A Method of Computing with Words to Answer Queries in Relational Database

Smita Rajpal
Associate Professor
Department of Computer Science &
Engineering, ITM University
Gurgaon, India

ABSTRACT
In this paper we are proposing a new method of intelligent search based on the concept of computing with words introduced by Prof. Zadeh[1] in 1996 to find the most suitable match for the predicates to answer any imprecise query made by the database users. The method is based on the theory of Computing with Words (CW). It is also to be mentioned that the proposed method could be easily incorporated in the existing commercial query languages of DBMS to serve the lay users better. So in this Paper Author is suggesting a new method called as α-CW-equality Search to answer the queries of Relational database based on ranks.

General Terms
Relational database Management System, Fuzzy Logic

Keywords
Probabilistic database, Computing with Words (CW), α-CW-equality Search, CW-Proximity Search, CW search

1. INTRODUCTION

Today Databases are Deterministic. An item belongs to the database is a probabilistic event, or a tuple is an answer to the query is a probabilistic event, and it can be extended to all data models. Probabilistic relational Data are defined in two ways,

- *Database is deterministic and Query answers are probabilistic* or
- Database *is probabilistic and Query answers are probabilistic*.

Probabilistic relational databases [2,3,4] have been studied from the late 80's until today. But today Application Need to manage imprecision's in data. Imprecision can be of many types: non-matching data values, imprecise queries, inconsistent data, misaligned schemas etc.

The quest to manage imprecision's is equal to major driving force in the database community. This is the Ultimate cause for many research areas like data mining, semi structured data, schema matching, nearest neighbor. Processing probabilistic data [5,6,7] is fundamentally more complex than other data models. Some previous approaches sidestepped complexity .Now our implementation includes Ranking query answers. Since our Database is deterministic, the query returns a ranked list of tuples but our user interested in top-k answers. Sometimes we get the empty answers for the user queries in the deterministic database. For e.g.

Try to buy a house in Seattle,

```
SELECT *
FROM Houses
WHERE bedrooms = 4
 AND style = 'craftsman'
 AND district = 'View Ridge'
 AND price < 400000
```

Here our database are unable to answer the queries because of imprecision in the query language.

Definition1.1 Ranking

Ranking is defined as computing a similarity score between a tuple and the query,

Consider the query

Q = SELECT *

FROM R

WHERE $A_1=v_1$ AND … AND $A_m=v_m$

Query is a vector: $Q = (v_1, \ldots, v_m)$

Tuple is a vector: $T = (u_1, \ldots, u_m)$

Consider the applications: personalized search engines, shopping agents, logical user profiles, and "soft catalogs"

To answer the queries related with the above application two approaches are given:

- Qualitative → Pare to semantics (deterministic)
- Quantitative→ alter the query ranking

Definition 1.2

An imprecise attribute value $t_m(a_i)$must be specified as a discrete probability distribution

Over Di, that is $t_m(a_i) = \{(z_j,P_j)\backslash z_j \in D_i \text{and } P_j \in [O, 1]\}$

With $\sum P_j = \alpha_{im}$, $0 <= \alpha_{im} <= 1$.

$(z_j,P_j) \in f_{vn}(a;)$

This definition covers both interpretations of null values as well as the usual interpretation of imprecise data: If $a_{im} = 1$, we certainly know that an attribute value exists, and with $a_{im} = 0$, we represent the fact that no value exists for this attribute. In the case of $0 < o_i, < 1$, o_i, gives the probability that an attribute value exists: For example, someone who is going to have a telephone soon gave us his number, but we are not sure if this number is valid already. With imprecise values specified this way, their probabilistic indexing weight can be derived easily.

Definition 1.3 Probabilistic Tuples

Let $R(A)$ be a relation scheme and let $t = (V_1; : : : ; V_n)$ be a tuple of cases of the relation scheme R. For each Vi, let V_1'' be the set of the $v_j' = (a_j ; l_j ; u_j ; p_j)$ such that $(a_j ; l_j ; u_j) \in Vi$, where p_j is the path associated with aj . A *probabilistic tuple* $t_0 = (v_1'; : : : ; v_n')$ is an element of the Cartesian product $V_1' \times ..... \times V_0'$.

