Intelligent Queries based on Fuzzy Set Theory and SQL

Tien-Chin Wang1 Hsien-Da Lee1,2 Chun-Ming Chen3

1Department of Information Management, I-Shou University, Kaohsiung, Taiwan (tcwang@isu.edu.tw)
2Department of Information Management, Fortune Institute of Technology, Kaohsiung, Taiwan (leesd@center.fotech.edu.tw)
3Department of Marketing and Distribution Management, Fortune Institute of Technology, Kaohsiung, Taiwan (bredchen@ms2.hinet.net)

Abstract

As professor Zadeh first introduced fuzzy theory, fuzzy theory has been applied to many fields. In many application fields, fuzzy query is capable of processing imprecise and ambiguity data.

The development of fuzzy query can be used on traditional databases or fuzzy databases. Many researchers develop many techniques to represent data schema. However, it increases difficulties to handle a fuzzy query. In this research, we present a fuzzy language architecture based on SQL and Fuzzy sets. This architecture can effectively reduced complexity of processing data and maintaining a database. It also can balance the variety of data and system performance. By applying fuzzy logic and SQL to the evaluation of records, we significantly improve the robustness of database query operations.

Keywords: Fuzzy Set Theory, Relational Database, Structured Query Language

1. Introduction

From professor Zadeh [1] first introduced fuzzy theory to now, many researchers have devoted their effort on this field. Among fuzzy-related researches, most of current fuzzy query researches are based on relational databases [2]. Structure Query Language (SQL) is the standard language to query databases. SQL gives business analysts and engineers a powerful method for seeking out and exploiting pattern in data. Combined the essence of fuzzy logic and SQL, fuzzy SQL is getting more attention by academic researchers.

Recently, fuzzy database researches focus on the fuzzy languages architectures, integrating various data types, and tuning weights of parameters [5].

2. Fuzzy Set Theory

Fuzzy logic allows us to model imprecise or vague data. The use of fuzzy logic also allows us to model vague knowledge.

Fuzzy sets are defined on a non-fuzzy universe of discourse, which is an ordinary set. A fuzzy set $F$ of a universe of discourse $U$ is characterized by a membership function $\mu_F(\cdot)$ which assigns to every element $x \in U$, a membership degree $\mu_F(x) \in [0,1]$. An element $x \in U$ is said to be in a fuzzy logic $F$ if and only if $\mu_F(x) > 0$ and to be a full member if and only if $\mu_F(x) = 1$ [6].

Fig. 1 shows the two membership functions of “medium young” and “about 35”.

Fig. 1 Membership functions of “medium young” and “about 35”

3. Database Query Languages

In the 1970s the concepts of relational database emerged as the standard way of representing large quantities of complex information. The basic building
block of a relational database is the table. Tables are connected through a process is called join. In order to provide high performance joins, primary and secondary keys are defined. Also, the database relationships are expressed among tables. Structure Query Language (SQL) developed by IBM is the standard tool to manage relational databases by means of text-based commands. SQL is a relational calculus approach to database access and information management: it selects and organizes sets of records through a high-level language that tells the systems what records to select. However, SQL has its principal analytical shortcomings: it lacks discrimination among database records. SQL requires crisp data in order to get results. In real world, much data exist in unclear and ambiguous status that SQL fails to process. Fuzzy Query Language (FQL) is an extension of SQL to deal with such situations. FQL introduces a new way of retrieving information from a relational database. Fuzzy query is able to extract information from databases based on concepts rather than numbers and text strings. This capability gives FQL more flexibility to process real-world data.

3.1. Fuzzy Query

The Fuzzy Query Language (FQL) is an extension of Structured Query Language (SQL). According to Bosc & Pivert [3] definition, a fuzzy query statement is defined as following:

```
SELECT <attribute(s) list>
FROM <table(s) list>
WHERE <fuzzy condition(s) list>
WEIGHT <fuzzy weight(s) list>
THRESHOLD <retrieval threshold value>
```

WHERE clause contains user-defined conditions. Users may use fuzzy term or fuzzy proposition to build two or more query conditions. The multi-relation blocks of WHERE clause may involve both Boolean and fuzzy predicates combined by several kinds of connectors, thereby achieving a large number of semantic effects [4]. V.Tahani suggested that extension of SQL base block in order to support the imprecise comparison between an attribute value and a constant or between two attribute values such as Join. These elementary predicates can be combined using the connectors AND and OR working as intersection and union of fuzzy sets.

