* in figure 1 is : {Milk and milk products, Animal food or animal products}.

Remark: In a fuzzy set in intention expressing preferences, values associated with a membership degree of zero must be excluded from the database scanning. For example, in the fuzzy set in intention $Substrate\_Preferences\_2$ = 1.0/Cheese + 0.0/Soft Cheese + 0.8/Camembert, Soft Cheese and more specific values except Camembert must be excluded from the scanning. In the enlargement process, selection criteria have to be relaxed and we choose to authorize also the scanning to those kinds of values. In the example of the fuzzy set in intention $Substrate\_Preferences\_2$, the value Soft Cheese belongs to $l\_gen_{f_e}$ and is candidate to belong to the support of the enlarged fuzzy set.

Definition 21 *Given a similarity relation SR defined on Ω^2, given a fuzzy set in intention f_i defined on a subset of Ω and its corresponding fuzzy set in extension f_e defined on Ω, we call $gen2(f_e)$, the enlarged fuzzy set in extension, calculated as follows :*

- *Step 1:* $\forall a \in \Omega$ *such that* $a \in support(f_e) \cup l\_gen_{f_e}$,

  $\mu_{gen2(f_e)}(a) = \max_{b\in\Omega}(\min(\mu_{f_e}(b), SR(a,b)))$

- *Step 2:* $\forall a \in \Omega \setminus (support(f_e) \cup l\_gen_{f_e})$,

  - *if the set* $(b_1, b_2, \ldots, b_n)$, *composed of the smallest values more general than* a *and not comparable*[3], *is not empty:*

    $\mu_{gen2(f_e)}(a) = max_{b_1,b_2,\ldots,b_n} \mu_{gen2(f_e)}(b_i)$

  - *otherwise* $\mu_{gen2(f_e)}(a) = 0$

According to this new definition, the enlarged fuzzy set in extension corresponding to $Substrate\_Preferences$ (shown in figure 1), taking into account the degrees of generalization (shown in figure 4) and parameter values $\delta = 0$ and $\epsilon = 5$ for the similarity relation, is: 1.0/Fresh Cheese + 1.0/Pasteurized Fresh Cheese + 0.9/Cheese + 0.9/Pressed Cheese + 0.9/Melted Cheese + 0.9/Cooked and Pressed Cheese + 0.8/Soft Cheese + 0.8/Camembert + 0.8/Soft Cheese with washed crust + **0.8**/Milk and milk products + **0.8**/Milk for con-

[3]using the partial order induced by the relation **a kind of**

sumption + 0/Animal food or animal products + 0/eggs or egg products + 0/Meat or meat products.

The use of similarity relations for attributes defined on a discrete domain has already been proposed in [15]. The objective was quite different: [15] meant to evaluate the degree of membership of an object, described by a set of attributes, to a set of hierarchised classes. In this approach, the relevance of an attribute A to a class C takes into account a degree of inclusion of A in the set of allowed values that a member of C may take for A. This degree of inclusion considers not only the equality but futhermore the similarity of values provided by similarity matrices defined for each domain of discrete values. In our work, firstly, we develop this approach in another context: we want to generalize a fuzzy set which represents preferences in a query and whose domain of values is hierarchised as seen in the definition of an ontology (see definition 3). Secondly, we improve the approach of [15] in the sense that we define how the similarity matrices are obtained from the ontologies.

