Enriching Vague Queries by Type-2 Fuzzy Orderings

Saad M. Darwish[1], Tamer F. Mabrouk[2], and Yasser F. Mokhtar[1]
[1]Institute of Graduate Studies and Research, Alexandria University, Alexandria, Egypt
[2]Alexandria Higher Institute of Engineering & Technology, Alexandria, Egypt
Email: saad.saad@alexu.edu.eg, tamer.fouad@aiet.edu.eg, yasserfakhry@hotmail.com

***Abstract*—Structural Query Language (SQL) is very restrictive and very dominant tool that handles data that is crisp and precise in nature; but it is unable to fulfill the needs for data which is uncertain, imprecise, and vague in nature. The human queries are rarely crisp, which need unusual requirements to deal with it based on world knowledge. These requirements are called Fuzzy Queries (FQ) that realizes some degrees of truth. Mixing the concepts of fuzzy set theory and SQL, FSQL is able to process imprecise and ambiguous data and also able to increase the facility of data retrieval based on linguistic terms. This paper describes a flexible query interface based on type-2 fuzzy logic. Hence, queries in natural language with pre-defined syntactical structures are executed, and the system uses a type-2 fuzzy process to provide answers. Type-2 fuzzy logic (T2FL) system offers the capability of handling a higher level of uncertainty than regular fuzzy logic, which is heavily used in the previous works. T2FL can be used when the situations are too uncertain to decide the exact membership functions. FSQL seems to be a practically feasible and efficient approach to contract with queries for crisp data that include a certain tolerance for imprecision compared to its SQL counterpart. Many experiments have been made on real database that show the effectiveness of the proposed model compared to the existing type-1 fuzzy systems and also show the high accuracy in the results.**

***Index Terms*—SQL, interval type-2 fuzzy logic, fuzzy linguistic values, fuzzy query, fuzzy database**

I. Introduction

Database systems are considered one of the actual vital tools of data processing in terms of protection, administration and retrieval the information particularly with the vast amount of data and the complexity of operations on them. The most database applications are based on relational database but in real-world applications, there are a big number of requirements are not being implemented because of lack of accuracy and clarity of the data which is a bit vague, imprecise and uncertain. It is no doubt that the quality and accuracy of information directly aid to make the right decision and be very valuable in data-intensive applications (e.g. CAD/CAM, geographical and environmental information systems, and decision support systems) [1].

Although SQL is a very powerful tool, it is impotent to placate needs for data selection based on linguistic expressions and degrees of truth [2]. Linguistic expressions are motivating for data extraction, analysis, dissemination and decision making. Several real applications need to manage vague or fuzzy information and to make benefit their users from flexible queries. Fuzzy set theory is a useful tool to handle imprecision. The application of this theory in the area of fuzzy databases, to compact with imprecision and vagueness has been widely addressed in the literature [2]-[4]. Fuzzy query is not only a querying tool; it improves the meaning of a query and extracts additional valuable information [5].

In general, the study area of fuzziness in database management systems (DBMS) has resulted in a number of models aimed at the representation of imperfect information in databases (fuzzy database), or at enabling non-precise queries (often called fuzzy queries) on conventional database schemas [1]-[5]. There are also other concerns in the use of fuzzy sets theory in relational databases such as efficiency of fuzzy queries execution; fuzzy functional dependencies/ constraints, fuzzy logical databases, but they are beyond the scope here. The crucial idea in fuzzy queries consists in extending the SQL language and adding a supplementary layer to the relational DBMS to assess the fuzzy predicates [6]. These fuzzy predicates permit to have a range of answers (each one with a membership degree under shapes of linguistic expressions) in order to offer to the user all intermediate variations between the completely satisfactory answers and those completely dissatisfactory.

Two possible solutions to implement a fuzzy relation database (FRDMS) [1], [4] :(1) develop a specific Fuzzy DBMS to evaluate the queries written in FSQL, by analogy with the strategy put in work in the usual DBMS, but the development cost would risk to be prohibitive, (2) use the capacities of the commercial DBMS (in particular their mechanisms of optimization) while attaching a software layer that allows to support the fuzzy concept. The last solution, characterized by its easiness realization, consists in cooperating the FSQL server and the DBMS. This solution has been concerned with the problem of how query interfaces to conventional databases with crisp data can be extended such that a flexible explanation of queries is possible-in particular, with the motivation to advise alternatives which are close to match the criteria in case that a query fails completely.

Manuscript received May 12, 2014; revised August 1, 2014.

doi: 10.12720/lnit.2.2.177-185

FSQL is an add-on to traditional relational databases that acts as a proxy between the user and the database [6]-[9]. Since FSQL connects with the underlying database only on the basis of standard SQL, no alterations to the database system or the data model have to be made, which lets easy integration into existing applications. Fuzzy upgrading of SQL queries has advantages in cases when the user cannot unambiguously define selection criteria or when the user wants to examine data that almost meet the given criteria.

The concept of a type-2 fuzzy set was announced as an extension of the concept of an ordinary fuzzy set. Type-2 fuzzy sets allow us to handle linguistic uncertainties or vagueness, as typified by the adage "words can mean different things to different people"[10]. A fuzzy relation of higher order (e.g., type-2) has been regarded as one way to increase the fuzziness of a relation and hence raises ability to handle inexact information in a logically correct manner. Type-2 fuzzy set is characterized by a fuzzy membership function, i.e., the membership grade for each element of this set is a fuzzy set in [0,1], unlike a type-1 set where the membership grade is a crisp number in [0,1]. Such sets can be used in situations where there is uncertainty about the membership grades themselves, e.g., an uncertainty in the shape of the membership function or in some of its parameters [11].