By $A_i.l$, $A_i.u$ and $Ai.p$ we denote l_j , u_j and p_j associated with a generic value of A_i in a given probabilistic tuple, respectively.

Definition 1.4 Probabilistic Relations

A probabilistic relation r of the scheme R (A) is a finite set of probabilistic tuples of R (A). By domr (A_i) we will denote the set of all values of the attribute A_i in the relation r.

Definition 1.5 Probabilistic Databases

A probabilistic database [8,9,10] of the database scheme $R = \{R_1(A_1), : : : ;R_m(A_m)\}$ is a finite set of probabilistic relations $r = (r1_{,.....,} r_m)$, where each ri is a relation of the scheme $R_i(A_i)$.

In order to avoid probabilistic ambiguities we assume that in each initial relation there cannot be identical tuples.

So the failure of the RDBMS is due to the presence of imprecise constraints in the query predicate .Which can not be tackled due to the limitation of the grammar in standard query languages which work on crisp environment only. But these types of queries are very common in business world and in fact more frequent than grammatical-queries, because the users are not always expected to have knowledge of DBMS and the query languages.

Consequently, there is a genuine necessity for the different large size organizations, specially for the industries, companies having world wide business, to develop such a system which should be able to answer the users queries posed in natural language, irrespective of the QLs and their grammar, without giving much botheration to the users. Most of these type of queries are not crisp in nature, and involve predicates with fuzzy (or rather vague) data, fuzzy/vague hedges (with concentration or dilation). Thus, these types of queries are not strictly confined within the domains always. The corresponding predicates are not hard as in crisp predicates. Some predicates are soft because of vague/fuzzy nature and thus to answer a query a hard match is not always found from the databases by search, although the query is nice and very real, and should not be ignored or replaced according to the business policy of the industry. To deal with uncertainties in searching match for such queries computing with Words will be the appropriate tool.

In this paper we propose a new type of searching techniques called as CW search which is a combination of α-CW-equality search and CW proximity search by using Computing with Words theory to meet the predicates posed in natural language in order to answer imprecise queries of the users. Thus it is a kind of an intelligent search for match in order to answer imprecise queries of the lay users. We call this method by CW search that is a combination of α-CW-equality search and CW proximity search.

Our method, being an intelligent soft-computing method, will support the users to make and find the answers to their queries without iteratively refining them by trial and error which is really boring and sometimes it seriously effects the interest (mission and vision) of the organization, be it an industry, or a company or a hospital or a private academic institution etc. to list a few only out of many. Very often the innocent (having a lack of DBMS knowledge) users go on refining their queries in order to get an answer. The users are from different corner of the academic world or business world or any busy world. For databases to support imprecise queries, our intelligent system will produce answers that closely match the queries constraints, if does not exactly. This important issue of closeness cannot be addressed with the crisp mathematics. That is why we have used the Computing with Words tools.

2. THEORY OF COMPUTING WITH WORDS (CW)

In its traditional sense, computing involves (for the most part) manipulation of numbers and symbols. We generally employ mostly words in computing and reasoning, but words have fuzzy denotations. The same applies to the role played by words in CW. Although the foundations of computing with words were laid some time ago, its evolution into a distinct methodology in its own right reflects many advances in our understanding of fuzzy logic and soft computing-advances which took place within the past few years. By Prof zadeh, a key aspect of CW is that it involves a fusion of natural languages and computation with fuzzy variables. It is this fusion that is likely to result in an evolution of CW into a basic methodology in its own right, with wide-ranging ramifications and applications. The point of departure in CW is the concept of a granule. In essence, a granule is a fuzzy set of points having the form of a clump of elements drawn together by similarity. **A** word w is a label of a granule g and, conversely, g is the denotation of *w.* A word may be atomic (as in *young)* or composite (as in *not **very** young).* Unless stated to the contrary, a word will be assumed to be composite. The denotation of a word may be a higher order predicate, as in Neutrosophic/Vague grammar [9,10]. In CW, a granule *g, which is the denotation of a word w,* is viewed as a fuzzy constraint on a variable. A pivotal role in CW is played by fuzzy constraint propagation from premises to conclusions. It should be noted that as a basic technique, constraint propagation plays important roles in many methodologies, especially in mathematical programming, constraint programming, and logic programming.

