FUZZY QUERIES IN ROMANIAN LANGUAGE. AN INTELLIGENT INTERFACE

Cornelia TUDORIE
Cristian NEACSU, Ionel MANOLACHE

Department of Computer Science, University "Dunarea de Jos", Galati
Domneasca 111, 800008 Galati, Tel, Fax: 460182
email: Cornelia.Tudorie@ugal.ro

Abstract: The most accessible interfaces querying databases must be so intelligent, able to understand natural language expressions and including vague terms in selection criteria. The paper proposes a general architecture for a flexible database interface and also a real implementation of such a system. It is general enough, so it can be connected to any database, after a specific knowledge base description. The natural language processing is mainly based on lexico-syntagmatical analyse; the vague criteria interpreting and evaluating are based on the fuzzy logic.

Keywords: Artificial Intelligence, Database, Fuzzy Logic, Fuzzy Queries, Natural Language

1. INTRODUCTION

Almost all languages for relational database systems are respecting the SQL norms. They are based on a Boolean interpretation of the queries: the selection criterion is always a logical expression, and the answer always contains only these tuples for what the expression takes the *true* value.
In an imprecise query, the selection criterion is linguistically expressed by vague terms and it can be satisfied to a certain extent, by some of database tuples. But, the classical systems don't accept this kind of queries.
On the other side, the direct access to databases, even for data selection, is possible through an artificial language; only using the language commands, we are able to express a correct and coherent query.

What can be a possibility for the operator, when he often wants to send vague queries to the database, using informal and familiar terms, avoiding the syntax rigidity?
As an answer to this question, we propose a flexible system architecture; actually it is an intelligent software interface, able to be connected to any classical relational database system, and it acts as a superior level of the architecture, coming more close to the operator natural behaviour.

Firstly, we will present the objectives of the proposed system; after that, the functionality and the structure of each component will be presented, and also the interactivity between them.
Finally, a real system, able to Romanian language vague queries processing, implemented at our Department, comes as a good example.

2. THE MAIN OBJECTIVES

Before discovering the system's characteristics, it is necessary to define the context where our proposed solution is applicable and the background took in account in our problem.

The access to databases is possible in the following two ways:

- operating with application programs, when a limited set of predetermined functions are available and
- operating directly on data, using relational command languages.

The second one is unavoidable when an occasional operation, in particular terms, is performed. Our

This paper was recommended for publication by Sergiu CARAMAN

problem regards exactly this situation, when the human operator is **directly connected to the database, in order to search data**. This kind of interaction is often useful to those who are not specialists in informatics; they are interested only in data looking up. They may be managers or analysts in various domains, or simply individuals accessing a public database.

The proposed system is one of the *flexible interfaces for database querying*. What are they, their aims, their functions, and the theoretical or practical approaches to their development, can be found in (Tudorie, 2003c).
In the following, some desirable features of the flexible user interface to access database will be presented.

The artificial languages generally require rigorous rules for the query expressing. Therefore, the direct access to the database through the query language is used only by the specialists; they well know not only the syntax, but the logic sense of the query. But there are other users which don't have enough knowledge to realize and express correct and precise queries (knowledge relating the conceptual data model, or relational structures, or relational language syntax); even if their target is clear enough. There is a common need to offer a better possibility, a more 'natural' one to express the request of the user, without having to accept the strictness of a command language.

The conventional database systems using query languages typically offer a mean to specify the selection criteria, as complex as it can be, very precise expressed, using Boolean expressions. The rigidity and specificity of the commonly used query languages can cause an empty result or a too complex one; in both cases the information is useless to the user.
A similar situation can be found when the domain of an attribute is very wide, the values are too varying and concrete, so the user has difficulties knowing or expressing precise criteria.
The solution would be accepting approximate or vague criteria in the search query; so only objects of a certain area of interest would be retrieved from the database. A natural consequence of accepting such type of criteria will be a result as a list of database objects, ordered by the grade of satisfaction of the original query criteria.