A. Paper Motivation and Contribution

The objective is to provide users with new querying capabilities based on conditions that involve preferences and describe more or less acceptable items, thus defining flexible queries. Since the problem is no longer to decide whether an element satisfies (or not) a condition but rather the extent to which it satisfies this condition, one of the advantages lies in the "natural" ordering of the answers (decision) that allows for calibration if chosen.

In this paper, we provide the key ideas how the functionality of FSQL can be extended such that a flexible interpretation of conditions like “is at least”, “is at most” and" is about" can be maintained. We extend the work of J. Mishra [9] to enhance dealing with fuzzy linguistic values on crisp database by using type-2 fuzzy logic, which can be very useful to handle high levels of uncertainties properly specially with the large size of databases. This language grants new concepts such higher order fuzzy attributes. A further important feature of the suggested system is the possibility of weighting both predicates and operands of algebraic operators so as to better fit user preferences/requirements and to capture more meaning of the data with suitable interpretations for the type-2 fuzzy membership functions.

Besides this introduction, this paper includes four sections. Section 2 presents the architectures already used for the flexible querying modeling. Section 3 presents our new architecture of the fuzzy query. Section 4 makes an evaluation of this work and Section 5 gives conclusion and some future perspectives of it.

II. Literature Overview

FSQL area of research is not new one but there are still many opportunities for the enhancement of existing approaches and for producing new approaches. Although there are some variations according to the discriminations of different implementations, the answer to a fuzzy query sentence is generally a list of records, ordered by the degree of matching. Brief analysis of these proposals and their variations can be found in [1], [12]. The most advantages of these proposals that it lead to develop many techniques to handle vague and imperfect data to get more accuracy. In the contrary of that, most of them use the fuzziness in database through fuzzy database which require some alterations in database structure and this involves new entities like fuzzy conditions, fuzzy comparators, fuzzy constants, fuzzy constraints, fuzzy thresholds, linguistic labels and so on. Moreover, these studies are limited to just some specific applications and not stranded on theories of fuzzy database query languages.

Research work on emerging a flexible natural language interface for relational crisp databases has practiced growth at a very high rate. This has led to incessant research on natural language interfaces and query execution related issues. Most existing natural language interface to relational databases (NLI2DBs) are quit stiff in interpreting natural language queries. They just look for keywords in the sentence or using some patterns in analyzing the user's input. Such approaches cannot deal with questions in random formats. For instance, the authors in [13] proposed architecture for fuzzy querying along with an experimental implementation of the same. The implementation is using some patterns that assist the lexical analysis of fuzzy terms and parsing the fuzzy query individually. Fuzzy query is interpreted by the parser and the resulting semantic actions are carried out on MySQL database. However, most NLI2DBs are domain-dependent, as they need predefined knowledge of the working domain in building templates or semantics rules.

In the context of fuzzy query language, many researchers proposed extension to relational algebra in order to develop a fuzzy SQL that offers the means for performing queries with some uncertain concepts. For example, the fuzzy query approach based on the fuzzy Generalized Logical Condition (GLC) was presented [7]. This GLC enables matching fuzzy and classical constraints in the same where clause and selects only records that have the query satisfaction greater than zero (true). It is also possible to use additional filtering functions to choose suitable number of records or to set the threshold value of the query command interface. Authors in [6] and [12] provided new technique to improve the fuzzy GLC for the where part of SQL in classical relational crisp databases. In this way, fuzzy queries are accessing relational databases in the same way as with SQL. In [4], the authors discussed how 'IS' predicates in the flexible query can be evaluated in the presence of data that is modeled by fuzzy sets.

The work done by A. Branco et al. [14] has shown a methodology that automatically creates fuzzy queries from a training data set. The fuzzy queries are translated

from a set of fuzzy weighted classifier rules. A pruning procedure to simplify the fuzzy rule base and the resulting set of fuzzy queries was also proposed. The fuzzy queries are able to retrieve an ordered list of the records according to the fuzzy rule based model learned by the fuzzy classifier. The precision and recall analysis of the selected records of the fuzzy query allow the user to select an optimal threshold for other data.

In [15] the authors presented a new approach for fuzzy query processing based on automatic clustering techniques. Their proposed algorithm does not need to define the number of clusters of the data in advance, which makes it more convenient and more flexible to cluster data. Other researchers in [16] presented a new aggregation operator and the corresponding algorithm to evaluate the fuzzy query. The main idea is to dynamically define sets of linguistic labels on limited attribute domains, determined by previous fuzzy selections. This operator provided an accurate model for the discussed vague expression, with respect to query semantic.

An important work presented by J. Mishra [9] where the aim is to develop a formula which will first generate the SQL statement of a given fuzzy query. Next the generated SQL is supplied to the database to get the resultant table. In this architecture, the author has defined an algorithm to find the membership value for each tuple on the relation based on the fuzzy attribute on which fuzzy query made. Next decision maker will supply a threshold value based on which corresponding SQL of a given fuzzy query will be generated. In this case, type-1 fuzzy logic is used to model attributes (i.e. membership value for each tuple is crisp number in [0, 1]).