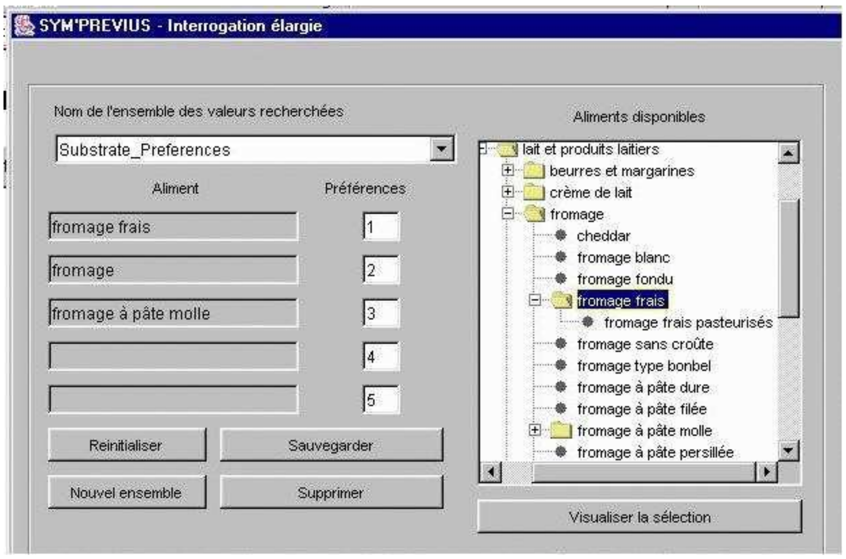
4 Implementation

Our laboratory is engaged in a national project which brings together government institutions and industry to build a tool for the analysis of microbiological risks in food products. This tool will include databases and a suitable information retrieval system available on the Internet. We have built a software, called MIEL [4], written in Java language which interacts with RDBMS databases (more precisely Microsoft Access and Oracle databases in the context of the application). A MIEL query is executed using a three-tier process architecture. This

[4]acronym for the French translation of Enlarged Querying Engine

architecture has been designed to minimise the data transfers between the user desktop and the MIEL server. Firstly, the MIEL Java client applet running under a usual Web browser implements the graphical user interface. Secondly, the MIEL Java server process running under a RMI (Remote Method Invocation) server implements all the calculus realized by the engine (query completion using rules, query enlargement, database scanning). Thirdly, the RDBMS server, belonging to the local area network of the MIEL Java server process, executes the SQL queries sent by the MIEL server. The MIEL software is completely generic: it is RDBMS-independent and application-independent. **RDBMS-independence:** Previous fuzzy query engines as FSQL and FQUERY are RDBMS dependent because they have been designed as internal extensions of the chosen RDBMS (respectively Oracle and Microsoft Access). On the contrary, the MIEL engine has been designed as a three-tier architecture. For this reason, it is easy to combine, in two different instances of this architecture, the MIEL java server with two different RDBMS. We have experimented it with Oracle and Microsoft Access. **Application-independence:** All the application-dependent information needed to manage queries is stored in the knowledge base of the MIEL server. It is composed of a set of relational tables stored in the RDBMS database. To use the MIEL server with another application, the manager of the MIEL system only needs to “feed” those tables with application-dependant information.

We present in the following an example of graphical user interface developed for the application in microbiology interacting with MIEL. Two steps of interaction with the GUI to execute a query are presented. Figure 5 shows the edition of the fuzzy set Substrate_Preferences shown in figure 1; words are in French because ontologies available for the current applica-


SYM'PREVIUS - Interrogation élargie

Nom de l'ensemble des valeurs recherchées: Substrate_Preferences

Aliment	Préférences
fromage frais	1
fromage	2
fromage à pâte molle	3
	4
	5

Aliments disponibles

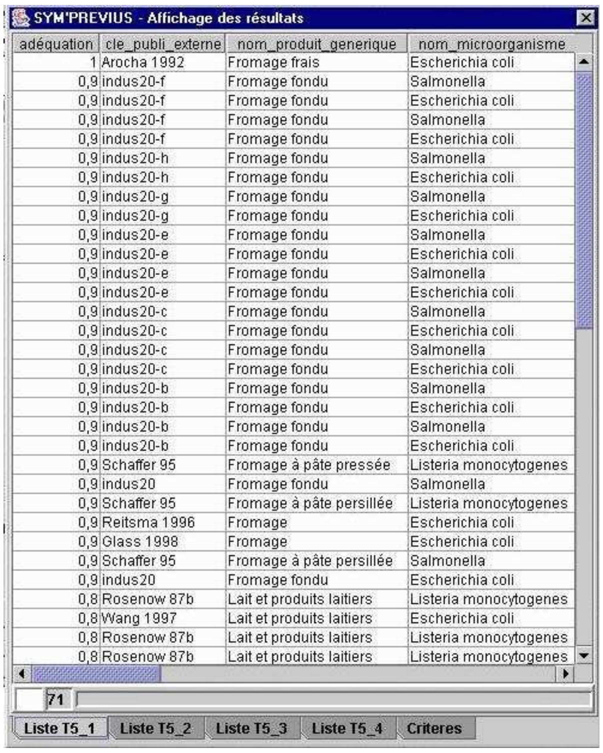
- lait et produits laitiers
  - beurres et margarines
  - crème de lait
  - fromage
    - cheddar
    - fromage blanc
    - fromage fondu
    - fromage frais
      - fromage frais pasteurisés
    - fromage sans croûte
    - fromage type bonbel
    - fromage à pâte dure
    - fromage à pâte filée
    - fromage à pâte molle
    - fromage à pâte persillée

Figure 5: The graphical component to edit a fuzzy set defined on an ontology

tion are in French. Figure 6 presents a sublist of projection attributes belonging to the answers retrieved by the MIEL server and sorted according to their degree of adequation to the selection criteria (Substrate_Preferences and list of microorganisms obtained by the query completion).