Generally following the idea to offer a much more natural access to a database, we set our point of view to the two aspects giving the flexibility of the interface:

- The possibility to express **the database query using the natural language (Romanian language)**. The vocabulary that can be used by the operator will be necessarily restricted to those specific terms of the domain the database is operating. The natural language query is translated into an equivalent form, using the command language for databases (standard SQL may be chosen), and then processed by the database server.
- The possibility to include **vague terms** into the query, more exactly into the selection criterion of the database tuples. Those tuples will then be correctly interpreted, accordingly to their signification, and the satisfaction degree of the selection criterion will be computed for each of them.

Other characteristics of the database query system that we consider refers to:

- The interpretation of the result to establish the satisfaction grade of the vague selection criteria; the tuples will be displayed in the natural order of the computed satisfaction degree, and the ones that don't respect the criterion at all will not be displayed.
- The generality character of the interface; it will act as an intelligent system, able to be connected to any relational database; provided that a particular set of meta-knowledge has been prepared, specifically to the database and the linguistic context.

To demonstrate the main characteristics of the system, without describing the details, we present a query expressed using the Romanian language, containing vague terms, and the response returned by the system.

Example 1. The answer to the question

'Care sunt studenții ***tineri*** *dar* ***mai ales*** *cu* ***notă bună*** *?'*

('Who are the young students but mostly with good mark?')

using the following table

STUD

Nume (Name)	**. . .**	**Nota (Mark)**	**Varsta (Age)**	**. . .**
Elena		7	20	
Ioana		6	23	
Maria		8	21	
Paul		9	26	
Vasile		4	22	
Costel		8	24	
Ion		10	20	

is:

Nume (Name)	Varsta (Age)	Nota (Mark)	Satisfaction degree of the criterion
Ion	20	10	1
Maria	21	8	0.66
Costel	24	8	0.66
Elena	20	7	0.6
Ioana	23	6	0.55
Vasile	22	4	0.5

3. THE INTERFACE'S COMPONENTS

Figure 1 presents a block-scheme that shows the architecture of the interface at a general level, but more importantly shows its position regarding the interaction with the user on one side, and with the database on the other side.

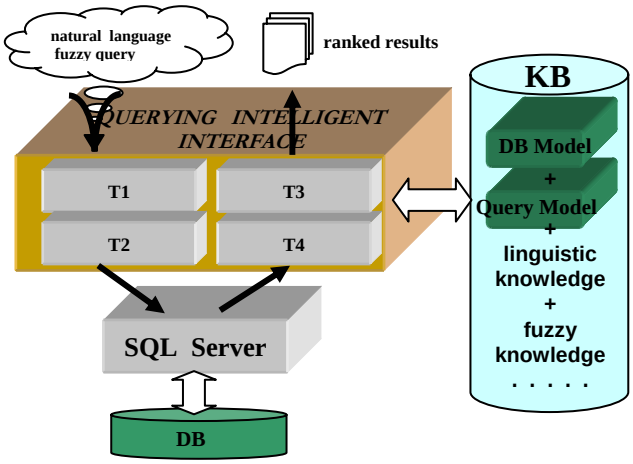

Fig. 1. The architecture of a flexible interface for databases

The interface is able to accept user queries, as natural language phrases and containing vague terms; after the query interpreting, a covering crisp query is generated, using the classical SQL language, and sent to the database server. The answer is then interpreted by the interface in order to evaluate the degree that the database tuples are satisfying the selection criterion, and the final result is displayed to the user.
It's easy to notice that the intelligent interface has a knowledge base associated that is specially built for that one database.

More elements are needed:

- knowledge referring the database scheme
- linguistic knowledge
- knowledge referring the definition of the vague terms that can be present into the query.

The same knowledge base serves to store the temporary internal model of the query as it is generated.

The component modules of the interface are figure on the scheme and described in the following. The common characteristic of all modules is their **translator** role, which means that, the input object (the text of the query or a table of values) in a certain form, is translated into another equivalent object but in a different form.