In general, management of SQL using traditional type-1 fuzzy sets to build flexible query cannot handle high levels of uncertainties appropriately particularly with the huge size of the databases and the vast amount of data that is usually largely similar and difficult to deal with it. In this paper, a revised system has been developed to harness the advantages of using type-2 fuzzy set inside SQL to retrieve records with high levels of uncertainties. The investigational appraisal shows that this type-2 fuzzy query system can yield good results on real world database, demonstrating its effectiveness towards solving the problem facing type-1 fuzzy query to significantly improve the robustness of database query operations.

III. Methodology

The suggested prototype is constructed by the addition of a layer around a classic DBMS as shown in Fig. 1. The translation mechanism generates a procedural evaluation program and determines the expressions that are used to compute the membership degrees and separate if necessary the tuples whose degree is lower to the fuzzy threshold. Furthermore, a meta-base named type-2 fuzzy meta-knowledge is defined that is formed by a table that extend the DBMS dictionary or catalog in order to store all necessary information to describe linguistic hedges and to manipulate membership values.

The present framework fully exploit fuzzy logic for: (1) modeling the similarity of the case attributes with respect to different approximate concepts; (2) modeling the retrieval conditions as well as the acceptability of the retrieved cases. The system deals with fuzzy attributes type 1 (FTYPE1): these are attributes with "precise data" classic or crisp (traditional, with no imprecision). However, they can have linguistic labels defined over them, which allow us to make the query conditions for these attributes more flexible. The use of flexible predicates and linguistic quantifiers interpreted in the framework of the type-2 fuzzy set theory is advocated for defining a query language.

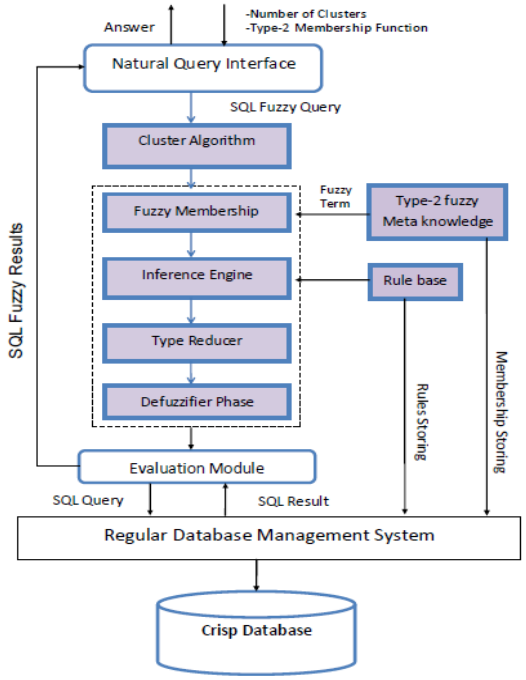


Figure 1. Proposed architecture of fuzzy SQL.

This architecture can effectively reduce complexity of processing data and maintaining a database by way of utilizing extra translation mechanism on the top of existing DBMS. It also can balance the variety of data and system performance through embedding more than type-2 fuzzy membership functions. By applying type-2 fuzzy logic and SQL to the evaluation of records, we significantly improve query robustness (i.e. we may found more suitable candidate records that are fit to the query operations). Complex multi-predicate queries can be formed by means of logical connectives, whose semantics is parameterized in order to adjust to specific scenarios. The following subsections discuss each component of the proposed architecture in detail.

A. Natural Query Interface

The interface actually is the first thing a user should encounter. Then the user gets started with the system by entering a query in his/her natural language. Many times when querying a database; users do not wish to define the precise limits of acceptance or rejection for a condition, that is, they want to be allowed some imprecision in the query. In other words, the satisfaction of a condition is a matter of degree and a flexible query should provide answers that would have had an empty response on a

classical relational SQL-type language. Moreover, it cans easily rank-order the best answers, rather than showing a long list of answers [17].

Natural language consists of fundamental terms called "Linguistic variable" and "Linguistic values". The purpose of using the linguistic variable is to provide a means of approximate characterization of phenomena that is not defined properly. In our work user must select the attribute (Linguistic variable) and its (linguistic terms or labels) like what is the employees whose have a big salary and small age. In our case, the linguistic terms for any attribute are determined through the number of attribute's clusters that is defined by the user. To avoid typing errors, the wanted scenario for the user is to determine what kind of data he wants to select by linguistic expressions and degrees of truth.

In formal, the linguistic variable is a quadruple $(V, E(V), C, M)$, where V is the name of the linguistic variable, $E(V) = \{e_i\}, i = 1,\ldots,n$ represents a set of linguistic values for the V and is ordered set $(e_i \le e_j, i \le j)$ having an odd cardinality, while C symbolizes the crisp referential domain of the V, and M is a mapping $E(V) \to \zeta(C)$ that maps a fuzzy set on C for each linguistic values of V. Thus, an order relation $\le$ on $E(V)$ is easy to define, for example little $\le$ intermediate $\le$ big. So $\le$ is a semantic order relation [18].

These linguistic variables (quantifiers) are represented as type-2 fuzzy set. This will improve the performance of crisp selection of traditional SQL where record would not be selected even if it is extremely close to the intent of the query. This is the penalty paid for using crisp logic in selection criteria. Herein, the user determines the shape of fuzzy set, lower limit of fuzzy set and value of full query satisfaction (fuzzy degree). The lower limit becomes part of the WHERE clause. This clause access the database and selects records that have Query Index (*QI*) >0. The *QI* is used to indicate how the selected record satisfies a query criterion and determined through query satisfaction. The *QI* has values from the [0,1] interval with the following meaning: 0-record does not satisfy a query, 1-record fully satisfies the query, interval (0,1)-record partially satisfies a query with the distance to the full query satisfaction. In the *QI* calculation step, the differences between fuzzy set shapes become important.