5 Conclusion

In this paper, we have introduced the concept of multi-view fuzzy query. Using this query, the user has only to specify his/her fuzzy preferences: the system facilitates the expression of a fuzzy query in a complex schema structure. First, the MIEL scans the database through several views simultaneously. Secondly, the engine uses a clear definition of fuzzy sets whose domain of values are ontologies. Thirdly, it proposes two mechanisms to help the user to adapt his/her preferences: (i) query completion using expert fuzzy rules, (ii) preferences



adéquation	cle_publi_externe	nom_produit_generique	nom_microorganisme
1	Arocha 1992	Fromage frais	Escherichia coli
0,9	indus20-f	Fromage fondu	Salmonella
0,9	indus20-f	Fromage fondu	Escherichia coli
0,9	indus20-f	Fromage fondu	Salmonella
0,9	indus20-f	Fromage fondu	Escherichia coli
0,9	indus20-h	Fromage fondu	Salmonella
0,9	indus20-h	Fromage fondu	Escherichia coli
0,9	indus20-g	Fromage fondu	Salmonella
0,9	indus20-g	Fromage fondu	Escherichia coli
0,9	indus20-e	Fromage fondu	Salmonella
0,9	indus20-e	Fromage fondu	Escherichia coli
0,9	indus20-e	Fromage fondu	Salmonella
0,9	indus20-e	Fromage fondu	Escherichia coli
0,9	indus20-c	Fromage fondu	Salmonella
0,9	indus20-c	Fromage fondu	Escherichia coli
0,9	indus20-c	Fromage fondu	Salmonella
0,9	indus20-c	Fromage fondu	Escherichia coli
0,9	indus20-b	Fromage fondu	Salmonella
0,9	indus20-b	Fromage fondu	Escherichia coli
0,9	indus20-b	Fromage fondu	Salmonella
0,9	indus20-b	Fromage fondu	Escherichia coli
0,9	Schaffer 95	Fromage à pâte pressée	Listeria monocytogenes
0,9	indus20	Fromage fondu	Salmonella
0,9	Schaffer 95	Fromage à pâte persillée	Listeria monocytogenes
0,9	Reitsma 1996	Fromage	Escherichia coli
0,9	Glass 1998	Fromage	Escherichia coli
0,9	Schaffer 95	Fromage à pâte persillée	Salmonella
0,9	indus20	Fromage fondu	Escherichia coli
0,8	Rosenow 87b	Lait et produits laitiers	Listeria monocytogenes
0,8	Wang 1997	Lait et produits laitiers	Escherichia coli
0,8	Rosenow 87b	Lait et produits laitiers	Listeria monocytogenes
0,8	Rosenow 87b	Lait et produits laitiers	Listeria monocytogenes

Figure 6: The graphical component to edit the result of the query

enlargement using ontologies. MIEL has been implemented in Java technology to be available on the Internet. MIEL can be coupled with any RDBMS of the market. It is currently used in a French national project, grouping together food industry partners, to query a database in the field of microbiology.

Concerning the storage and querying of heterogeneously data and due to the fact that it is expensive to modify the structure of a relational database (structure updating, data migration, GUI updating), it is preferable to release a new version of the relational schema with a significant periodicity (every 3 years for example). As a matter of fact, we are currently working on a unified fuzzy querying system that simultaneously scans two separate bases (see [9]): a relational database containing the more structured information and a conceptual *graph knowledge base containing the pieces of information that do not fit the structure of the relational database.*

Other immediate prospects of our work are presented in the following. Firstly, we want to use the MIEL engine to scan simultaneously our database and the Com'Base database (another database in microbiology). Vocabularies used in those two databases being different (product names for example), we have to build mappings between them to be able to scan both bases through the MIEL engine. Secondly, we are also working on another project, called E.dot, to design a datawarehouse which includes on the one hand our incomplete database and on the other hand, information found semi-automatically on the Web and stored in XML format (to complement information stored in our database). By means of those projects, our aim is to extend the MIEL engine in order to be able to scan simultaneously structured information stored in a relational database, less-structured information stored in a conceptual graph knowledge base and weakly-structured information retrieved from the Web.

References

[1] A. Bidault, C. Froidevaux, and B. Safar. Repairing queries in a mediator approach. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000)*, pages 406–410, August 2000.

[2] G. Bordogna and G. Pasi. A fuzzy object oriented data model managing vague and uncertain information. *International Journal of Intelligent Systems*, 14(6):SCI 3495, 1999.

