Dipartimento di Informatica
Università del Piemonte Orientale "A. Avogadro"
Via Bellini 25/G, 15100 Alessandria
http://www.di.unipmn.it

A fuzzy approach to similarity in Case-Based Reasoning suitable to SQL implementation

Authors: Luigi Portinale (luigi.portinale@unipmn.it)
Stefania Montani (stefania.montani@unipmn.it)

TECHNICAL REPORT TR-INF-2007-10-03-UNIPMN
(October 2007)

The University of Piemonte Orientale Department of Computer Science Research Technical Reports are available via WWW at URL
`http://www.di.mfn.unipmn.it/`.
Plain-text abstracts organized by year are available in the directory

Recent Titles from the TR-INF-UNIPMN Technical Report Series

2007-02 *Space-conscious compression*, Gagie, T., Manzini, G., June 2007.

2007-01 *Markov Decision Petri Net and Markov Decision Well-formed Net Formalisms*, Beccuti, M., Franceschinis, G., Haddad, S., February 2007.

2006-03 *New challenges in network reliability analysis*, Bobbio, A., Ferraris, C., Terruggia, R., November 2006.

2006-03 *The Engineering of a Compression Boosting Library: Theory vs Practice in BWT compression*, Ferragina, P., Giancarlo, R., Manzini, G., June 2006.

2006-02 *A Case-Based Architecture for Temporal Abstraction Configuration and Processing*, Portinale, L., Montani, S., Bottrighi, A., Leonardi, G., Juarez, J., May 2006.

2006-01 *The Draw-Net Modeling System: a framework for the design and the solution of single-formalism and multi-formalism models*, Gribaudo, M., Codetta-Raiteri, D., Franceschinis, G., January 2006.

2005-06 *Compressing and Searching XML Data Via Two Zips*, Ferragina, P., Luccio, F., Manzini, G., Muthukrishnan, S., December 2005.

2005-05 *Policy Based Anonymous Channel*, Egidi, L., Porcelli, G., November 2005.

2005-04 *An Audio-Video Summarization Scheme Based on Audio and Video Analysis*, Furini, M., Ghini, V., October 2005.

2005-03 *Achieving Self-Healing in Autonomic Software Systems: a case-based reasoning approach*, Anglano, C., Montani, S., October 2005.

2005-02 *DBNet, a tool to convert Dynamic Fault Trees to Dynamic Bayesian Networks*, Montani, S., Portinale, L., Bobbio, A., Varesio, M., Codetta-Raiteri, D., August 2005.

2005-01 *Bayesan Networks in Reliability*, Langseth, H., Portinale, L., April 2005.

2004-08 *Modelling a Secure Agent with Team Automata*, Egidi, L., Petrocchi, M., July 2004.

2004-07 *Making CORBA fault-tolerant*, Codetta Raiteri D., April 2004.

2004-06 *Orthogonal operators for user-defined symbolic periodicities*, Egidi, L., Terenziani, P., April 2004.

2004-05 *RHENE: A Case Retrieval System for Hemodialysis Cases with Dynamically Monitored Parameters*, Montani, S., Portinale, L., Bellazzi, R., Leonardi, G., March 2004.

2004-04 *Dynamic Bayesian Networks for Modeling Advanced Fault Tree Features in Dependability Analysis*, Montani, S., Portinale, L., Bobbio, A., March 2004.

A Fuzzy Approach to Similarity in Case-Based Reasoning suitable to SQL Implementation

Luigi Portinale, Stefania Montani

Dipartimento di Informatica
Universita' del Piemonte Orientale "A. Avogadro", Alessandria Italy

Abstract

The aim of this paper is to formally introduce a notion of acceptance and similarity, based on fuzzy logic, among case features in a case retrieval system. This is pursued by first reviewing the relationships between distance-based similarity (i.e. the standard approach in CBR) and fuzzy-based similarity, with particular attention to the formalization of a case retrieval process based on fuzzy query specification. In particular, we present an approach where local acceptance relative to a feature can be expressed through fuzzy distributions on its domain, abstracting the actual values to linguistic terms. Furthermore, global acceptance is completely grounded on fuzzy logic, by means of the usual combinations of local distributions through specific defined *norms*. We propose a retrieval architecture, based on the above notions and realized through a fuzzy extension of SQL, directly implemented on a standard relational DBMS. The advantage of this approach is that the whole power of an SQL engine can be fully exploited, with no need of implementing specific retrieval algorithms. The approach is illustrated by means of some examples from a recommender system called MyWine, aimed at recommending the suitable wine bottles to a customer providing her requirements in both crisp and fuzzy way.

Key words: fuzzy similarity, fuzzy matching, fuzzy case retrieval, fuzzy-SQL, CBR

1 Introduction

Case-Based Reasoning (CBR) systems rely on the well-known R4 cycle, consisting in the four "R" steps namely Retrieval, Reuse, Revise and Retain [1]. Despite the generality of the above process, case retrieval is the step that

Email address: portinal@di.unipmn.it, stefania@di.unipmn.it (Luigi Portinale, Stefania Montani).

has received most attention in the past, since there are a variety of applications and tasks, where just retrieving the most similar cases to the input one is an added value.

Case retrieval algorithms usually focus on implementing Nearest-Neighbor (NN) techniques, where local distance metrics relative to individual features are combined in a weighted way to get a global distance between a retrieved and a target case. A k-NN case retrieval algorithm will then return the k closest cases to the target one, with respect to the defined notion of distance; the underlying assumption is that they are the k most similar cases to the target one. In [5], it is argued that the notion of *acceptance* can be used as an alternative to distance, in order to represent the needs of a flexible case retrieval methodology. As for distance, local acceptance functions can be combined into global acceptance functions to determine whether a target case is acceptable (i.e. it is retrieved) with respect to a given query. In particular, very often local acceptance functions take the form of fuzzy distributions; in this way, the definition of a fuzzy linguistic term over the domain of a feature can be exploited to characterize the acceptance of cases having similar (in the fuzzy sense) values for that particular attribute or feature.

The definition of a fuzzy linguistic term over a case feature or attribute can be considered as a "similarity dimension" over which to compare the feature's values.
Given two values x and y of a feature f and a linguistic term $\mathcal{L}$ defined over f via a fuzzy membership function $\mu_f^{\mathcal{L}}$, we can characterize the similarity of x and y with respect to $\mathcal{L}$ through the absolute difference in the membership in $\mathcal{L}$ of x and y (i.e. $|\mu_f^{\mathcal{L}}(x) - \mu_f^{\mathcal{L}}(y)|$). Smaller is the difference, greater is the similarity of x and y with respect to $\mathcal{L}$. This captures the intuition that values that belong to the fuzzy set $\mathcal{L}$ with a close degree are considered semantically similar with respect to $\mathcal{L}$ (see also [17]).

A second aspect related to case retrieval concerns the fact that, in order to implement a particular case retrieval algorithm, suitable case structuring and case base organization have to be devised [19,27]. This may be a further burden in the construction of a CBR system, especially if data concerning cases are already available in standard relational databases, as in many applications. For this reason, the use of database technologies for supporting the construction of case-based systems is recently attracting serious attention; the reasons are twofold:

- if data of interest are already stored in a database, the database itself can be used as a case base;
- part of the technological facilities of a DBMS may be exploited in the CBR cycle, in particular for case retrieval.

In particular, with respect to the second point, it would be extremely convenient to be able to exploit standard query languages, like SQL, for the retrieval of the data of interest, i.e. the cases matching the query case. As mentioned above, we can denote a stored case as matching the query if the former is acceptable for the latter with respect to a predefined notion of acceptance. In the case of standard SQL (and so in the case of standard Relational Data Base Management Systems - RDBMS), the notion of acceptance can only be modeled through a boolean condition over the case feature values. This results in a great limitation for a retrieval system based on plain SQL, since more sophisticated notions of acceptance (like those adopted in CBR system) cannot be directly modeled.

A limited effort has been made, in order to define distance-based approach directly exploiting SQL [26]; on the contrary, in the last years, we assisted to a growing interest for the definition of fuzzy extensions to standard SQL [12,11,30,8,15,22,23,3], leading to several proposals for the definition of an SQL-like language able to deal with fuzzy conditions for the selection of data. The availability of such tools can provide a direct implementation of a case retrieval system based on a fuzzy notion of acceptance, over a standard RDBMS.