❑ *The T1 translator*

The module identified by **T1** is by itself a complex system, which handles the initial phrase, typed by the user. Generally the query is expressed using natural language and contains fuzzy terms. The T1 module (fig. 2) transforms the natural language (Romanian language) query by translating it into an intermediate language which is equivalent and much more close to the database query language, SQL. The semantics of the initial request and the fuzzy nature are maintained. The fuzzy model of the query is created at this step.

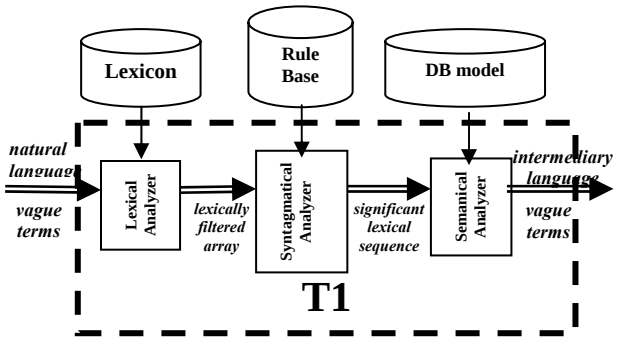

Fig. 2. The block-scheme of the T1 module

This module processes the query by executing certain transformation steps. Each step is figured on the schema by a sub-module that has a well known role during the transformation.

The lexical analyzer identifies within the original phrase the words that are defined in the lexicon and replaces them with their equivalent form, either a synonym or a non flexed form of the word; usually this word is the lexical root of the identified word. The result is an array of words *lexically filtered*.

The syntagmatic analyzer handles the array of words produced earlier, it successively applies some transformation rules, and in the end it generates an intermediary list of symbols, directly interpretable by the semantic analyzer, named *significant lexical sequence*.

The semantical analyzer transforms the list of symbols generated by the syntagmatic analyzer and, by consulting the database graph (the description of the data model), it generates an equivalent form of the initial query, expressed using an intermediary language.

The whole ensemble that forms the T1 module will be designed to be general, able to be used for any application, that is any database in any domain. But the auxiliary information that is used by this module (the lexicon, the rule base, the database model) is specifically created for every particular application domain.

❑ *The T2 translator*

The module identified by **T2** has the role to understand the form that the T1 translator has generated (the fuzzy form of the query) and to create the final SQL form. Moreover, the **T2** module (fig 3) has to interpret the fuzzy terms contained in the intermediate form and generates "where" clauses, by modifying the vague criteria into crisp ones SQL command. This module uses a knowledge base containing the information necessary to transform the fuzzy terms into SQL clauses.

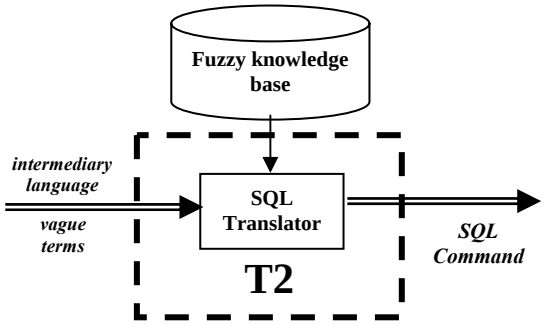

Fig. 3. The block-scheme of the module T2

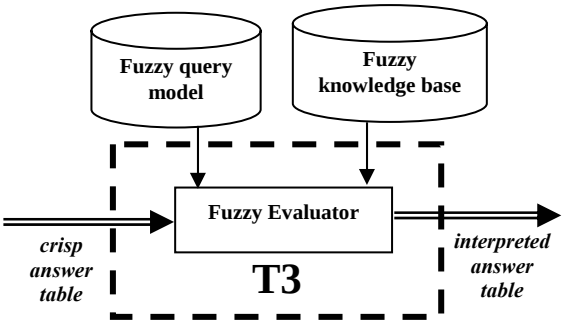

Fig. 4. The block-scheme of the T3 module

❑ *The T3 translator*

The module identified by **T3** (fig. 4) is to be found on the feedback way of the data, and it receives the answer from the database system and interprets it accordingly to the significations of the fuzzy terms present in the initial query.