[3] P. Bosc, L. Lietard, and O. Pivert. Soft querying, a new feature for database management system. In *Proceedings DEXA'94 (Database and EXpert system Application), Lecture Notes in Computer Science #856*, pages 631–640. Springer-Verlag, 1994.

[4] P. Bosc, L. Lietard, and O. Pivert. *Fuzziness in Database Management Systems*, chapter Fuzzy theory techniques and applications in data-base management systems, pages 666–671. Academic Press, 1999.

[5] P. Bosc and O. Pivert. *On the evaluation of simple fuzzy relational queries: principles and measures*, pages 355–364. Kluwer academic publishers, Boston, 1993.

[6] P. Bosc and O. Pivert. SQLf: a relational database language for fuzzy querying. *IEEE Transactions on fuzzy systems*, 3(1):1–17, February 1995.

[7] P. Bosc and O. Pivert. *Extending SQL retrieval features for the handling of flexible queries*, pages 233–251. John Wiley and Sons, Inc., 1997.

[8] P. Bosc and O. Pivert. On representation-based querying of databases containing ill-known values. In *Proceedings of ISMIS'97*, pages 477–486, 1997.

[9] P. Buche, O. Haemmerlé, and R. Thomopoulos. Integration of heterogeneous, imprecise and incomplete data: an application to the microbiological risk assessment. In *Proceedings of the 14th International Symposium on Methodologies for Intelligent Systems, ISMIS2003*, pages 98–107, Maebashi, Japan, October 2003. Lecture Notes in AI #2871, Springer.

[10] SM Chen and WT Jong. Fuzzy query translation for relational database system. *IEEE Transactions on Systems, Man and Cybernetics, part B*, 27(4):714–721, August 1997.

[11] J.C. Cubero and M.A. Vila. A new definition of functional dependency in fuzzy relational databases. *Journal of Intelligent Systems*, 9:441–448, February 1994.

[12] D. Dubois and H. Prade. *Fuzziness in Database Management Systems, P. Bosc and J. Kacprzyk eds.*, chapter Tolerant fuzzy pattern matching : an introduction, pages 42–58. Heidelberg: Physica Verlag, 1995.

[13] N. Foo, B.J. Garner, and A. Rao. Semantic distance in conceptual graphs. In *Proceedings of the 4th International Conference on Conceptual Structures*, pages 1–9. Springer-Verlag, 1989.

[14] J. Galindo, J.C. Cubero, O. Pons, and J.M. Medina. A server for fuzzy SQL queries. In *Proceedings of the 1998 workshop FQAS'98 (Flexible Query-Answering Systems)*, pages 161–171, Roskilde, Denmark, May 1998. Springer-Verlag.

[15] R. George, R. Srikanth, B. P. Buckles, and F.E. Petry. *An approach to modelling impreciseness and uncertainty in the object-oriented data model*, pages 325–337. John Wiley and Sons, Inc, 1997.

[16] R. George, A. Yazici, B. P. Buckles, and F.E. Petry. *Modeling Impreciseness and Uncertainty in the Object-Oriented Data Model - A Similarity-Based Approach*, pages 63–95. Advances in Fuzzy systems- Applications and Theory, Vol. 13, World scientific, 1997.

[17] J. Minker. An overview of cooperative answering in databases. In *Proceedings of the Third International Conference on Flexible Query Answering Systems (FQAS'1998)*, pages 282–285, Copenhagen, Denmark, 1998. Springer-Verlag.

[18] H. Prade. Lipski's approach to incomplete information data bases restated and generalized in the setting of Zadeh's possibility theory. *Information Systems*, 9(1):27–42, 1984.

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

[20] T.A. Roberts. *Microorganisms in Foods 6*. Blackie academic & professional, 1998.

[21] G. De Tré and al. *Recent research issues on the management of fuzziness in databases*, chapter A generalized object-oriented database model, pages 155–182. Bordogna, G. and Pasi, G. (eds), Studies in Fuzziness and Soft computing, Vol. 53, Physica-Verlag, Heidelberg, Germany, 2000.

[22] G. De Tré and R. De Caluwe. A generalized object-oriented database model with generalized constraints. In *Proceedings of the 18th NAFIPS Conference*, pages 381–386, New York, USA, 1999.

[23] L.A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

[24] S. Zadrozny and J. Kacprzyk. Implementing fuzzy querying via the internet/WWW: Java applets, activeX controls and cookies. In *Proceedings of the 1998 workshop FQAS'98 (Flexible Query-Answering Systems)*, pages 358–369, Roskilde, Denmark, May 1998. Springer-Verlag.