WEIGHT clause let users to decide which attributes may have more weights to determine a query result. We can use fuzzy term to describe a weight or directly assign a value which may vary from 0 to 1.

THRESHOLD clause is used to decide criterion value that meet minimum similarity degree of a fuzzy query. The alphabet $\alpha$ denotes the threshold value which ranges from 0 to 1.

4. Experimental Results

In this section, we apply Fuzzy Query Languages to process imprecise data. Fig.2 shows the flows of processing system. Through a pre-processing layer, Transformation Layer, we can transform imprecise data based on fuzzy set theory stated in Section 2. Then, evaluate each data by utilizing Possibilities Theory. Finally, output query result which $\alpha$ value meets threshold value.

4.1. Flows of a Fuzzy Query

The process steps are illustrated as follows:

Step 1: read different types of data. Data may be either crisp or imprecise.

Step 2: According to WHERE clause, we select fuzzy data from fuzzy databases. Then use pre-defined threshold $\alpha$ to classify data into some groups.

Step 3: Check query statement if WEIGHT clause exists. If it does, then we have to take membership degree into consideration. If not, then membership degree doesn’t need to be considered. Based on weighted value, find out the correspondent membership degree from Fig.3. Then fuzzy weights become crisp.

Step 4: If user directly assigns a weight value, then use that specific value to calculate membership degree. If not, we apply average value as weight.
Step 5: Transfer the data according to membership degree to calculate possibility value (PV) to evaluate similarity.

Step 6: If possibility value is larger or equal to threshold value $\alpha$, then output the result.

4.2. Processing Weights

As the matter of processing weights, we follow Chen & Chen’s [5] methodology that fuzzy term or precise value $\omega \in [0,1]$. The method allows to determine weights of attributes in the WHERE clause.

Chen & Chen developed a fuzzy term/weights translation table. The method builds a table (Table 1) that each linguistic triangular membership function has its correspondent weighted value according to importance of the linguistic term.

<table>
<thead>
<tr>
<th>Table 1 Fuzzy Term / Fuzzy Number Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linguistic</td>
</tr>
<tr>
<td>very very unimportant</td>
</tr>
<tr>
<td>very unimportant</td>
</tr>
<tr>
<td>unimportant</td>
</tr>
<tr>
<td>less unimportant</td>
</tr>
<tr>
<td>medium</td>
</tr>
<tr>
<td>less important</td>
</tr>
<tr>
<td>important</td>
</tr>
<tr>
<td>very important</td>
</tr>
<tr>
<td>very very important</td>
</tr>
</tbody>
</table>

Our research adapts Chen & Chen’s method to match linguistic terms in Table 2 to its triangular fuzzy number. Then use defuzzified method [7] to calculate. The crisp weight value is calculated as following equation:

$$\hat{\omega} = \frac{t_1 + 2t_2 + t_3}{4}$$  \hspace{1cm} (1)

$\hat{\omega}$ is the crisp weight value, $t_3$ is the value of membership degree at center position, $t_1$ and $t_3$ are the position of right and left, respectively.

4.3. Experiment

To explain how fuzzy data are stored and how to apply Fuzzy Query Language to retrieve data, Let us consider a salary table of some employees in Table 2.

<table>
<thead>
<tr>
<th>Table 2 Employees Salary in linguistic terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>David</td>
</tr>
<tr>
<td>Johnson</td>
</tr>
<tr>
<td>Kevin</td>
</tr>
</tbody>
</table>

The fuzzy logic of “high income” can be defined as High income= \{0/1300,0.5/1500, 0.7/1700,0.8/1800, 1/2000\}

Our research adapts Chen & Chen’s method to match linguistic terms in Table 2 to its triangular fuzzy number. Then use defuzzified method [7] to calculate. The crisp weight value is calculated as following equation:

$$\hat{\omega} = \frac{t_1 + 2t_2 + t_3}{4}$$  \hspace{1cm} (1)

$\hat{\omega}$ is the crisp weight value, $t_3$ is the value of membership degree at center position, $t_1$ and $t_3$ are the position of right and left, respectively.