The answer received from the server is a table that contains all the information requested by the fuzzy terms. This module computes for each tuple the percentage, or a degree that it satisfies the global fuzzy criteria. This degree is a [0,1] number according to how much or less the criteria is satisfied.

❑ *The T4 translator*

The processed answer by the **T3** translator is then analyzed by the next module, the **T4** module, to make the final adjustments in order for this answer to be presented to the user in a comprehensible format. The precision of the answer may be reflected by the user's preferences, and it can be:

- very restrictive (containing only the tuples that certainly satisfy the criteria, meaning that the satisfaction degree = 1)
- very permissive (displays all the tuples, corresponding to not null satisfaction degree)
- moderate (displays the tuples satisfying at least on half the criteria, degree > 0.5)
- gradually (displays the resulted tuples, with their satisfaction degree, ordered by it's value)
- partially (displays only the tuples satisfying in a certain measure the criteria)
- or another way.

One can see that the proposed interface interacts on three directions:

- firstly with the user, reading it's request, and presenting back the answer.
- with the knowledge base, at all course of interpreting the query, both for the natural language understanding and for the vague terms processing.
- and finally with the database, that is the main information support.

4. KNOWLEDGE TYPES. REPRESENTATION AND PROCESSING

The flexible interface is a part of the intelligence system category. During the processing, certain types of information (knowledge) are manipulated, like: some information are previously presets and load to the knowledge base (static knowledge); other are generated, like intermediary results, or like transfer information among modules (dynamic knowledge); some of the last ones are stored in the knowledge base and others in the external memory only.

In the following, all types of knowledge of the proposed system, are presented one by one.

4. 1. Linguistic knowledge. Interpretation of the natural language query

The linguistic knowledge interferes in interpreting of the natural language query, sent by the human

operator. Some types of linguistic knowledge are necessary in the sentence analyzing and that result from module T1 running (fig. 2).

❑ *The Lexicon*

The Lexicon represents the collection of all synonym classes, each of them are represented by a standard word.

The words which are contained in this vocabulary can interfere with the expression of the natural language query, which are addressed to a domain specific database. Obviously, the lexicon has to be complete (as many possible), in order to any word from a sentences to be correctly lettered, to be part of one but only one class. These words can represent following:

- a relevant term for the database (attribute name, table name, for example) or
- a term which can be an operator (logical or aggregation), or
- fuzzy terms, which participate at criteria selection statement, or
- words with no database signification, but be part of syntagmes (are defined in previous paragraph)

Example 2. For example, some lexicon (for the Romanian language):
DISCIPLIN →DISCIPLINA DISCIPLINE DISCIPLINELE OBIECT OBIECTUL OBIECTE OBIECTELE MATERIA MATERIE MATERII EXAMEN EXAMENUL EXAMENE EXAMENELE
LOCALIT →LOCALITATE LOCALITATEA ADRESA ADRESE ADRESELE LOCUIESTE LOCUIESC ORAS DOMICILIU DOMICILIAZA
EGAL →EGAL EGALE EGALI
SAU→ SAU
CAT →CAT CATI CATE CIT CITI CITE
JUR →JUR JURUL
APROXIMATIV →APROXIMATIV CIRCA
RESTANT →PICA RESTANTA RESTANTE RESTANTIER RESTANTIERI PICAT PICATI
FIECAR→ FIECARE FIECARUI FIECARUIA
DE→ DE
CU→ CU

The words which are not present in the lexicon are considered as constant values, which are part of the selection criteria and are marked in distinct way.

❑ *Rules base (syntagmatical rules)*

The context is a compact words group. It can be an empty group ("~").
The syntagme is a pair of two contexts which, appearing in the text at indefinite distance, define a certain concept of the speech universe.
The rule base is a collection of production rules which are applied for the iterative transformation of the lexical filtered string until to significant lexical sequence.