The aim of this paper is to formally introduce a notion of acceptance and contextual similarity, based on fuzzy logic, among case features in a case retrieval system, while proposing at the same time an actual retrieval architecture, based on the above notions and realized through a fuzzy extension of SQL directly implemented on a standard SQL engine.

The rest of the paper is organized as follows: section 2 reviews the relationships between distance-based and fuzzy-based similarity; section 3 formalizes the case retrieval process we are interested to model, semantically grounded on fuzzy logic; section 4 and section 5 deal with the implementation of the proposed framework and section 6 reports on some application examples from a recommender system. Finally, in section 7, a comparison with related works is presented and the conclusions are drawn.

2 Modeling Case Acceptability with Fuzzy Logic

The Nearest-Neighbor approach to case retrieval is based on distance measures defined on case features (local distance) that are then combined in a suitable way, in order to get the actual distance between a query or target case and a retrieved case (global distance). Case similarity is then defined as a complementary or dual function of case distance: greater is the distance, smaller is the similarity and vice versa. On the other hand, we can also view case retrieval as a way of accepting a set of cases, by means of some measure

of acceptance defined on the case features. Standard distance-based similarity, implicitly defines an acceptance-based retrieval, by means of the induction of acceptability regions around the target or query case.

In [5], a correspondence between acceptability and distance has been pointed out through the so-called *characteristics similarity curves* defining the acceptability regions in the feature space. Acceptability functions can then be used instead of distance or similarity functions and local acceptability can be combined into global acceptability functions to determine whether the case should be retrieved, that is whether it is acceptable with respect to the query case and a particular acceptability threshold.

In standard distance-based retrieval acceptability regions are induced from the local and global distance functions adopted. Another approach could be that of directly starting from measures directly modeling acceptability, Acceptability is however a concept related to the query at hand, so an acceptability measure must be defined according to a reference query context. A concrete example of an acceptability measure can be a fuzzy membership function, defining linguistic terms over a feature. A value x of a feature f has an acceptability of $\mu_f^{\mathcal{L}}(x)$, with respect to the reference $\mathcal{L}$ if $\mu_f^{\mathcal{L}}(x)$ is the membership degree of x in the fuzzy set $\mathcal{L}$. The standard notion of α-cut of a fuzzy set (i.e. the set of elements having degree of membership in the fuzzy set greater than or equal to α) can be used to define an acceptability region: all the elements in the α-cut are accepted. As in the case of characteristics similarity curves, different acceptability thresholds (i.e. different αs) give rise to different acceptability regions.

In fuzzy logic, the reference context for characterizing acceptability is usually a linguistic term, i.e. a fuzzy set; this differs form standard distance-based CBR, where such a reference is a feature value within the admissible range (or the "unknown" value is the feature is missing). As pointed out in [6], the relationship between a standard similarity function $SIM(u, v)$ and a fuzzy membership function can be established by considering the latter to be the membership of the fuzzy set SIM modeling the notion of similarity between two values on the "universe of discourse" (i.e. the range of the considered case attribute), such that $\mu_{SIM}(u, v) = SIM(u, v)$

On the other hand, if one starts by considering a particular fuzzy set $\mathcal{L}$ over the universe (in practical a linguistic term defined over the attribute's range), then the membership function of a value u in $\mathcal{L}$ (i,e. $\mu_{\mathcal{L}}(u)$) can be interpreted as the similarity of u with respect to any value a which is *focal* for $\mathcal{L}$. A value a is *focal* for a fuzzy set $\mathcal{L}$ if $\mu_{\mathcal{L}}(a) = 1$; this means that a is a fully representative value for the concept represented by $\mathcal{L}$, so the membership of u in $\mathcal{L}$ measures how close (i.e. similar) is u to the considered concept or linguistic term.

Finally, in a fuzzy context, global acceptability is easily obtained by using fuzzy combination through suitable t-norms o t-conorms [20]; this means that the standard framework of fuzzy logic provides a sound way of aggregating local information (at the feature level) into a global measure (at the case level).

Despite that, it is worth noting that the problem of aggregation of local fuzzy measures deserves particular attention. In fact, as precisely pointed out in [6], there may be different interpretations of the actual meaning of a local (at the feature level) fuzzy acceptability function, depending on how we interpret a similarity value equal to 0 (0-similarity). If this constraints the elements to be definitely not equivalent, then a t-norm based combination should be used, while on the contrary (i.e. if a value of similarity equal to 0 just does not contribute to aggregation) a t-conorm would be the right choice. We will return on this aspect in the next section.

The above considerations open the doors to a different starting point for case retrieval, semantically grounded on fuzzy logic. The following section will discuss how this can be formalized.

3 Fuzzy Case Retrieval

Since its first proposal, Case-Based Reasoning has been always identified as a way of retrieving (and then reusing possibly by adapting) a set of cases matching to a specified level of precision a specific query (often a target case). The problem of approximate matching is then intrinsic to CBR, but the preferred way of approaching this problem does not usually exploit fuzzy logic as the main tool for performing such a matching process. Actually, this possibility has been evidenced quite early in the CBR literature [17], but the mainstream of subsequent works largely focused on distance-based approaches.

In this section, we aim at formalizing the case retrieval process, based on a notion of approximate matching directly grounded on fuzzy logic. As pointed out in the previous section, this does not totally depart from distance-based approaches, since there are strong relationships between the two categories; however, it provides a different starting point able to exploit the sound framework of fuzzy set theory and, as we will show in the next section, a powerful computational framework based on SQL for implementing the retrieval process.

Let start by defining what is a case and which are the characteristics of case attributes.

Definition 3.1 *A* storable case *(or simply a* case*) c is a set of pairs* $\langle f, v \rangle$

where f is a feature and v an admissible value for f. We denote as $Range(f)$ the set of admissible values for the feature f.
A case base *is a set of storable cases $CB = \{c_i / c_i$ is a storable case$\}$.*

In CBR, a case is usually structured in two different parts: the problem description and the problem solution; without lack of generality, we will assume that the $\langle feature, value \rangle$ description is adopted for both parts.

Definition 3.2 *A feature f is said to be* nominal *if $Range(f)$ is a finite set of elements with no ordering relation among them.*
A feature f is said to be linear *if $Range(f)$ is an ordered set; in particular f is* discrete linear *if $Range(f)$ is isomorphic to $X \subseteq \mathbb{N}$ and f is* continuous linear *if $Range(f)$ is isomorphic to $Y \subseteq \mathbb{R}$.*

Feature categorization [28] is important in order to define the similarity measure associated with each single feature. In this paper, we will introduce feature similarities, by considering a set of linguistic terms as reference contexts for such similarities. These contexts formally take the form of fuzzy predicates. Depending on the category of the considered feature, three different types of fuzzy predicates are considered:

- *fuzzy predicates with continuous distribution*: corresponding to predicates defined through fuzzy sets with a continuous membership function; they can be defined on *linear* (both continuous and discrete) features (e.g. a trapezoidal distribution for the linguistic term `young` defined over the discrete domain of integers of the attribute `age`).
- *fuzzy predicates with discrete distribution*: corresponding to predicates defined through fuzzy sets with a point-based membership function; they can then be defined on both *nominal* as well as on *discrete linear* features. (e.g. a vector distribution for the linguistic term `bright` defined over the attribute `color`, associating to each color the membership function to the fuzzy set).

Moreover, approximate matching may require the use of suitable operators for comparing values in the stored cases with those provided in the query. For this reason, we consider the possibility of defining *fuzzy operators* to relate features values. Also in this case we have different types of operators:

- *continuous operators*: characterized by a continuous distribution function (e.g. the operator `near` over linear features, characterized by a trapezoidal, triangular or Gaussian-like curve centered at 0 and working on the difference of the operands);
- *discrete operators*: defining a *similarity relation* characterized by a symmetric matrix of similarity degrees (e.g. the operator `compatible` over the attribute `job` defining to what degree a pair of different jobs are compatible).

Given the above definitions, we can now characterize the Case Retrieval pro-

cess based on fuzzy logic. First of all, let us introduce the concept of query case.