4.3. Experiment

To explain how fuzzy data are stored and how to apply Fuzzy Query Language to retrieve data, Let us consider a salary table of some employees in Table 2.

<table>
<thead>
<tr>
<th>Table 2 Employees Salary in linguistic terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>David</td>
</tr>
<tr>
<td>Johnson</td>
</tr>
<tr>
<td>Kevin</td>
</tr>
</tbody>
</table>

The fuzzy logic of “high income” can be defined as High income= \{0/1300,0.5/1500, 0.7/1700,0.8/1800, 1/2000\}

Operations in Fuzzy logic include operations such as intersect, union, compliment, equality and level sets. Raw data in Table 2 can be converted into fuzzy logic as in Table 3.

<table>
<thead>
<tr>
<th>Table 3 Employees Salary (Fuzzy Data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>David</td>
</tr>
<tr>
<td>Johnson</td>
</tr>
<tr>
<td>Kevin</td>
</tr>
</tbody>
</table>

If we want to find “select the employees’ name and income of all employees whose income is very high and weight (income)= very important and threshold (income)=0.4". The fuzzy query statement can be written as:

SELECT *
FROM Employees
WHERE income is very high
WEIGHT income is very important
THRESHOLD income= 0.4
Querying the fuzzy databases involves intensive computation and defuzzification of fuzzy data.

Step 1: read data from fuzzy databases.

Step 2: According to Shenoi & Melton [8], we cluster “income” into several subgroups. Income={\{0-1,300\},\{1,301-1,500\},\{1,501-1,700\},\{1,701-1,800\},\{1,801-2,000\},\{2,001-5,000\}}.

Step 3: Since the statement “income is very important” in WHERE clause, we transfer the fuzzy term “very important” into its correspondent triangular membership function \{0.7, 0.8, 0.9\}. Then we use method in Equation 1 to get crisp weight. The calculated crisp value is 0.8.

Step 4: The term “very high” is computed as “very high”\(^3\) = \{(0/1300, 0.25/1500, 0.49/1700, 0.64/1900, 1/2000)\}

Step 5: The query is defuzzified by computing the \(\alpha\) value of each employee.

Step 6: Each row in the table is compared with threshold value to obtain the answer in Table 4.

<table>
<thead>
<tr>
<th>Name</th>
<th>Income</th>
</tr>
</thead>
<tbody>
<tr>
<td>David</td>
<td>{1/1500, 1/1700}</td>
</tr>
<tr>
<td>Johnson</td>
<td>{0.5/1500,0.7/1700,0.8/1800,1/2000}</td>
</tr>
</tbody>
</table>

4.4. Discussion

Integrating the concepts of fuzzy set theory and SQL, FQL is able to process imprecise and ambiguous data. FQL could extract more potential or suitable records compared to its SQL counterpart. Through the concept of membership degree, a query threshold is a form of alpha cut threshold applied to the aggregate truth membership. The fuzzy query approach could balance the process by including records that to some degree meet the requirement of a query. Thus, using a fuzzy query approach instead of a conventional SQL query, we may found more suitable candidate records that are suitable to the query operations.

5. Conclusion

FQL is an extension of standard SQL and adapts the concept of fuzzy logic and possibilities theory. The development of fuzzy query languages continues from traditional databases to fuzzy databases. Many representation methodologies have been proposed. In this paper, we propose a fuzzy query language which may be applied to query imprecise data. We have shown how the values of one imprecise statement could be represented by an appropriate distribution of possibility. We also present a transformation layer to handle transformation of different types of fuzzy data. It also can balance the variety of data and system performance. FQL makes it possible that the databases query language becomes closer to oral languages.

6. References