Structurally, a rule is defined by following:

(ss, {s1, s2, ... sn}) => (sd, <cd1, cd2>)

where:

- ***ss***, ***sd*** are the left state and the right state, are integer numbers representing the analyze automate states (syntagmatical analyser) before and after the rule is applied.
- ***s1, s2, ... sn*** are the left syntagme. Every syntagme is: *<cs1, cs2>*, where *cs1*, *cs2* are two left contexts. At list *cs1* must be different of empty string ("~"). Notice that the rule given represents only a compact writing of *n* array rules with the same *ss, sd, <cd1, cd2>*, but with another pair of left contexts *<cs1, cs2>*;
- ***cd1***, ***cd2*** are two right contexts; them also can be the empty string.

The contexts which are part of the rules are called pattern-contexts. For every pattern-context exists a multitude of context-instances; this is the case of constants for example, which inside of the rule are interfering with there indicative. For example, rule:
`(1,< DIN &C,~ >) => (1, < LOCALIT &C,~ >)`
can generate in the same manner any of the following transformation:
`DIN &C_GALATI → LOCALIT &C_GALATI`
`DIN &C_TECUCI → LOCALIT &C_TECUCI`

4. 2. Database model. Query command generation

Our system is able to connect as interface to any relational database.
The database model which the semantically analyzer module is based contains the structural description of the database, that is:

- tables name
- table structure (columns name and type)
- the relations between the database tables

A set of syntax rules to show the data model, can be:
<database>::= <table_description>
< table_description >. . .
< table_description >
<table_description> ::=<table_name><column_list>
<column_list>::= <column_description>
<column_description>. . .
<column_description>
<column_description> ::=
&<column_type>_<column_name>[<link>]
<column_type> ::= **N** | **C** | **D** | **L**
<link> ::= <**integer_number**>
Example 3. Now let's look at the database structure that we have used to evaluate the natural language queries:

STUDENT &N_*MATR***3** &C_*NUME* &C_*LOCALIT* &N_*TEL* &D_*DN* &N_*GRUPA* &N_*AN* &C_*SPEC*
DISC &N_*CODD***3** &C_*DEN* &C_*PROF*
CATALOG &N_*CODS***1** &N_*DIS***2** &N_*NOTA*

4. 3. Fuzzy knowledge. The fuzzy queries and answer interpretation

Fuzzy knowledge is used to the fuzzy criteria interpreting, which sometimes may be included by the operator in the database queries.

Also this knowledge is very important in the answer interpretation, in order to provide a useful response to the operator.

The knowledge is represented by a list of fuzzy terms definition, significant for the database domain application.

The theory of fuzzy sets and fuzzy logic is the appropriate framework in the vague expressions modelling and vague queries evaluation.

Fuzzy predicate is an assertion that may be more or less true, depending on the variables values. It is an extension of the classical logical predicate, that can be either definite true, or definite false. The truth-value of the vague predicate may be express as a number in [0,1], with standing 1 for definitely true and 0 for definitely false.
Usually, the fuzzy predicate is defined by a fuzzy set or a fuzzy number.

The following description must be contained in the fuzzy knowledge base:

- ***linguistic domain*** of the tables attributes, which can interfere in the data selection fuzzy criteria; it contains a set of labels corresponding to the fuzzy predicate which are defined also as fuzzy sets on the attribute crisp domain.
- common ***linguistic modifiers*** (applicable for every fuzzy predicate), or specific linguistic modifiers (applicable to particular fuzzy predicates).
- ***linguistic comparators***, which generally are common and are applicable for every linguistic domain
- ***fuzzy logic connectives***, which are generally used in any fuzzy logic expression
- ***linguistic quantifiers***, applicable for any group of tuples, not mater what the selection criteria is.

A review of several categories of linguistically terms with vague meaning, their fuzzy model and specific operations are presented in (Tudorie, 2003a), (Tudorie, 2003b), and many others.

We can notice the **general** character of the system, explained by following:

- all particular information, specific to the database, have been grouped into a knowledge base very large, perfectly separated by the interface itself.
- the database query command is generated in the SQL standard language, known by the commonly relational database systems.

Therefore, to adapt the system for a new database, from a new domain, we have to:

- make a new knowledge base, for the particular application domain;
- eventually, a minor configuration of the connect parameters, only in case we choose another system to store the database.