Definition 3.3 *A* query case *is a set of pairs* $q = \{\langle f_1, v_1\rangle, \ldots \langle f_n, v_n\rangle\}$ *such that each* f_i *is a feature and each* v_i *is either*

(1) a fuzzy linguistic term defined over $Range(f_i)$ *with membership function* $\mu_{v_i} : Range(f_i) \rightarrow [0, 1]$ *or*
(2) an expression of the kind $op_i x_i$ *such that* $x_i \in Range(f_i)$ *and* op_i *is a binary operator (either crisp or fuzzy) defined over* f_i*.*

The first point refers to the use of a generic fuzzy value, including those generated by *fuzzy modifiers*; indeed, if a fuzzy modifier is used (e.g. *very*, *slightly*, etc...), the result is just a modified membership function that can then be used instead of the original one. We can consider as a fuzzy modifier any boolean expression of fuzzy terms defined on the feature at hand: for instance, if the fuzzy term `young` and `old` are defined over the feature `age`, we could be interested in the fuzzy expression

$$\langle \texttt{age}, (\texttt{young}\ OR\ (\texttt{old}\ AND\ NOT\ very\ \texttt{old}))\rangle$$

The fuzzy condition on feature `age` denotes a new fuzzy set $\mathcal{F}$ with relative membership computable from the original memberships for terms `young` and `old`; the whole expression can then be reduced to the more simple expression $\langle \texttt{age}, \mathcal{F}\rangle$.

A special (and common) case for the second point is the situation when the considered operator is the *equality operator* (=). In this case the expression $\langle f_i, = v_i\rangle$ simply denotes the standard condition of feature f_i assuming a crisp value v_i.

The use of a fuzzy linguistic term in the query can be useful also in situations when the end user specifies an actual (crisp) value for a feature, but it requires to abstract that value. In such a case, the user can specify the equality on a crisp value (as in point 2) asking the system to fuzzify it, on the basis of the fuzzy sets defined over the feature. After the fuzzification provided by the system, the situation is then reduced to the first point of definition 3.3.

For the sake of generality, let us indicate as $P(f_i, v_i)$ the predicate related to the assignment of a condition v_i to feature f_i; it can be a standard boolean predicate (in case v_i involves a crisp operator), or a fuzzy predicates if linguistic values are specified. For example $P(\texttt{age}, = 40)$ is true iff the feature `age` assumes the value 40; $P(\texttt{age}, \texttt{young})$ is true with degree $\mu_{\texttt{young}}(x)$ if $\mu_{\texttt{young}}$ is the membership function of the fuzzy set `young` defined over `age` and the feature `age` assumes values x.

It is worth noting that in our framework, each case in the case base (i.e. each storable case) does not have any fuzzy specification in its structure; fuzzy information is used only at the query level to retrieve stored cases on the basis of an approximate (fuzzy) match.

Definition 3.4 *Given a query case* $q = \{\langle f_i, v_i \rangle (1 \leq i \leq n)\}$*, the* retrieval condition *induced by* q *is the fuzzy predicate* $RC_q \equiv \mathcal{E}_{i=1}^{n} P(f_i, v_i)$ *where* $\mathcal{E}_{i=1}^{n}$ *is a boolean expression involving all the predicates* $P(f_i, v_i)$

The above definition allows one to determine which kind of role have to play the matching features in case retrieval; this is related to the discussion about the interpretation of a similarity degree equal to 0 at the end of section 2 (see also [6]). If for some features we ask to combine similarities in such a way that a 0-similarity fully contributes to non-equivalence, then a t-norm combination should be used, resulting in a predicate composition through the AND ($\wedge$) connective. If on the contrary, a 0-similarity does not contribute at all to non-equivalence, then a t-conorm should be the choice and the predicate composition should be performed through OR ($\vee$) connective. In case this two different kinds of concept should be combined, then the corresponding AND/OR formula can be adopted.

In the following, we will concentrate on the case when $RC_q \equiv \bigwedge_{i=1}^{n} P(f_i, v_i)$; this is the most common situation in CBR systems, where the features' requirements explicitly stated by the user ($P(f_i, v_i)$ in the above definition) are actually interpreted as (possibly fuzzy) constraints that must simultaneously hold.

Definition 3.5 *Given a storable case* c*, a query case* q *and the retrieval condition* RC_q*, the* matching degree *of* c *to* q *is* $\alpha(c, q) = \mu_{RC_q}(c)$

Definition 3.6 *Given a case base* CB*, a query case* q *and a threshold* $\lambda \in [0, 1]$*, the* retrieval set *of* q *with respect to* CB *and* λ *is the set* $RS(q, CB, \lambda) = \{c_i \in CB / \alpha(c_i, q) \geq \lambda\}$

The problem of finding the stored cases that best match the query, is then reduced to that of finding the set of cases satisfying, to an acceptable degree of match (represented by λ), the conditions specified in the query itself. What the present framework proposes is to fully exploit fuzzy logic for:

- modeling the similarity of the case features with respect to different approximate concepts (i.e. the fuzzy linguistic terms definable on the feature values);
- modeling the retrieval conditions as well as the acceptability of the retrieved cases.

Furthermore, an important aspect mentioned in the introduction, is the fact

that a fuzzy characterization of case matching can be directly captured in SQL based tools. In the next section we will show how a particular fuzzy extension of SQL can be exploited, in order to implement our framework.

4 Fuzzy Querying in Databases

It is well-known that standard relational databases can only deal with precise information and standard query languages, like SQL, only support boolean queries. Fuzzy logic provides a natural way of generalizing the strict satisfiability of boolean logic to a notion of satisfiability with different degrees; this is the reason why considerable efforts has been dedicated inside the database community toward the possibility of dealing with fuzzy information in a database.

Two main different approaches to this problem can be identified: (1) explicit representation of fuzzy information inside the database [4,23]; (2) fuzzy querying on a regular database [3,18].
By considering the case retrieval framework proposed in the previous section, it follows that the latter class of approaches is the one we are interested in. In fact, since storable cases are supposed to be stored as standard tuples, the interest is in defining a flexible querying approach on a standard database, to be used to implement case retrieval.

In [3] standard SQL is adopted as the starting point for a set of extensions able to improve query capabilities from boolean to fuzzy ones. The implementation of the SQL extensions can be actually provided on top of a standard relational DBMS, by means of a suitable module able to transform a fuzzy query into a regular one through the so called *derivation principle* [2].

In the following, we concentrate on this methodology, by defining a set of fuzzy extensions to SQL that may be of interest for CBR and by showing how they can be processed, in order to transform them in standard queries. In particular, for the aim of the present work, we are interested in simple SQL statements with no nesting (i.e. we consider the `WHERE` clause to be a reference to an actual condition and not to nested SQL statements); the condition in the `WHERE` clause, differently from standard SQL, can be a composite fuzzy formula involving both crisp and fuzzy predicates and operators as defined in section 3.

By allowing fuzzy predicates and operators to form the condition of the `WHERE` clause, the result of the `SELECT` is actually a fuzzy relation, i.e. a set of tuples with associated the degree to which they satisfy the `WHERE` clause. Such a degree can be characterized as follows: let

`SELECT` A `FROM` R `WHERE` fc

be a query with fuzzy condition fc; the result will be a fuzzy relation Rf with membership function

$$\mu_{Rf}(a) = \sup_{(x \in R) \wedge (x.A = a)} \mu_{fc}(x)$$

. The fuzzy distribution $\mu_{fc}(x)$ relative to fc must then be computed by taking into account the logical connectives involved and their fuzzy interpretation. It is well known that the general way to give a fuzzy interpretation to logical connectives is to associate negation with *complement to one*, conjunction with a suitable *t-norm* and disjunction with the corresponding *t-conorm* [20]. In the following, we will concentrate on the simplest t-norm and t-conorm, namely the min and max operators such that $\mu_{A \wedge B}(x) = \min(\mu_A(x), \mu_B(x))$ and $\mu_{A \vee B}(x) = \max(\mu_A(x), \mu_B(x))$.