Conditions in queries contain these basic comparison operators: >, <, and = when numerical attributes in query conditions are used. In our case, these crisp comparison operators are adapted for fuzzy queries (linguistic quantifiers) in the following way: operator > was improved with fuzzy set high value, operator < was improved with fuzzy set small value and operator = was improved with fuzzy set about value. Furthermore, proportional quantifiers such as "most" can be represented by fuzzy subsets of the unit interval. According to the above analysis, three types of linguistic terms in this research are supported: high value, small value and about value of attribute. The WHERE clause contains one attribute or more attributes that are connected with fuzzy aggregation operators. These connectors are able to compute a global satisfaction degree starting from the satisfaction degrees of each vague selection criterion with respect a certain model of the fuzzy connections. Usually, the minimum and maximum functions stand for fuzzy conjunctive and disjunctive connections; the complement stands for the fuzzy negation. But there are many other propositions in the literature for defining aggregation connections [19].

In formal, a query is a set of pairs $q = \{\langle f_1, v_1\rangle, .. \langle f_n, v_n\rangle\}$ such that each f_i is an attribute (field) and each v_i is either (1) a fuzzy linguistic term defined over $Range(f_i)$ with type-2 membership function such that $Range(f_i) \to [0,1]$ 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 [16]. For instance, if the fuzzy terms young and old are defined over the attribute age, we could be interested in the fuzzy expression:

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

let us indicate $P(f_i, v_i)$ the fuzzy predicate related to the assignment of a condition v_i to attribute (linguistic values are specified). For example $P(age, young)$ is true with degree $\mu_{young}(x)$ if μ_{young} is the membership function of the fuzzy set young defined over age and the attribute age assumes values *x*. Notice that in our framework, each attribute in the database 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.

The retrieval condition induced by q is the fuzzy should be performed through OR $\cup$ connective. Predicate $R_q \equiv \varepsilon_{i=1}^{n} P(f_i, v_i)$ where $\varepsilon_{i=1}^{n}$ is a Boolean expression involving all the predicates *P*, and *n* is the number of attribute specified in the query. If for some attributes 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 $\cap$ 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 $\cup$ connective.

Given a set of tuples *c*, a query case *q* and the retrieval condition *R*, the matching degree of *c* to *q* is $\alpha(c,q) = \mu_R(c)$. So given a database *DB*, and a fuzzy threshold λ, the retrieval set is the set $S(q, DB, \lambda) = \{c_i \in DB / \alpha(c_i, q) \ge \lambda\}$. 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. In particular, we can consider the following syntax:

SELECT (λ) *A* FROM *R* WHERE f_c

which meaning is that a set of tuples with attribute set A, from relation set *R*, satisfying the condition f_c with degree $\mu \geq \lambda$ is returned. If a fuzzy operator θ is defined on attribute *f*, the expression $\langle f = v \rangle$ can be fuzzified by transforming the crisp operator = into θ.

B. Clustering

The first step after identifying variables through the interface (choosing the attributes to be queried and number of attribute's linguistic terms in each attribute) is to semantically cluster the attribute' values according to required linguistic labels. Clustering is a mathematical tool that attempts to discover structures or certain patterns in a data set, where the objects inside each cluster show a certain degree of similarity [20]. The output from a clustering algorithm is basically a statistical description of the cluster centroids with the number of components in each cluster. The k-means algorithm is well known for its efficiency in clustering large data sets [21].

Given a set of numeric attribute $F(f_i \in F, i = 1...n)$, the *k*-means algorithm searches for a partition of *F* into *k* clusters that minimizes the within groups sum of squared errors. This process is often formulated as the following mathematical problem *P* [22]:

$$\text{Minimize } P(W,Q) = \sum_{l=1}^{k} \sum_{i=1}^{n} w_{i,l} d(f_i, Q_l)$$

$$\text{Subject to } \sum_{l=1}^{k} w_{i,l} = 1 \;\; 1 \leq i \leq n, 1 \leq l \leq k \;\; w_{i,l} \in \{0,1\} \qquad (1)$$

where *W* an $N \times K$ partition matrix, $Q = \{Q_1, ...Q_K\}$ is a set of objects in the same object domain, and $d(.,.)$ is the squared Euclidean distance between two objects. The method can be very useful to develop the initial metaknowledge base, containing type-2 fuzzy definitions of vague terms in the application domain, but ever later, to maintain the actuality of these definitions with the instantly database content.

C. Parsing

In this step, type-2 fuzzy interpreter should be designed on the frame of given RDBMS using lexical and syntactical analysis of queries. The interpreter transforms the fuzzy SQL into a native SQL. Based on type-2 fuzzy attribute, the interpreter fetches the domain set of that attribute from the database and then finds the membership value for fuzzy equality of each domain value of the fuzzy attribute. If the query has more than one fuzzy attributes then in similar way it finds the membership value for equality for other fuzzy attribute. Finally it takes fuzzy intersection or union of all membership values depending on connector type between attributes (and/or) to get the membership value of each tuple of the relation.