5. **RoFQuery** – FUZZY QUERYING IN ROMANIAN LANGUAGE INTERFACE

The ***RoFQuery*** system is an intelligent database interface built like the model presented previously. The system allows database querying, without interact with standard tool for access that. The natural language is the user expression means; it is about a limited language (vocabulary), containing only terms proper to the database domain application. The general scheme, which explains the information flow in the RoFQuery interface, is presented in figure 5.

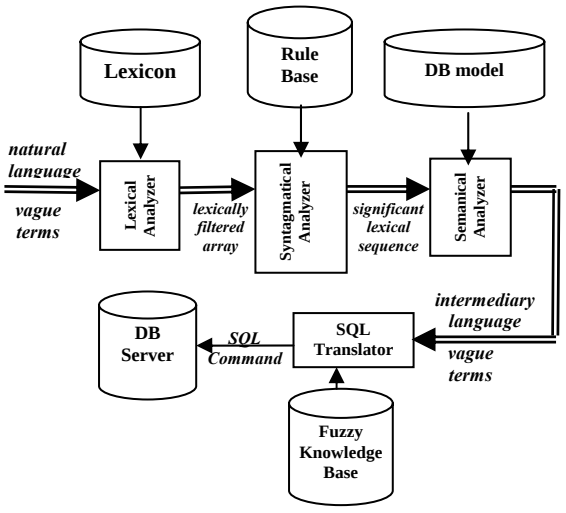

Fig. 5. Query processing steps in ***RoFQuery***

The main functions of the system are:

- *analyzing the natural language sentence* entered by user, *containing or not fuzzy terms* ; translating it in a crisp formal language, and sending it as a usual query to the database server.
- *connecting to any database*, but only after already having prepared a knowledge base proper to the database context.

- *analyzing each row of the answer from the system* and assigning a value according to the satisfaction degree of fuzzy criteria.
- *assisting the knowledge engineer to update the knowledge base, directly throw the interface.* The connection is possible anytime from anywhere throw web interface, login and password.
- *providing a password and user privileges system.* The administrator has all privileges about the knowledge base and database; other users only to query the database.

In figure 6 the main components of the system are presented, with interactions between user and information contained in the system.

The lexico – syntagmatical analyzer transforms the user phrase, using the knowledge from the system, to an array of symbols, named *significant lexical sequence*. The analyzer, firstly executes a lexical analyse and next a collocation analyse. The lexical analyzer deals with elimination of separators from user phrase and then normalize the input, replacing the words with their representatives from the lexicon. Any word which it is not founded in the lexicon it is considered a constant value, a lexical atom which have meaning in association with others.

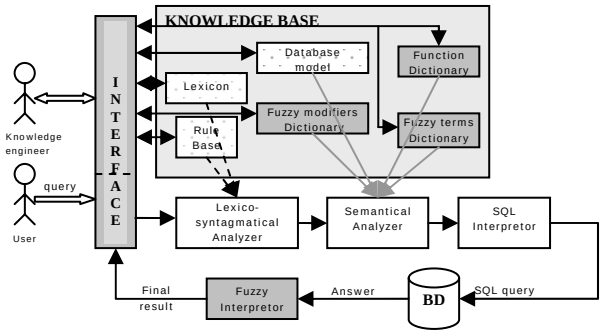

Fig. 6. The ***RoFQuery*** system’s architecture

Example 4. The query

’Ce studenți foarte batrâni au nota în jur de 8 la Grafică?’

(*’What student very olds have grade close to 8 at Graphics?’*)

is addressed to the database:

STUDENT [matr, nume, localitate, dn, grupa]
DISC [codd, Den, Prof]
CATALOG [matr, codd, nota]

The result of the lexical analyse is:

* STUDENT DEN &C_GRAFICA NOTA &F_JUR &N_8

The syntagmatical analyzer takes up the list of words and transforms it, throw successive operations, to a list of symbols which is interpreted by the semantic analyzer. It uses the rule base existing into the knowledge base.