Definition 2.1:

Computing with words and perceptions, or CWP for short is a mode of computing in which the objects of computation are words, propositions and perceptions described in a natural language.

Definition 2.2:

Let **Ω** be a set of (non-ambiguous) words that are used in a given context. The elements of **Ω** are considered to be pairs (representation, meaning). Representations may be taken from different syntactic domains, but meanings, from a single semantic domain. Moreover, let **Γ** be a finite set of functions **{γ 1, γ 2, …, γ k},** with **γ i : Ω n _ Ω**

The functions **{γ 1, γ 2, …, γ k}** must be *designed* in such a way, that applied to words of **Ω** will produce reasonable words of **Ω** for the given context.

Definition 2.3:

In this paper when speaking of computing with words, the *objects* of computation are specified in definition 1. The *Agents* of computation are functions taken from **Γ** , as stated in definition 2.

Examples: Let **Ω** be a subset of English.

If **γ 1** is an appropriate "association" function. Then

γ 1 (teacher, student) = school

If **γ 2** is a synonym function. Then

γ 2 (happy) = celebrate

If **γ 3** is an antonym function. Then

γ 3 (day) = night

These examples show that since the functions in **Γ** represent some "linguistic transformation", their interpretation should be taken take in account in the line of language precisiation.

3. A NOTE ON INTERVAL MATHEMATICS

Dealing with the Computing with Words theory, the crisp theory of interval mathematics is sometimes useful. In this section, we recollect some basic notions of interval mathematics. For our purpose in this paper, we need to consider intervals of non-negative real numbers only.

Let $I_1 = [a,b]$ and $I_2 = [c,d]$ be two intervals of non-negative real numbers. A point valued non-negative real number r also can be viewed, for the sake of arithmetic, as an interval [r,r].

3.1 Some Algebraic Operations

(i) Interval Addition: $I1 + I2 = [a+c, b+d]$

(ii) Interval Subtraction: $I1 - I2 = [a-c, b-d]$

(iii) Interval Multiplication: $I1 * I2 = [ac, bd]$

(iv) Interval Division: $I1 \div I_2 = [a/d, b/c]$, when $c, d \neq 0$.

(v) Scalar Multiplication: $k. I_1 = [ka, kb]$.

3.2 Ranking Of Intervals

Intervals are not ordered. Owing to this major weakness, there is no universal method of ranking a finite (or infinite) number of intervals. But in real life problems dealing with intervals, we need to have some tactic to rank them in order to arrive at some conclusion. We will now present a method of ranking of intervals, which we shall use in our work here in subsequent sections. We consider a decision maker (or any intelligent agent like a company manager, a factory supervisor, an intelligent robot, an intelligent network, etc) who makes a pre-choice of a decision parameter $\beta \in [0,1]$. The intervals are to be ranked once the decision-parameter β is fixed. But ranking may differ if the pre-choice β is renewed.

Definition 3.1 *β-value of an interval*

Let $J = [a,b]$ be an interval. The β-value of the interval J is a non-negative real number J_β, given by $J_\beta = (1-\beta).a + \beta.b$.

Clearly, $0 \leq J_\beta \leq 1$, and for $\beta = 0$ $J_\beta = a$ which signifies that the decision-maker is pessimistic, and also for $\beta = 1$ $J_\beta = b$ which signifies that the decision-maker is optimistic. For $\beta = .5$ it is the arithmetic-mean to be chosen usually for a moderate decision.