In our case, The FSQL server uses fuzzy metaknowledge base to model the different types of type-2 fuzzy attributes. This additional table stores attributes which admit fuzzy treatment and different information related to fuzzification of the different attributes such as linguistic variables, type-2 membership values (T2MF) and description of atomic values. Using predefined parameters of membership function and attribute crisp domain limits; the algorithm can obtain the definition for linguistic values on a database attribute. A review of several categories of linguistically terms with vague meaning, their fuzzy modeling and specific operation are presented in [19].

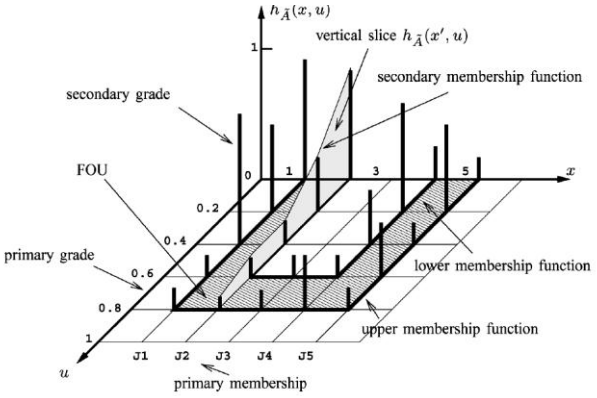


Figure 2. Three dimensional T2MF.

Fig. 2 illustrates all terminology of type-2 fuzzy set (T2FS). Two important concepts distinguish T2MF from T1MF [10], [11], [23]: secondary MF and FOU. The secondary MF is a vertical slice of T2MF, $h_{\tilde{A}}(x,u)$, at each value of $x = x$; i.e. the function $h_{\tilde{A}}(x,u) u \in J_x$. The amplitude of the secondary MF is called the secondary grade. The domain $J_x \in [0,1]$ of the secondary MF is called the primary membership of x, and u is the primary grade. The FOU is a bounded uncertain region in the primary memberships of a T2FS, and is the union of all primary memberships. An upper and lower MF are two T1MF that are bounds for FOU denoted by $h_{\bar{A}}(x)$ and $h_{\underline{A}}(x), \forall x \in X$ if $h_{\tilde{A}}(x,u) = 1 \; \forall x \in X, \forall u \in J_x \subseteq [0,1]$, the secondary MFs are interval sets, which reflect a uniform uncertainty at the primary memberships of *x*. Because all the secondary grades are unity, an IT2FS can be denoted by the interval of upper and lower MFs, i.e. $h_{\tilde{A}}(x) = [\underline{h}_{\tilde{A}}(x), \overline{h}_{\tilde{A}}(x)]$. The membership grade $h_{\tilde{A}}(x)$ of x in an T2FS is a T1FS in [0,1]. A useful theoretical overview about T2FS and fuzzy operator can be found in [23]. In this work, we utilize trapezoidal MF to describe all variables that are defined as:

$$\underline{\mu}_{\tilde{A}}(x) = \begin{cases} (x+a)/(a-c), & \text{if } -a \leq x \leq -c \\ 1, & \text{if } -c \leq x \leq c \\ (a-x)/(a-c), & \text{if } c \leq x \leq a \\ 0, & \text{otherwise} \end{cases} \qquad (2)$$

$$\overline{\mu}_{\tilde{A}}(x) = \begin{cases} (x+b)/(b-d), & \text{if } -b \leq x \leq -d \\ 1, & \text{if } -d \leq x \leq d \\ (b-x)/(b-d), & \text{if } d \leq x \leq b \\ 0, & \text{otherwise} \end{cases} \qquad (3)$$

where $0 \leq c \leq a$ *and* $0 \leq d \leq b$ The range of $[\underline{\mu}_{\tilde{A}}(x), \overline{\mu}_{\tilde{A}}(x)]$ specify the bounds of FOU according to the uncertainty of the model. A greater uncertainty will result in a larger spread of the FOU and increase the range

of $[\underline{\mu}_{\tilde{A}}(x), \overline{\mu}_{\tilde{A}}(x)]$. Herein, we use the following formulas [23].

$$\underline{\mu}_{\tilde{A}}(x) = \mu - k_u * \sigma, \overline{\mu}_{\tilde{A}}(x) = \mu + k_u * \sigma \quad k \in [0,+\infty) \tag{4}$$

k_u is uncertainty factor, the larger k_u the larger uncertainty, μ and σ are mean and variance of the attribute respectively. Here, we depend on category's cluster center to find parameters for building membership function. These parameters are obtained through fuzzy metaknowledge base where there is more than one type of curves and the selection of the membership function type depends on work requirements.

The computation complexity of general type-2 fuzzy logic systems (T2FLSs) is very high, which makes them very difficult to be deployed into practical applications; hence, only an interval T2FLS (a special case of general type-2 FLS) is today the most widely used T2FLS [24], [25]. Once all crisp attribute' values have been fuzzified into their respective linguistic values, the inference engine will access the fuzzy rule base of the fuzzy system to derive linguistic values for the intermediate as well as the output linguistic variables. The two main steps in the inference process are aggregation and composition. Aggregation is the process of computing the values of the IF (antecedent) part of the rules while composition is the process of computing the values of the THEN (conclusion) part of the rules.