4.1 Deriving Standard SQL from Fuzzy-SQL

In order to process a query using a standard DBMS, we have to devise a way of translating the Fuzzy-SQL statement into a standard one. The most simple way is to require the fuzzy query to return a boolean relation R_b which tuples are extracted from the fuzzy relation R_f, by considering a suitable threshold on the fuzzy distribution of R_f. We consider, as in [3], the following syntax

`SELECT (`λ`)` A `FROM` R `WHERE` fc

which meaning is that a set of tuples with attribute set A, from relation set R, satisfying the condition fc with degree $\mu \geq \lambda$ is returned (in fuzzy terms, the λ-cut of the fuzzy relation R_f resulting from the query is returned).

If we restrict attention to the kind of queries previously discussed (which are suitable to model standard case retrieval) and if we adopt min and max operator as norms, then it is possible to derive from a Fuzzy-SQL query, an SQL query returning exactly the λ-cut required; if other norms are used, we are only guaranteed that a superset of the λ-cut is returned and a further filter must be applied to the result [2].

This can be easily verified as follows (see also [2] for more details): let P be a fuzzy predicate; we write $P \geq \lambda$ to indicate that P is satisfied with degree greater or equal than λ. Let $DNC(P, \geq, \lambda)$ be the derived necessary condition for $P \geq \lambda$, i.e. a boolean condition such that $P \geq \lambda \rightarrow DNC(P, \geq, \lambda)$.
It is trivial to verify that

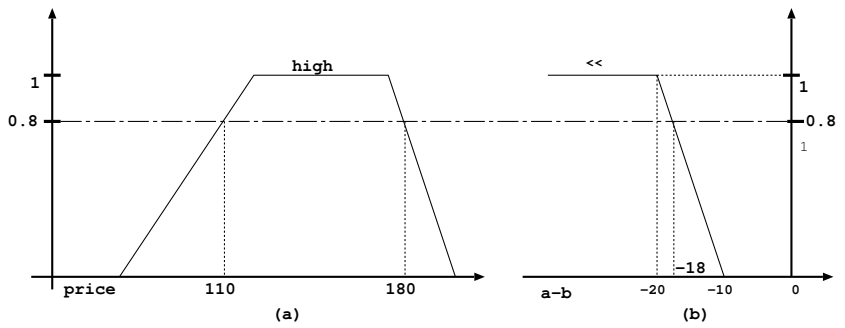

Fig. 1. Fuzzy Distributions

$AND(P_1, \ldots, P_n) \geq \lambda \leftrightarrow \min(P_1, \ldots, P_n) \geq \lambda$
$\leftrightarrow DNC(P_1, \geq, \lambda) \wedge \ldots, \wedge DNC(P_n, \geq, \lambda)$

$OR(P_1, \ldots, P_n) \geq \lambda \leftrightarrow \max(P_1, \ldots, P_n) \geq \lambda$
$\leftrightarrow DNC(P_1, \geq, \lambda) \vee \ldots, \vee DNC(P_n, \geq, \lambda)$

This implies that, using min and max operators, each derived necessary condition is also a sufficient one and thus the obtained boolean condition can be used to return exactly the required λ-cut [1] . Each DNC can then be obtained from the fuzzy distribution associated to the involved predicate.

Example. Consider a generic relation `PRODUCT` containing the attribute `price` over which the linguistic term `high` is defined. Figure 1 shows a possible fuzzy distribution for `high` as well as the distribution of a fuzzy operator $\ll$ (much less than), defined over the difference $(a - b)$ of the operands, by considering the expression $a \ll b$. Let C be a generic condition and $D(C)$ the fuzzy degree of C; $D(\texttt{price} = \texttt{high} \wedge \texttt{price} \ll 100) \geq 0.8$ will hold iff $\min(D(\texttt{price} = \texttt{high}), D(\texttt{price} \ll 100)) \geq 0.8$, iff $D(\texttt{price} = \texttt{high}) \geq 0.8 \wedge D(\texttt{price} \ll 100) \geq 0.8$ iff $(110 \leq \texttt{price} \leq 180) \wedge (\texttt{price} - 100) \leq -18$. The latter condition can be easily translated in a standard WHERE clause of SQL.

In the rest of this work we will assume that min and max operators are adopted as t-norm and t-conorm.

[1] Similar results hold for $(P, \leq, \lambda)$, to be used when negation is involved.

5 Implementing Fuzzy Case Retrieval

In order to make effective our fuzzy case retrieval approach on top of a relational database we have defined the following implementation framework.

- Cases are represented as tuples of relations. It could happen that the relevant information for a case is scattered in different relations of a relational scheme; in such a situation a suitable view is built, in order to reconstruct relevant case information in a single table-based structure. We will refer to this view as the case base CB.
- Case features are represented by standard relational attributes; basic SQL types will determine the category of the feature (e.g. `int` for a discrete linear feature, `float` for a continuous linear feature, `varchar` for nominal features, etc...).
- Fuzzy terms and fuzzy operators are defined through a suitable meta-database containing all the fuzzy knowledge.
- A query case is defined by specifying conditions on the values for a set of features as indicated in definition 3.3.
- Retrieval takes place, after specifying an acceptability threshold λ, by generating a Fuzzy-SQL query on the case base (with threshold λ), returning the set of cases (tuples) within the λ limit of acceptability (i.e. the λ-cut of the resulting relation).

As already mentioned in section 3, concerning the query case specification, a particular attention must be paid when an expression of the type $\langle f, = v\rangle$ is specified. This may be dealt with in two different ways, depending on the user intentions:

- using the crisp condition $f = v$, stating that the feature f must have value v;
- transforming the expression $\langle f, = v\rangle$ into the expression $\langle f, \mathcal{F}\rangle$ where $\mathcal{F}$ is the fuzzy set resulting from the fuzzification of the value v of the feature f.

In the last case, fuzzification may take place in different ways.

If the feature f is linear, the fuzzy expression f $\texttt{near}_{\texttt{f}}$ v can be generated, with $\texttt{near}_{\texttt{f}}$ being a fuzzy operator modeling the proximity of values for the feature f. This fuzzification strategy requires the knowledge engineer to specify such proximity operators for the features potentially subject to fuzzification.

In case f is a nominal attribute, a fuzzification method can be defined by considering the membership $\mu_1(v)$ with respect to a given fuzzy set F_1 defined on $Range(f)$. In that case, a new fuzzy set F_2 with membership distribution μ_2 can be defined such that $\mu_2(v') = 1$ for every v' such that $\mu_1(v') = \mu_1(v)$ and by scaling proportionally the membership function for other values.

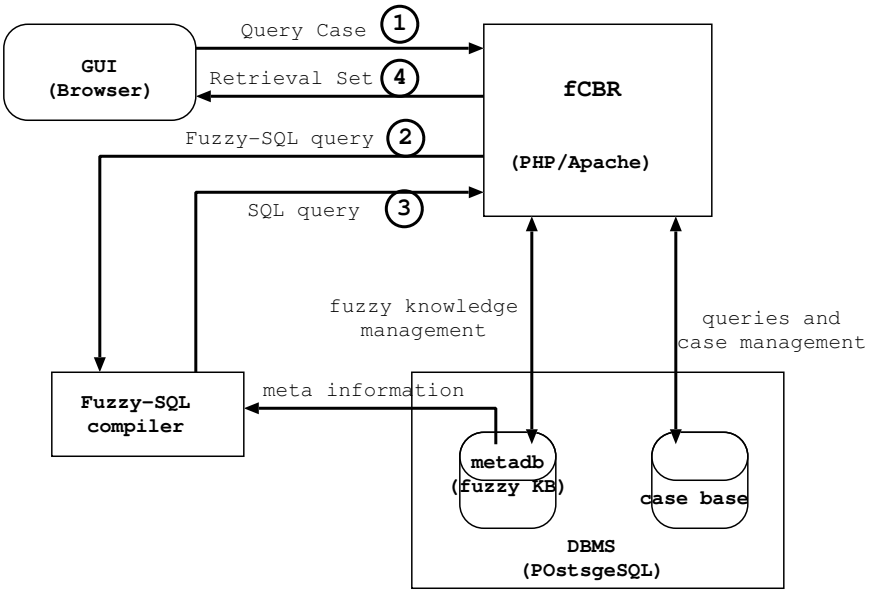

Fig. 2. The fCBR architecture

In [17] another simple kind of fuzzification is proposed (applicable to both linear and nominal features). This fuzzification takes also into account the level of acceptability λ specified for retrieval; it simply transforms the expression $\langle f, = v \rangle$ into the expression $\langle f, \bigvee_i \mathcal{F}_i \rangle$ for each $\mathcal{F}_i$ such that $\mu_{\mathcal{F}_i}(v) \geq \lambda$.