Comparison of two or more intervals we will do here on the basis of β-values of them. If the value of β is renewed, the comparison-results may change. The following definition will make it clear. Now Author is proposing α-Neutrosophic-equality search.

4. α-CW EQUALITY SEARCH

Consider the STUDENTS database given below. Consider a normal type of query like

PROJECT (STUDENT_NAME)

WHERE AGE = "approximately 20".

The standard SQL is unable to provide any answer to this query, as the search for an exact match for the predicate will fail. The value "approximately 20" is not a precise data. Any data of type "approximately x", "little more than x", "slightly less than x", much greater than x" etc. are not precise or crisp. Denote any one of them, say "approximately x" by the notation I(x). Clearly for every member $a \in$ dom(AGE), there is a membership value $t_{I(x)}(a)$ proposing the degree of equality of this crisp number a with the quantity "approximately x", and a non-membership value $f_{I(x)}(a)$ proposing the degree of non-equality . Thus, in Computing with words of Prof. Zadeh, every element of dom(AGE) satisfies the predicate AGE = "approximately 20" upto certain extent and does not satisfy too, upto certain extent. But we will restrict ourselves to those members of dom(AGE)

which are α-CW-equal, the concept of which we will define below. Any imprecise predicate of type AGE = "approximately 20", or of type AGE = "young" (where the attribute value "young" is not a member of the dom(AGE)), is to be called by CW-predicate, and a query involving CW-predicate is called to be a CW-query.

Definition 4.1

Consider a choice-parameter $\alpha \in [0,1]$. A member of a of dom (AGE) is said to be α-CW-equal to the quantity "approximate x" if $a \in I_\alpha(x)$, where $I_\alpha(x)$ is the α-cut of the CW number I(x). The degree or amount of this equality is measured by the interval $m_{I(x)}(a) = [t_{I(x)}(a), 1-f_{I(x)}(a)]$. Denote the collection of all such α-CW-equal members from dom (AGE) by the notation $AGE_\alpha(x)$, which is a subset of dom(AGE). If AGE $_\alpha(x)$ is not a null-set or singleton, then the members can be ranked by ranking their corresponding degrees of equality.

Definition 4.2

Consider a choice value $\beta \in [0,1]$. At β level of choice, for every element a of AGE_α (x), the truth-value *t* (p_1, p2) of the matching of the predicate p_1: given by AGE = "approximately x" with the predicate p_2: AGE = a is equal to the β-value of the interval $m_{I(x)}(a)$.

5. CW- PROXIMITY SEARCH

The notion of α-CW-equality search as explained above is appropriate while there is an CW-predicate in the query involving Vague Numbers. But there could be a variety of vague predicates existing in a CW query, many of them may involve CW hedges (including concentration/dilation) like "good", "very good", "excellent", "too much tall", "young", "not old", etc. In this section we present another type of search for finding out a suitable match to answer imprecise queries. In this search we will use the theory of CW-proximity relation. We know that a CW-proximity relation on a universe U is a CW relation on U, which is both CW-reflexive and CW-symmetric.

Consider the STUDENTS database as described and a query like

PROJECT (STUDENT_NAME)

WHERE EYE-COLOR = "dark-brown".

The value/data "dark-brown" is not in the set dom (EYE-COLOR). Therefore a crisp search will fail to answer this. The objective of this research work is to overcome this type of drawbacks of the classical SQL. For this we notice that there may be one or more members of the set dom (EYE-COLOR), which may closely match the eye-color of "brown" or "dark-brown".