We build collection of some IF–THEN rules which reflect the semantic of fuzzy query in the first stage and provide the behavior of process to achieve the result. The inference engine combines rules and gives a mapping from input type-2 fuzzy sets to output type-2 fuzzy sets using fuzzy rule base. It is necessary to compute the join $\cup$ (union) and the meet $\cap$ (intersections). In formal, if *A* and *B* are two type-2 fuzzy sets of the universe *u* and $\mu_A(x)$ is the grade of membership element *x* in the set *A*, then the fuzzy union $\cup$ and fuzzy $\cap$ intersection are defined as [3]:

$$QI = A \bigcup_{fuzzy} B = \{(x, \max\{\mu_A(x), \mu_B(x)\}) : x \in u\} \tag{5}$$

$$QI = A \bigcap_{fuzzy} B = \{(x, \min\{\mu_A(x), \mu_B(x)\}) : x \in u\} \tag{6}$$

Regarding new aggregation operator like "within" where the fuzzy model for the conjunction is defined as:

$$QI = A \bigwedge_{Within} B = \{(x, \min\{\mu_{A/B}(x), \mu_B(x)\}) : x \in u\} \tag{7}$$

in which $\mu_{A/B}(x)$ is the satisfaction degree of the first criterion relative to second one. Other types of fuzzy operators could be added in the future to catch other linguistic expressions.

After that, the Karnik-Mendel (KM) algorithm has been employed for centroid type-reduction. The type-reducer generates type-1 fuzzy set outputs which are then converted into a numeric output through running the defuzzifier. The defuzzified output of an interval singleton type-2 FLS is computed as [26]:

$$f(x) = \frac{y_l + y_r}{2} \tag{8}$$

$$y_l = \frac{\int_{-\infty}^{c_l} x\overline{\mu}(x)\,dx + \int_{c_l}^{\infty} x\underline{\mu}(x)\,dx}{\int_{-\infty}^{c_l} \overline{\mu}(x)\,dx + \int_{c_l}^{\infty} \underline{\mu}(x)\,dx} = \frac{\sum_{i=1}^{M} \mu_l^i c_l^i}{\sum_{i=1}^{M} \mu_l^i} \tag{9}$$

$$y_r = \frac{\int_{-\infty}^{c_r} x\underline{\mu}(x)\,dx + \int_{c_r}^{\infty} x\overline{\mu}(x)\,dx}{\int_{-\infty}^{c_r} \underline{\mu}(x)\,dx + \int_{c_r}^{\infty} \overline{\mu}(x)\,dx} = \frac{\sum_{i=1}^{M} \mu_r^i c_r^i}{\sum_{i=1}^{M} \mu_r^i} \tag{10}$$

where *M* presents the number of rules. A perfect FLS should have $f(x) = 0$; where 0 is the desired output but, generally, there exist errors between the desired output and actual output. We, therefore, need a design procedure for tuning the parameters of the FLS in order to minimize such errors. Herein, the satisfaction degree of each linguistic value is computed for each table row, and the min or max function is used to implement the fuzzy connection between them to compute the global criterion satisfaction degree of each tuple. Once this phase is finished, the DBMS manages the crisp data translated by the FSQL Server with a transparent way.

After the transformation of natural language query into SQL, the application program having first established a connection to the relational database, will now transfers the SQL query to the RDBMS. The interface here can be viewed as a reverse automated machine that displays the output of the search process. This makes the entire database search a cycle-like process. As a result of fuzzy classification by using conventional SQL queries and type-2 fuzzy interpreter, fuzzy classified data is presented. The crucial difference between fuzzy queries and exact queries is the number of records brought into the memory. A large number of tuples will be selected by fuzzy condition in comparison to the crisp one.

In a vague query, the selection criterion is no longer Boolean, so it can be more or less satisfied by the database tuples. Therefore, for each tuple, a satisfaction degree is estimated, which stands for the measure of its compatibility with the vague criterion. Including vague criteria in a database query may have the possibility to refine the results, assigning to each tuple the corresponding degree of criteria satisfaction. Note that, the membership function in a fuzzy set is not a matter of true or false but a matter of degree. This approach decreases the amount of transferred data across nets and calculation of QI_s is not significant burden for client computers.

IV. Experimental Results

In this section we present evaluation results showing the efficiency and effectiveness of the proposed system in answering imprecise queries. We used two real-life databases: (1) the car database Yahoo Autos (http://autos.yahoo.com) and (2) the Census Dataset from UCI Machine Learning Repository

(http://www.ics.uci.edu/learn/MLRepository.html), to evaluate our system. In conventional DBMS, the problem of query evaluation remains somewhat open since given a query; in general the optimal evaluation way cannot be reached. For fuzzy queries the process becomes more complex for two reasons: (i) the available access paths cannot be directly used, and (ii) a larger number of tuples is selected by fuzzy conditions with respect to Boolean ones. In the experiments, we select the probing queries from a set of spanning queries i.e. queries which together cover all the tuples stored in the database.

The evaluation focuses on two aspects: (1) the accuracy of retrieval records in terms of precision rate (the fraction of retrieved cases that are relevant from user queries); and (2) the accuracy of retrieval records in terms of recall rate (is the fraction of relevant cases that are retrieved). Both precision and recall are therefore based on an understanding and measure of relevance. During evaluation, databases with different sizes are generated for the series of experiments from the original database. The experiments were conducted by testing queries under different fuzzy threshold levels and different number of records including 500, 1000, 2000 and 5000 records. In general, the database querying access is usually limited because the difficulty to realize and express precise criteria to locate the information. The experimental results are obtained by averaging from 5 independent trials.