Once a query case q and an acceptability threshold λ have been specified on the case base CB, a Fuzzy-SQL query can then be generated as follows:
`SELECT` (λ) `*` `FROM` CB `WHERE` RC_q
Retrieved cases will then be obtained as the tuples of the result table obtained from the query. Of course, if one is interested in just a subset A of the case features, the target list of the query will be A instead of $*$.

Figure 2 shows the block scheme of the fCBR architecture, implementing the described framework. It is a web-based architecture implemented following the classical 3-tier approach. A standard browser provides the Graphical User Interface through which the user can interact (by providing query cases and by analyzing retrieved information) and the system administrator can manage the implemented application (by defining and controlling both the fuzzy knowledge base and the case base). A Fuzzy-SQL compiler is invoked every time a fuzzy query is generated from the user specified query case, returning the corresponding standard SQL code to be executed by the application.

To demonstrate the capabilities of the approach, we present an example of product recommendation: the MyWine application [2]. MyWine is a web-based application, exploiting the fCBR architecture described in the previous section, with the goal of recommending wine bottles, available from an e-catalog, matching a set of user requirements. Such requirements may contain precise (e.g. a wine from a particular producer) as well as vague or fuzzy specifications (e.g. a wine with a cheap price and a medium aging).

The relational scheme for the database is composed by the following tables:

```
WINE_REGION(region_name,country);
PRODUCER(prod_name,prod_address,prod_email,prod_website, region_name);
WINE_TYPE(wtype_name,type,color);
WINE(wine_name,prod_name,wtype_name,region_name,category,cru);
BOTTLE(wine_name,prod_name,year,availability,price)
```

Underlined fields represent primary keys. Foreign key constraints are defined as follows:

```
PRODUCER:foreign key(region_name)
                        references WINE_REGION(region_name)

WINE:foreign key(prod_name) references PRODUCER(prod_name)

WINE:foreign key(wtype_name) references WINE_TYPE(wtype_name)

WINE:foreign key(region_name) references WINE_REGION(region_name)

BOTTLE:foreign key(wine_name,prod_name)
                        references WINE(wine_name,prod_name)
```

The table `WINE_REGION` stores information about a particular wine region that may be the region of a producer as well as the region where a wine is produced (a producer may produce wines in different regions, while having residence in a particular wine region); table `WINE_TYPE` concerns information about a generic wine type (e.g. Barolo, Chianti Classico, Chardonnay California, etc...) such as the name, the possible type or classification (e.g. DOCG in Italy, AC in France) and the color; table `WINE` concerns specific wines of given producers belonging to a given wine_type (`wtype_name`) and stores information about the category of the wine (dry, sweet, strong sweet, passito) and the possible *cru* from which the wine is produced; table `BOTTLE` stores the specific information

[2] A preliminary version of this application has also been described in [24].

Field	Fuzzy Values	Distrib./Domain Type
producer	common, important	discrete/discrete
wtname	prestigious	discrete/discrete
regionname	important	discrete/discrete
aging	young, medium, old	contin./discrete
price	very_cheap, cheap, medium, expensive, very_expensive	contin./contin.

Table 1
Fuzzy predicates for MyWine.

about a particular production of a wine such as the year, the price and the store availability; finally table `PRODUCER` stores all the specific information of a producer.

As we can notice that database is quite structured since it has been designed by taking into account standard DB design techniques; however, in order to maintain relevant product information in a more compact form, the following view has been defined:

```
CREATE VIEW PRODUCT(winename,year,producer,price
                    wtname,wttype,color,category,
                    region,aging) AS
       SELECT B.wine_name,B.year,B.prod_name,B.price,
              WT.wtype_name,WT.type,WT.color,W.category,
              W.region_name,($CURRENT_YEAR-B.year) AS aging
       FROM BOTTLE B, WINE W, WINE_TYPE WT
       WHERE (W.wtype_name=WT.wtype_name) AND
                  (B.wine_name=W.wine_name)
```

The `PRODUCT` view represents the case base CB. Every row in such a view is a storable case. Notice that, once a suitable product has been retrieved by using the above view, if more details on the producer, the wine region or the wine itself are needed, they can be obtained by joining the view with the suitable tables.

In our example we consider the set of fuzzy predicates shown in table 1. The membership functions for the two continuous distributions (for `aging` and `price`) are reported in figure 3.

In addition to these predicates, two similarity relations (discrete operator) are also defined: sim_r on the attribute `regionname` modeling a fuzzy similarity among wine regions (with respect to "soil" characteristics) and sim_w on attribute `wtname`, modeling the similarity among wine types (with respect to

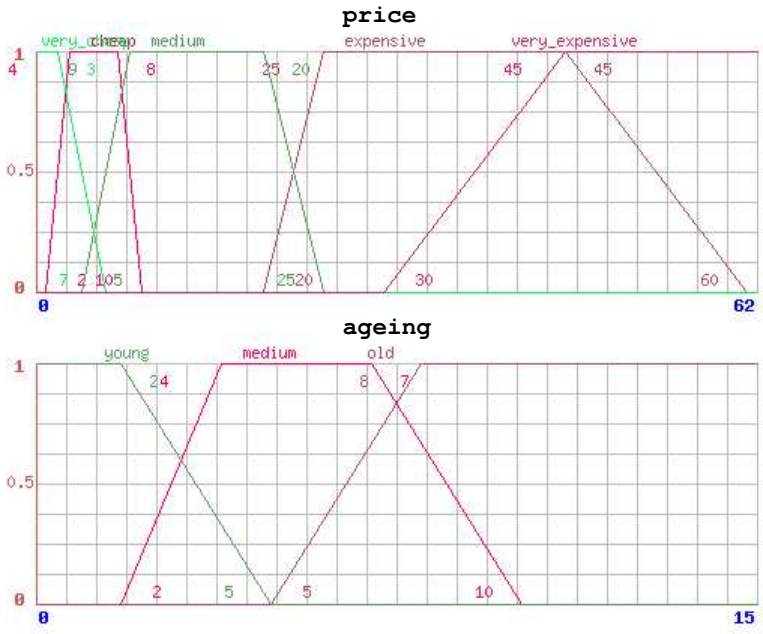

Fig. 3. Plotting of the fuzzy memberships for attributes `price` and `aging`

basic wine's characteristics). For instance, if the user requires a wine "similar to" *Barolo*, the system is able to exploit specific similarities in order to propose bottles of *Barbaresco* (and possibly of *Nebbiolo*) as well.

Finally, some continuous fuzzy operators have been defined; for example the operator *much less than* for the attribute `price` (*mlt_p*), the operator *near* for attributes `price` (*near_p*) and `year` (*near_y*), the operator *about* for attribute `aging` (*a_a*). In this way, we can model for instance a requirement of the type "the user wants a bottle of wine of about (*a_a*) 3 years of aging and with a price of about (*near_p*) 25.5 euros". Figure 4 shows the membership functions of such operators, each one defined on the difference between the operands.
A query case is constructed by the user through a suitable template, where the feature she is interested in (i.e. her requirements) are selected and instantiated (in a mixed crisp and fuzzy way). Notice that in a deployed application, the user does not necessarily has to specify each requirement directly. Requirements (and so fuzzy conditions) may be elicited through a guided user-machine dialogue; for example, the system may ask the user something like "Do you need the product for making an important gift?". If the answer is yes, the system may add to the retrieval condition an expression like $\langle$`wtname`, `important`$\rangle$.