Example 5. In the process of syntagmatical analyze, the query from example 4 goes throw following steps:

Step 1 : CE STUDENT FOARTE BATRAN ARE NOT #JUR &N_8 LA &C_GRAFICA $$
Step 2 : CE STUDENT FOARTE BATRAN NOT #JUR &N_8 LA &C_GRAFICA $$
Step 3 : CE STUDENT FOARTE BATRAN NOT #JUR &N_8 DEN &C_GRAFICA $$
Step 4 : * STUDENT FOARTE BATRAN NOT #JUR &N_8 DEN &C_GRAFICA $$
Step 5 : * STUDENT FOARTE DN #BATRAN NOT #JUR &N_8 DEN &C_GRAFICA $$
Step 6 : * STUDENT DN FOARTE #BATRAN NOT #JUR &N_8 DEN &C_GRAFICA $$
Step 7 : * STUDENT DN #FOART #BATRAN NOT #JUR &N_8 DEN &C_GRAFICA $$
Step 8 : * STUDENT DN #FOART #BATRAN NOTA #JUR &N_8 DEN &C_GRAFICA $$

A similar natural language analyse processing is presented in (Cristea, 1987) and it has strongly influenced our present approach.

The semantical analyzer takes up the array of symbols resulted from the syntagmatical analyze and builds the crisp SQL query, using knowledge from the database graph, operators dictionary, functions dictionary, fuzzy terms dictionary and fuzzy modifiers dictionary. A few notes about dictionaries:

The operators dictionary contains caption of math operators used to build SQL “where” condition. Each line looks like:

<operator_name> | <math_symbol>

For example for math operator “ >= “ :

MAREGAL|>=

The functions dictionary contains name of functions used to process data, used into SQL query.

The lines are in the following form:

<function_name>#<table>#<task> .

Example for calculate the student age:

VARSTA#STUDENT#
round (MONTHS_BETWEEN (sysdate,
to_date(student.dn, 'DD-MM-YYYY'))/12,2)

The fuzzy terms dictionary contain the fuzzy model of the vague terms, expected to be included into the user queries.

Each term is defined specifically for one table column.

The lines are in the following form:

!<table> <column>
#<term_name>|[<applied_function>]|
<int1> <int2> <int3> <int4>
<sense> <pitching>

Where *<sense>* is the direction (left/centre/right) of the action of the condensation modifier (or, in other

words, the position of the linguistic value on the attribute domain),
and <*pitching*> refers to, on modifier action, the fuzzy support is limited or not, in the direction given by <*sense*> (left/right).

For example:
!STUDENT DN
#TANAR|VARSTA|0 0 20 22 s b
The defined term is #TANAR and applicable to field DN (birthday) from table STUDENT, throw function VARSTA.

The fuzzy modifiers dictionary contains the model of the linguistic modifiers which can be used in queries expression. They are universal and can be applied to any fuzzy term, which his definition can be extended to left or to right. The lines are to following form:

#<modifier_name>|< percentage>

For example:
#FOART|25

The fuzzy modifiers can be used in some fuzzy terms context, with role to modify their definition. In previous example, the modifier #FOART is defined like to represent 25%; so that, the support and core which define the fuzzy term, is modified accordingly to modifier percentage.

Example 6. Following-up the query evaluation from example 4 and 5, the result of the semantical analyze is:

```
SELECT
   STUDENT.*,
   year(sysdate()) - year(STUDENT.DN) VARSTA ,
   CATALOG.NOTA,DISC.DEN
 FROM
   STUDENT,CATALOG,DISC
 WHERE
    year(sysdate()) - year(STUDENT.DN) > 28.75
  AND
    year(sysdate()) - year(STUDENT.DN) <= 100.0
  AND CATALOG.NOTA > 7.5
  AND CATALOG.NOTA < 8.5
  AND DISC.DEN LIKE '%GRAFICA%'
  AND STUDENT.MATR=CATALOG.CODS
  AND CATALOG.DIS=DISC.CODD
 ORDER BY
   CATALOG.NOTA, VARSTA DESC,
    STUDENT.DN, DISC.DEN
```

The SQL interpreter is the one which connecting to database, through the database management system, deals with the SQL query running.