Consider a new universe given by

$$W = \text{dom (EYE-COLOR)} \cup \{\text{dark-brown}\}.$$

Propose a CW-proximity relation R over W. Choose a decision-parameter $\alpha \in [0,1]$. We propose that search is to be made for the match $e \in$ dom(EYE-COLOR) such that

$$t_R(\text{dark-brown}, e) \geq \alpha.$$

(It may be mentioned here that the condition $t_R(\text{dark-brown},e) \geq \alpha$ does also imply the condition $f_R(\text{dark-brown},e) \leq 1-\alpha$ **).**

We say that e is a close match with "dark-brown" with the degree or amount of closeness being the interval $m_{\text{dark-brown}}(e)$ given by

$$m_{\text{dark-brown}}(e)=[t_R(\text{dark-brown},e),\ 1-f_R(\text{dark-brown},e)\ ].$$

At β level of choice, the truth-value $t(p_1,p_2)$ of the matching of the predicate p_1: given by EYE-COLOR = "dark-brown" with the predicate p_2: AGE = e is equal to the β-value of the interval $m_{\text{dark-brown}}(e)$.

6. CW-SEARCH

In this section we will now present the most generalized method of search called by CW-search. The CW-search of matching is actually a combined concept of α-CW-equality search, CW-proximity search and crisp search.

For example, consider a query like

PROJECT (STUDENT_NAME)

WHERE (SEX = "M", EYE-COLOR = "dark-brown", AGE= "approximately 20") .

This is a CW-query.

To answer such a query, matching is to be searched for the three predicates p_1, p_2 and p_3 given by

(i) p_1: SEX = "M",

(ii) p_2: EYE-COLOR = "dark-brown" and

(iii) p_3 : AGE = "approximately 20",

where p_1 is crisp and p_2, p_3 are CW(imprecise).

Clearly, to answer this query the proposed CW search method is to be applied, because in addition to crisp search, both of α-CW-equality search and CW-proximity search will be used to answer this query. The truth-value of the matching of the conjunction p of p1, p_2 and p_3 will be the product of the individual truth-values, (where it is needless to mention that for crisp match the truth-value will be exactly 1). There could be a multiple number of answers to this query, and the system will display all the results ordered or ranked according to the truth-values of p.

7. CONCLUSION

In this paper, we have introduced a new method to answer imprecise queries of the lay users from the databases (details of the databases may not be known to the lay (users). We have adopted computing with words tools to solve the problem of searching an exact match or a close match (if an exact match is not available) of the predicates so that we will be able to get the answer of 'evidence for you'(i.e. exact/ truth match) and 'evidence against you'(i.e false match) and the 'undecidability'(i.e. indeterminacy) This is a complete new Method of Answering Queries based on the concept of Computing with Words.

REFERENCES

[1] L.A. Zadeh, from computing with numbers to computing with words. From manipulation of measurements to manipulation of perceptions. *IEEE Trans. Circuits and Systems* 45 (1): 105-119, 1999

[2] D. Barbara, H. Garcia-Molina and D. Porter (1992). "The management of probabilistic data". IEEE Trans. Knwl. Data Eng. 4,(5), 487-502.

[3] V. Biazzo and A. Gilio (1999), "A generalization of the fundamental theorem of de Finetti for imprecise conditional probability assessments". International Journal of Approximate Reasoning .

[4] Biazzo V., Gilio A., and Sanfilippo G., Efficient Coherence Checking and Propagation of Imprecise Probability Assessments (working paper), 1999.

[5] R. Cavallo and M. Pittarelli (1987). "The theory of probabilistic database". In Proceedings of the 13th VLBDB Conference. brighton, Eng., 71-78.

[6] D. Dey and S. Sarkar (1996). "A probabilistic relational model". ACM Trans. Database Systems,21, (3), 394-405.

[7]. C. R´e, N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In Proceedings of ICDE, 2007.

[8]. C. R´e and D. Suciu. Efficient evaluation of having queries on a probabilistic database. Technical Report TR2007-06-01, University of Washington, Seattle, Washington, June 2007

[9] Smita Rajpal, M.N.Doja and Ranjit Biswas, A Method of Neutrosophic Logic to Answer Queries in Relational Database, JCS, Vol.4(4): 309-314 ,USA, 2008.

[10] Smita Rajpal, M.N.Doja and Ranjit Biswas, A Method of Imprecise Query Solving, IJCSNS, Vol. 8 No. 6 pp. 133-139,South Korea,June2008.