Let us show an example of user consultation of MyWine. Suppose the user is interested in knowing which bottles of wines similar to "Barolo" are available in the catalog. She is interested in viewing in the result only the name of the wine, the producer, the year and the price. She also wants a relatively high degree of match to her requirements and she decided to specify an acceptability

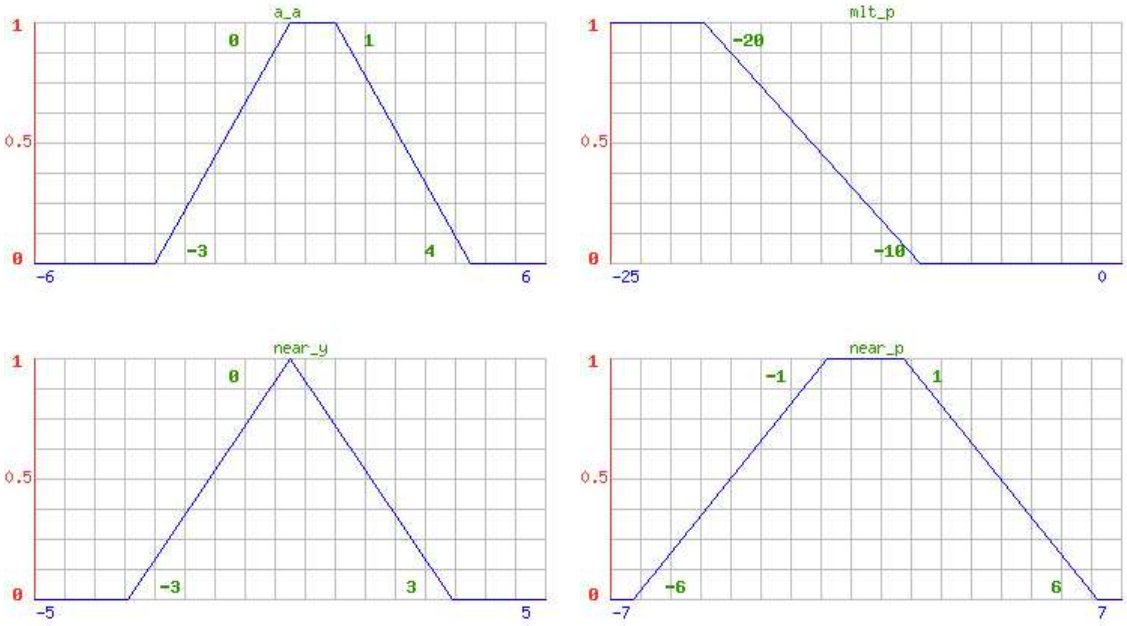

Fig. 4. Membership function for continuous operands in MyWine.

threshold of 0.7. The following query is generated by the system:

```
SELECT (0.7) winename, producer, year, price
FROM PRODUCT
WHERE wtname |sim_w| 'Barolo'
```

where $|op|$ is the syntax used to specify a discrete operator op. The translation into standard SQL produces the following query:

```
SELECT winename, producer, year, price
FROM PRODUCT
WHERE ( ( ( wtname LIKE 'Barbaresco' ) OR ( wtname LIKE 'Barolo' )
OR ( wtname LIKE 'Bolgheri_Rosso' ) OR ( wtname LIKE 'Bolgheri_Sassicaia' )
OR ( wtname LIKE 'Bordeaux_Rouge' ) OR ( wtname LIKE 'Cabernet_Sauvignon' )
OR ( wtname LIKE 'Chianti_Classico' ) OR ( wtname LIKE 'Langhe_Rosso' )
OR ( wtname LIKE 'Sicilia_Rosso' ) ) )
```

The wine types listed in the `WHERE` clause of the above query are those satisfying the relation sim_w with a degree greater than or equal to 0.7. The retrieved products are shown on table 2. As we can see, not every bottle is of Barolo wine, but several products are bottles of wine comparable with Barolo. Suppose now that the user is not satisfied since too many products are shown; a possibility is to impose a more strict acceptability level, by rising the threshold. If we set $\lambda = 0.8$, the last 3 tuples (from N. 15 to N. 17) are pruned, since their degree of matching is below the threshold.

N.	winename	year	producer	price
1	Barbaresco	1997	Angelo_Gaja	145
2	Barbaresco	1998	Angelo_Gaja	120.5
3	Chianti_Classico	1998	Banfi	18
4	Chianti_Classico	2000	Banfi	12
5	Barolo_Arione	1997	Gigi_Rosso	35
6	Barolo_Arione	1996	Gigi_Rosso	45.25
7	Langhe_Sperss	2006	Angelo_Gaja	120
8	Sito_Moresco	2005	Angelo_Gaja	45
9	Barolo_Castello	1993	Terre_del_Barolo	25.7
10	Chateau_Margaux	1990	Chateau_Margaux	234
11	Sassicaia	1996	Tenuta_San_Guido	225
12	Barolo_PerCristina	2000	Domenico_Clerico	77
13	Barolo	2003	Domenico_Clerico	51.5
14	Barolo	2002	Domenico_Clerico	55
15	La_Segreta_Nero	2000	Planeta	30
16	Alfeo	1998	Ceralti	35
17	Cabernet_Sauvignon_Reserve	1993	Mondavi	66

Table 2

Consider now the user willing to add a requirement about the price, and in particular that among the selected products, the interest is on a medium price. If we denote as $RT1$ the result table containing tuples from N.1 to N. 14 (i.e. the second retrieval set), the following query is generated

```
SELECT (0.8)  winename, producer, year, price
FROM RT1
WHERE price=[medium]
```

Again the syntax $f = [l]$ means that we ask for a feature f taking value in the fuzzy set l with continuous distribution. The derived SQL query will be

```
SELECT  winename, producer, year, price
FROM RT1
WHERE price BETWEEN 9 AND 21
```

The new retrieval set $RT2$ just contains tuples (cases) N.3 e N.4, since the

other bottles are too expensive with respect to the query. The user can decide now to rise the price level from medium to expensive, but by stating a limit as well, that is that the price must be much less than 70 euros. The new query that the system generates is:

```
SELECT (0.8)  winename, producer, year, price
FROM RT1
WHERE price=[expensive] AND price mlt_p 70
```

that is equivalent to:

```
SELECT (0.8)  winename, producer, year, price
FROM PRODUCT
WHERE price=[expensive] AND price mlt_p 70 AND wtname |sim_w| 'Barolo'
```

The derived boolean condition on `price` will be

```
(price BETWEEN 24 AND 48) AND (price-70 <= -18.0)
```

At this point only tuples (cases) N. 5,6,8,9 are retrieved and the user can decide to stop, since a reasonable set of products among which to decide has been recommended.

What is important to notice is that, in this approach, the acceptability level (i.e. the λ threshold) is global on the whole retrieval condition; this means that, in order to distinguish different acceptability levels on the different features, a well defined system-user interaction must devised.

As another example, consider a user looking, with a rather high degree of matching, for old-aged wines similar to "Barbaresco" and from an important wine region. The generated query will be:

```
SELECT (0.8) winename, producer, year, price
FROM PRODUCT
WHERE (wtname |sim_w| 'Barbaresco') AND (regionname = {important})
AND (aging = [old])
```

Syntax $f = \{l\}$ concerns the match of a feature f with a fuzzy value l with discrete distribution. This result in the following SQL code:

```
SELECT winename, producer, year, price
FROM PRODUCT
WHERE ( ( ( wtname LIKE 'Barbaresco' ) OR ( wtname LIKE 'Barolo' )
OR ( wtname LIKE 'Bolgheri_Rosso' ) ) )
AND ( ( (regionname = 'Bordeaux' ) OR ( regionname = 'Chianti' )
OR ( regionname = 'Langhe' ) OR ( regionname = 'Napa_Valley' )
```

N.	winename	year	producer	price
1	Barbaresco	1997	Angelo_Gaja	145
2	Barbaresco	1998	Angelo_Gaja	120.5
5	Barolo_Arione	1997	Gigi_Rosso	35
6	Barolo_Arione	1996	Gigi_Rosso	45.25
9	Barolo_Castello	1993	Terre_del_Barolo	25.7
16	Alfeo	1998	Ceralti	35
12	Barolo_PerCristina	2000	Domenico_Clerico	77

Table 3

```
OR ( regionname = 'Bolgheri' ) OR ( regionname = 'Sauternes' ) ) )
AND ( ( aging >= 7 ) )
```