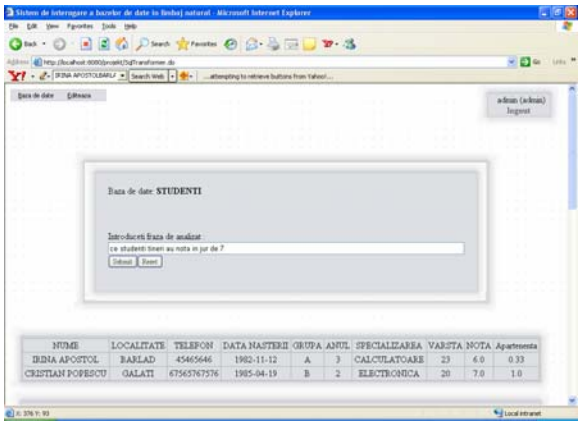

Fig. 7. The user interface of the ***RoFQuery*** System

The fuzzy interpreter analyzes the answer from the database system and processes it, gives to each answer line a value representing the fuzzy criteria satisfaction degree.

The user interface, functionally speaking, is split in two parts, by the role regarding the connected user. So that, the knowledge engineer, with ‘administrator’ role has right to modify any dictionary or parameter involved in the internally functionality of the system (that is the knowledge base) and in the same time can query the databases; the simple operator can only send queries to the system.

The user interface is Web type, developed with last technologies, which result in a simple installation, connecting and utilization (user friendly interface). Application screenshot in figure 7.

6. CONCLUSIONS

This paper presented a flexible intelligent interface for database querying. The two big advantages of the system are: using natural language (Romanian language) for expressing of the query (not need to learn SQL language) and using fuzzy terms (for expression gradual properties, approximation or intensify properties). The system has a knowledge base containing linguistic knowledge for the database application domain, and fuzzy terms definition. The comprehension level of query (that is the system flexibility) is higher that the knowledge base is more completed. So that, through the user interface, the knowledge base can be more continuous enriched by any authorized user, having the role of the knowledge engineer. We remark the general character of the system, which can be connected and used like an interface to query any database, but with condition the particular knowledge base must be prepared in advance for this task.

7. REFERENCES

Project BADINS. (1995). Bases de données multimédia et interrogation souple. *Rapport d'activité scientifique*, Institut de recherche en informatique et systèmes aléatoires, Rennes

Cristea, D. (1987). Sistemul QUERNAL, In C. Giumale, D. Preoţescu, L.D. Şerbănaţi, D. Tufiş, D. Cristea (eds.), *LISP*, Editura Tehnică, Bucureşti, 1987, pp 215-229

Dubois, D., Prade, H. (1996). Using fuzzy sets in flexible querying: Why and how?, In H. Christiansen, H.L. Larsen, T. Andreasen (eds.), *Workshop on Flexible Query-Answering Systems*, pp. 89-103

Kacprzyk, J.., Zadrozny, S. (2001). Computing with words in intelligent database querying: standalone and Internet-based applications, *Information Sciences*, **134**, Elsevier, pp.71-109

Pivert, O. (1991). Contribution à l'interrogation flexible de bases de données - Expression et évaluation de requêtes floues, T*hèse*, Université de Rennes I

Tudorie, C. (2003a). Cercetări privind aplicarea tehnicilor de inteligență artificială pentru interogarea bazelor de date. Scientific Rapport, University 'Dunărea de Jos', Galați

Tudorie, C. (2003b). Vague criteria in relational database queries. In *Bulletin of "Dunarea de Jos" University of Galați*, **III/2003**, pp. 43-48

Tudorie, C. (2003c). Contribuții la realizarea unei interfețe inteligente pentru interogarea bazelor de date. *Scientific Report*, University 'Dunărea de Jos', Galați, 2003

Yager, R.R. (1991). Connectives and quantifiers in fuzzy sets, In *Fuzzy Sets and Systems*, **40-1**, Elsevier Science, pp 39-75