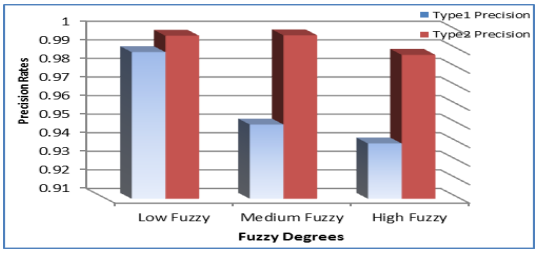


Figure. 3. Comparative results.

Figure. 4. Precision rates for different database sizes with high fuzzy level.

Figure. 5. Precision rates for different database sizes with medium fuzzy level.

The first set of experiments were conducted to compare between type-1 and type-2 fuzzy based vague querying for each fuzzy threshold level including low, medium and high. This procedure was repeated for database sizes and an average is calculated. The x-axis corresponds to the thresholds level and the y-axis corresponds to retrieval accuracy values. The values plotted are precision as shown in Fig. 3. The results reveal that the use of T2FL generates a further precision rate improvement of 1-2% for low level compare to T1FL. Furthermore, this improvement increases for medium and high fuzzy threshold. This improvement comes from the ability of T2FL to deal with higher level of uncertainty than regular fuzzy logic, which is heavily used in the previous works.

Validity of the results is confirmed by test group 2, aimed at testing retrieval accuracy when they are applied to the various database sizes with different fuzzy threshold. Again, precision rates are growing by using T2FL-based retrieval for both high and medium fuzzy levels in all database sizes as outlined in Fig. 4 and Fig. 5, while the actual precision rates for T2FL are convergent to the T1FL for low fuzzy level as illustrated in Fig. 6. These results support the claims made about the ability of the existing T2FL –based system to deal effectively with vague querying.

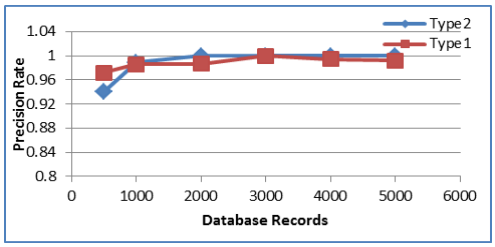


Figure. 6. Precision rates for different database sizes with low fuzzy level.

Fig. 7 summarizes recall rates for different database sizes with different fuzzy levels. As we can see, the proposed system performs well for retrieving correct answers for vague query depending on fuzzy threshold values.

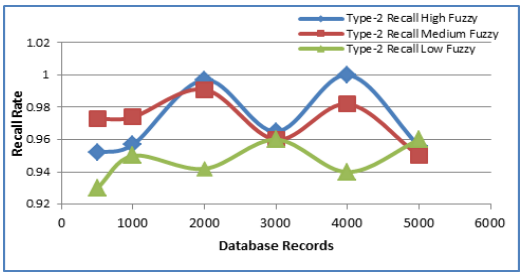


Figure. 7. Recall rates for different database sizes with different fuzzy level.

V. Conclusion

This paper examines situations when database querying process by the two valued realization of Boolean algebra is not adequate and offers solution based on type-2 fuzzy logic because the fuzzy logic is an approach for computing based on "degrees of truth" rather than the

usual "true or false" logic. The suggested FSQL language allows for dealing within a common framework with several aspects relevant to similarity query processing as well as with the inherent imprecision that characterizes data, user requests, and query results.

Here we only address the problematic of building a human-oriented interface capable of handling flexible queries, since our focus is on the representation of attributes by means of fuzzy sets to allow pseudo-natural language queries. The main advantages of our work are: (1) the existing implemented systems do not have to be modified; (2) the fuzzy attributes are built from the raw data; (3) the dialog with the system is done in a language very close to natural language; (4) the answers are given in a linguistic form, as well as a numeric form, which helps the user to better understand the results obtained; and (5) the interface can be used for other relational databases, after an initial preprocessing to define the fuzzy components with eventual slight changes in the grammar.

We believe that the research on the strict relationships between retrieval from database and fuzzy logic will eventually lead to the constructions of flexible reasoning systems, able to deal with problems of greater and greater complexity. As futures perspectives of this work, we mention the automatic mapping of existing relational DB to FRDB. This point is theoretically done but not implemented yet, so we think that it will contribute to make easier the use of the FRDB in real applications.

REFERENCES

[1] Z. M. Ma and L. Yan, "A literature overview of fuzzy database models," *Journal of Information Science and Engineering*, no. 24, pp. 189-202, 2008.

[2] M. Hudec, "An approach to fuzzy database querying, analysis and realization," *Computer Science and Information Systems*, vol. 6, no. 2, pp. 127-140, 2009.

[3] A. Garg and R. Rishi, "Querying capability enhancement in database using fuzzy logic," *Journal of Computer Science and Technology*, vol. 12, issue 6, pp. 38-46, 2012.

[4] A. Perović, A. Takači, and S. Skrbic, "Towards the formalization of fuzzy relational database queries," *Acta Polytechnica Hungarica*, vol. 6, no. 1, pp. 185-193, 2009.

[5] A. Raipurkar and G. Bamnote, "Fuzzy logic based query optimization in distributed database," *Int. Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, no. 2, pp. 422-426, 2013.

[6] D. Wei, L. Yi, and Z. Pei, "Application of fuzzy query based on relation database," in *Proc. Int. Conference on Intelligent Systems and Knowledge Engineering*, China, 2007, pp. 168-172.

[7] J. Galindo, "New characteristics in FSQL, a fuzzy SQL for fuzzy databases," *WSEAS Transactions on Information Science and Applications*, vol. 2, pp. 161-169, February 2005.