The retrieval set $RT3$ is shown in table 3. Now the user decides to filter the result with another requirement: the wine she's looking for must be of an important producer. The new query takes the form of

```
SELECT (0.8) winename, producer, year, price
FROM RT3
WHERE producer={important}
```

or equivalently

```
SELECT (0.8) winename, producer, year, price
FROM PRODUCT
WHERE (wtname |sim_w| 'Barbaresco') AND (regionname = {important})
AND (aging = [old]) AND (producer={important}
```

This retrieves only tuples N. 1,2,12. However, the user could decide to add the requirement on the producer, but with a smaller acceptability degree with respect to the rest, for instance 0.7. This has to be implemented by the following query

```
SELECT (0.7) winename, producer, year, price
FROM RT3
WHERE producer={important}
```

that is not equivalent to

```
SELECT (0.7) winename, producer, year, price
FROM PRODUCT
WHERE (wtname |sim_w| 'Barbaresco') AND (regionname = {important})
```

N.	winename	year	producer	price
1	Barbaresco	1997	Angelo_Gaja	145
2	Barbaresco	1998	Angelo_Gaja	120.5
3	Chianti_Classico	1998	Banfi	18
4	Chianti_Classico	2000	Banfi	12
5	Barolo_Arione	1997	Gigi_Rosso	35
6	Barolo_Arione	1996	Gigi_Rosso	45.25
16	Alfeo	1998	Ceralti	35
10	Chateau_Margaux	1990	Chateau_Margaux	234
11	Sassicaia	1996	Tenuta_San_Guido	225
12	Barolo_PerCristina	2000	Domenico_Clerico	77

Table 4

```
AND (aging = [old]) AND (producer={important}
```

Indeed, with the first query we only retrieve tuples N. 1,2,5,6,12, while with the second, we would relax also the requirements on the wine region, the wine type and the aging, by producing a totally different retrieval set (see table 4). Figure 5 shows some screenshots of the described session.

7 Conclusions and Related Works

We have presented an approach where local acceptance relative to a feature can be expressed through fuzzy distributions on its domain, abstracting the actual values to linguistic terms. Global acceptance is then completely defined in fuzzy terms, by means of the usual combinations of local distributions through specific defined *norms* and so it is completely grounded on fuzzy logic.

As noted by many researchers [17,21], the use of fuzzy set theory for indexing and retrieval of cases has several advantages: the conversion of numerical features into qualitative ones (with the advantage of making simple the retrieval as well as of providing a suitable level of abstraction for judging similarity); the use of multiple retrieval keys or indices for a given feature (i.e. different fuzzy sets defined on the same feature); the use of modifiers for making retrieval more flexible, etc... In [9] it is argued that fuzzy logic is really relevant in both case representation and in case retrieval, where "fuzzy knowledge representation" and "fuzzy matching" methods can provide the suitable tools.

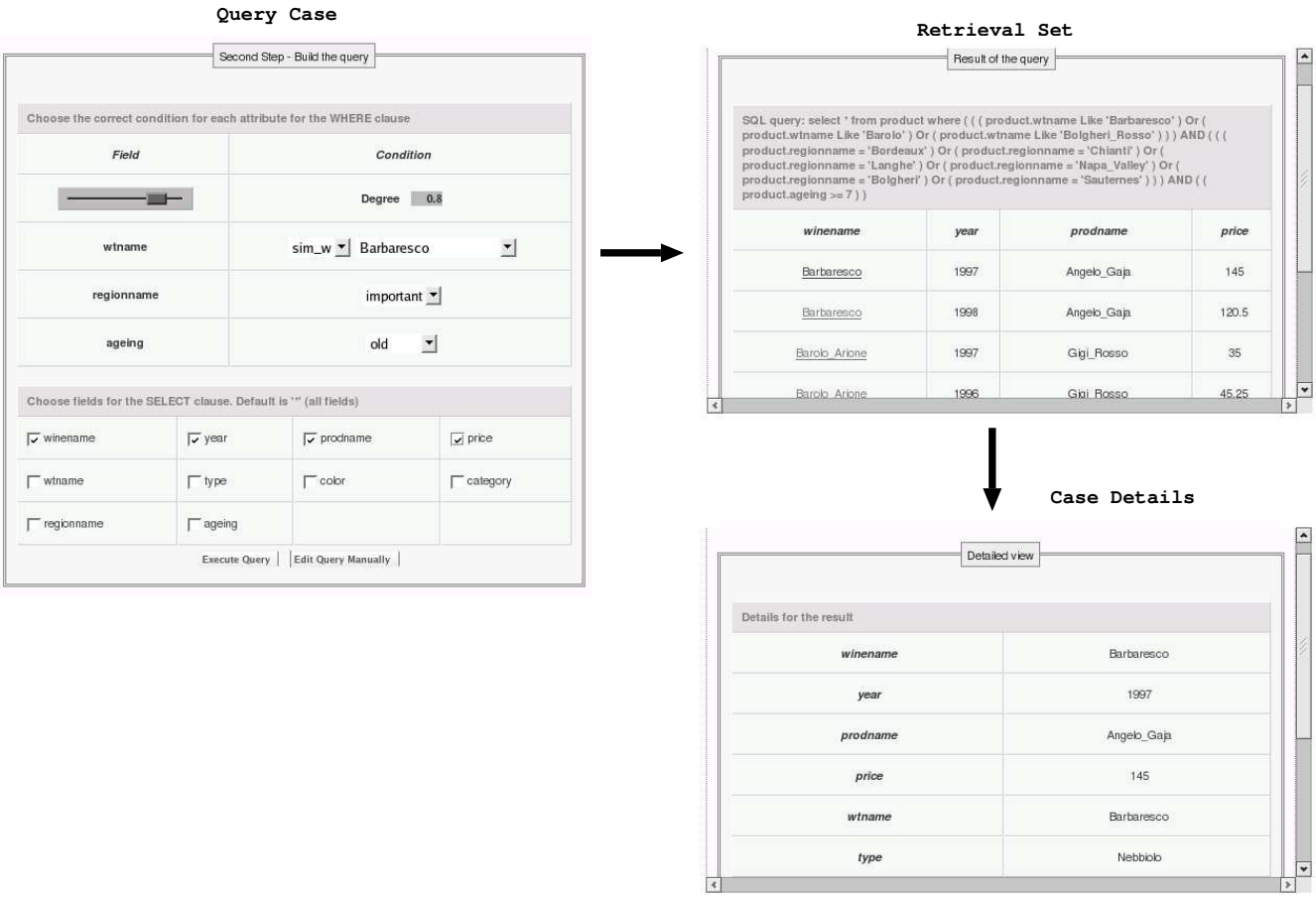

Fig. 5. Screenshots from `MyWine`

Also Yager in [29] argues for a unified view of CBR and fuzzy reasoning systems and works in [6,10] are clearly steps in this direction. Fuzzy rules have also been proposed as a way of formalizing the case-based reasoning process [14], by making clear the distinctions among the different interpretations of the so-called CBR hypothesis ("similar problems have similar solutions").

Despite that, the attempt of building CBR systems directly exploiting fuzzy logic for retrieval has not received as much attention as the definition of distance-based approaches, which are by now much more popular in the CBR community. Remarkable exceptions are the works in [16,17,7,13].

In [16] the system CAREFUL is described; it is based on a hierarchical object-oriented case representation, where case features can assume fuzzy linguistic values as well. A fuzzy classification algorithm is then exploited to implement case retrieval.

In [17] an attempt is described to directly use fuzzy set theory for indexing and retrieving cases; however, the emphasis is more on making possible a simple indexing of numerical features through abstractions to fuzzy linguistic terms.

An index structure is then built for every case feature, using fuzzy terms as index keys. The matching and retrieval process neither addresses the problem of properly combining query requirements as done in our work, nor it addresses the issue of standard DBMS retrieval.

The work in [7] reports on some experiences at General Electric in using fuzzy logic (and soft computing techniques in general) for CBR. In particular, in the *property valuation* application fuzzy sets are used to model a preference criterion on the relevant features, by setting a membership degree equal to 1 for values that are considered definitely similar and by scaling other values (until 0 is reached) for values at the boundary of the allowed deviations. In the *plastics color matching* application, a fuzzy preference function is used to compute the similarity of a single feature with the corresponding target feature and linguistic terms are associated to the level of similarity (excellent for very small differences, good for small differences, etc...) In these works, fuzzy logic is essentially exploited to model similarities, but no attempt is done to build a global fuzzy retrieval condition and in implementing it (as done by our Fuzzy-SQL approach). In particular, similarity is directly expressed through fuzzy sets (defined for instance on the difference of values), while in our approach, similarity is induced as a difference in membership and is contextualized by the possible linguistic terms that are defined on the feature's range.

Finally, the approach described in [13], deals with the definition of a fuzzy CBR strategy for whether prediction. The problem described there requires to retrieve the set of k most similar cases to the target, in order to obtain predictions on the whether conditions, by means of a weighted aggregation of the retrieved outcomes. This is a quite standard task in CBR, but the novelty is that a fuzzy k-NN classification is adopted. Fuzzy logic is then used as a way of comparing case features, in order to get a degree of membership in the resulting classes on which to base the final prediction.

In the present paper we have defined a case retrieval approach suitable for query cases expressible (either directly or indirectly) with fuzzy concepts. An extended version of SQL, able to deal with fuzzy predicates and conditions, is introduced as a suitable way to directly query a case base stored on a relational DBMS; this approach is based on the language proposed in [3], extending standard SQL in order to deal with *fuzzy queries*. The advantage of this approach is that the whole power of an SQL engine can be fully exploited, with no need of implementing specific retrieval algorithms. Moreover, the use of SQL and of standard DBMS allows us to obtain an efficient retrieval even in very large case bases.

Once the retrieved cases has been obtained, they can be used as starting point for different kind of CBR systems. For example, in a CBR-based recommender system (as described in section 6), the case base CB can be a table or view,

containing the set of products to be recommended; the user can then specify a set of characteristics the product she looks for should have, without the commitment of being precise about such characteristics. The fuzzy retrieval engine will finally propose the set of products in the case base that match the user requirements at the desired level (possibly after a suitable set of interactions).

In a medical decision support system, one case feature can be the diagnostic class of a patient (i.e. the case solution). The fuzzy specification of a set of features of a prototypical patient belonging to one class can support the diagnostician in determining the standard characteristics of a diagnostic class, as well as the possibility of actually performing a diagnosis [25].

We believe that the research on the strict relationships between CBR and fuzzy set theory will eventually lead to the constructions of flexible reasoning systems, able to deal with problems of greater and greater complexity.

References

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations and system approaches. *AI Communications*, 7(1):39–59, 1994.

[2] P. Bosc and O. Pivert. Fuzzy queries in conventional databases. In L. Zadeh and J. Kacprzyk, editors, *Fuzzy Logic for the Management of Uncertainty*, pages 645–672. John Wiley, 1992.

[3] P. Bosc and O. Pivert. SQLf: a relational database language for fuzzy querying. *IEEE Transactions on Fuzzy Systems*, 3(1), 1995.

[4] B.P. Buckles and F.E. Petry. A fuzzy representation of data for relational databases. *Fuzzy Sets and Systems*, 7:213–226, 1982.

[5] H-D. Burkhard. Extending some concepts of CBR: foundations of case retrieval nets. In M. Lenz, B. Bartsch-Spoerl, H-D. Burkhard, and S. Wess, editors, *Case Based reasoning Technology: from Foundations to Applications*, pages 17–50. LNAI 1400, Springer, 1998.

[6] H-D. Burkhard and M.M. Richter. On the notion of similarity in Case-Based Reasoning and fuzzy theory. In S.K. Pal, T.S. Dillon, and D.S. Yeung, editors, *Soft Computing in Case Based Reasoning*, pages 29–46. Springer, 1998.

[7] W. Cheetham, P. Cuddihy, and K. Goebel. Applications of soft CBR at General Electric. In *Soft Computing in Case-Based Reasoning*, pages 335–365. Springer, 2000.

[8] E. Cox. Fuzzy sql: A tool for finding the truth. the power of approximate database queries. *PC AI Magazine*, 14(1), 2000.

[9] R. Lopez de Mantaras and E. Plaza. Case-based reasoning: An overview. *AI Communications*, 10:21–29, 1997.

[10] D. Dubois, F. Esteva, P. Garcia, L. Godo, R. Lopez de Mantaras, and H. Prade. Fuzzy modelling of case-based reasoning and decision. In *Proc. 2nd Intern. Conference on Case-Based Reasoning (ICCBR-97), LNAI*, pages 599–610, Providence, RI, 1997.

[11] J. Galindo, J.M. Medina, O. Pons, and J.C. Cubero. A server for fuzzy SQL queries. In *Lecture Notes in Artificial Intelligence 1495*, pages 164–174. Springer, 1998.

[12] J. Galindo, A. Urrutia, and M. Piattini. *Fuzzy Databases: modeling, design and implementation*. IGI Publ., 2006.

[13] B.K. Hansen. *Whether prediction using CBR and fuzzy set theory*. Master Thesis, Dalhousie University, 2000. `http://www.cs.dal.ca/~bjarne/thesis.pdf`.

[14] H. Huellermeier, D. Dubois, and H. Prade. Formalizing case-based inference using fuzzy rules. In *Soft Computing in Case-Based Reasoning*, pages 47–72. Springer, 2000.

[15] Scianta Intelligence. Fuzzy sql (http://scianta.com/products/fuzzysql.htm).

[16] M. Jaczynski and B. Trousse. Fuzzy logic for the retrieval step of a case-based reasoner. In *2nd European Workshop on Case-Based Reasoning - EWCBR94*, pages 313–320, Chantilly, France, 1994.

[17] B.C. Jeng and T-P. Liang. Fuzzy indexing and retrieval in case-based systems. *Expert Systems with Applications*, 8(1):135–142, 1995.

[18] J. Kacprzyck and A. Ziolowski. Database queries with fuzzy linguistic quantifiers. *IEEE Transactions on Systems, Man and Cybernetics*, 16(3), 1986.

[19] J.L. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, 1993.

[20] C.T. Liu and C.S. George Lee. *Neural Fuzzy Systems*. Prentice Hall, 1996.

[21] J. Main, T.S. Dillon, and S.C.K. Shiu. A tutorial on case-based reasoning. In *Soft Computing in Case-Based Reasoning*, pages 1–18. Springer, 2000.

[22] B. Malysiak, D. Mrozek, and S. Kozielski. Processing fuzzy sql queries with flat, context-dependent and multidimensional membership functions. In *Proc. IASTED Int. Conference on Computational Intelligence, CI 2005*, pages 36–41, Calgary, Canada, 2005.

[23] J.M. Medina, O. Pons, and M.A. Vila. An elementar processor of fuzzy SQL. *Mathware and Soft Computing*, 1(3):285–295, 1994.

[24] L. Portinale and S. Montani. A fuzzy case retrieval approach based on SQL for implementing electronic catalogs. In *proc. 6th European Conference on Case-Based Reasoning*, Aberdeen, 2002. Springer.

[25] L. Portinale, S. Montani, and R. Bellazzi. Fuzzy approach to case retrieval through fuzzy extension of SQL. *International Journal on Engineering Intelligent Systems*, 10(3):159–171, 2002.

[26] J. Schumacher and R. Bergmann. An efficient approach to similarity-based retrieval on top of relational databases. In E. Blanzieri and L. Portinale, editors, *Proc. 5th EWCBR*, pages 273–284, Trento, 2000. Lecture Notes in Artificial Intelligence 1898.

[27] I. Watson. *Applying Case-Based Reasoning: Techniques for Enterprise Systems*. Morgan Kaufmann Publ., 1997.

[28] D.R. Wilson and T.R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.

[29] R.R. Yager. Case-based reasoning, fuzzy systems modelling and solutions composition. In *Proc. 2nd Intern. Conference on Case-Based Reasoning (ICCBR-97), LNAI*, pages 633–642, Providence, RI, 1997.

[30] Q. Yang, W. Zhang, J. Wu, and H. Nakajima. Efficient processing of nested fuzzy SQL queries in a fuzzy database. *IEEE Transaction on Knowledge and Data Engineering*, 13(6):884–901, 2001.