[8] M. Hudec, "Fuzzy improvement of the SQL," *Yugoslav Journal of Operations Research*, no. 2, pp. 239-251, 2011.

[9] J. Mishra, " Fuzzy query processing," *International Journal of Research and Reviews in Next Generation Networks*, vol. 1, no. 1, pp. 35-38, 2011.

[10] M. Owais, "Subjective decision making using type-2 fuzzy logic advisor," in *Proc. IEEE International Conference on Information and Communication Technologies*, Pakistan, 2009, pp. 127-133.

[11] C. Wagner and H. Hagras, "Toward general type-2 fuzzy logic systems based on zslices," *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 4, pp. 637-660, 2010.

[12] M. Hudec, Infostat, and S. Republic, "Fuzzy structured query language (SQL) for statistical databases," in *Proc. Joint Meeting on the Management of Statistical Information Systems*, no. 12, Luxembourg, 7-9 April 2008, pp.1-10.

[13] T. Mehta and P. Shah, "Handling fuzzy SQL on crisp databases using Lex-YACC," *International Journal of Computer Applications*, vol. 4, no. 2, pp. 5–8, July 2010.

[14] A. Branco, A. Evsukoff, and N. Ebecken," Generating fuzzy queries from weighted fuzzy classifier rules," in *Proc. ICDM Workshop on Computational Intelligence in Data Mining*, USA, 2005, pp. 21-28.

[15] S. M. Chen and H. R. Hsiao "A new approach for fuzzy query processing based on automatic clustering techniques," *Information and Management Sciences*, vol. 18, no. 3, pp. 223-240, 2007.

[16] L. Portinale and S. Montani, "A Fuzzy logic approach to case matching and retrieval suitable to SQL implementation," in *Proc. 20th IEEE International Conference on Tools with Artificial Intelligence*, USA, Nov. 2008, pp. 241 – 245.

[17] N. Nihalani, S. Silakari, and M. Motwani, "Natural language interface for database: A brief review," *International Journal of Computer Science Issues*, vol. 8, no. 2, pp. 600-608, March 2011.

[18] C. Turorie, S. Bumraru, and L. Dumitriu, "Relative aggregation operator in database fuzzy querying," Technical Report, University of Galati Fascicle, 2005.

[19] K. Singh, K. Prasad, M. Kumar, and A. K. Sharma, "Study of imperfect information representation and FSQL processing," *International Journal of Scientific and Enginnering Reasrch*, vol. 3, no. 5, pp. 1-7, May 2012,

[20] T. Madhulatha, "An overview of clustering methods," *Journal of Engineering*, vol. 2, no. 4, pp. 719-725, Apr. 2012.

[21] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881-892, July 2002.

[22] B. Manthey and H. Röglin, "Improved smoothed analysis of the k-means method," in *Proc. Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, USA, pp. 461-470, 2009.

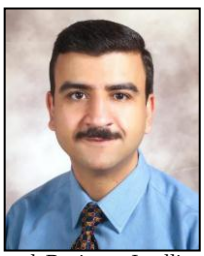
[23] O. Castillo and P. Melin, "Type-2 fuzzy logic: Theory and applications," *Springer-Verlag Berlin Heidelberg*, 2008.

[24] Q. Liang and J. Mendel, "Interval type-2 fuzzy logic systems: Theory and design," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 5, pp. 535-550, 2000.

[25] O. Castillo and P. Melin, "A review on the design and optimization of interval type-2 fuzzy controllers," *Applied Soft Computing*, vol. 12, pp. 1267–1278, 2012.

[26] F. Liu, "An efficient centroid type reduction strategy for general type-2 fuzzy logic system," *Information Sciences*, vol. 178, no. 9, pp. 2224–2236, 2008.

Saad M. Darwish received his Ph.D. degree from the Alexandria University, Egypt. His research and professional interests include image processing, optimization techniques, security technologies, and machine learning. He has published in journals and conferences and severed as TPC of many international conferences. Since Feb. 2012, he has been an Associate Professor in the Department of Information Technology, Institute of Graduate Studies and Research, Egypt.



Tamer F. Mabrouk received the B.Sc. in Computer Engineering from Arab Academy for Science & Technology and Maritime Transport (AASTMT) in 1997. He held the M.Sc. degree in Information Technology from the Institute of Graduate Studies and Research (IGSR), Department of Information Technology, Alexandria University in 2004. He received his Ph.D. degree from Alexandria University for a thesis in Multi-Agent System and Business Intelligence in 2009. His present research interests are currently focused on Database Management, Cloud Computing and Artificial Intelligence. He has published in journals and conferences and severed as TPC of many international conferences. Since Dec.2010, he has been an Assistant Professor in the Department of Computer engineering, Alexandria Higher Institute of Engineering & Technology, AIET.

Yasser F. Mokhtar received a bachelor's degree in accounting from the faculty of commerce, Alexandria University, Egypt in 1993. He held the diploma degree in information technology and accounting automated from faculty of commerce, Alexandria University, Department of Information Technology, University of Alexandria in 1995. He works as Computer science teacher in faculty of Commerce Alexandria University. And currently is the director of administration and technical support databases in the Egyptian Petrochemicals Co. He completed study of Pre-Masters Research from the Institute of Graduate Studies and Research (IGSR), Department of Information Technology, University of Alexandria and is preparing to discuss Master's Thesis.