한국산학기술학회논문지
Vol. 6, No. 3, pp. 265-269, 2005

Fuzzy Structured Query Language (FSQL) for Relational Database Systems

Eun-Young Jung[1], Soon Cheol Park[2] and Sang Bum Lee[3*]

관계형 데이터베이스 시스템을 위한 퍼지 질의어 (FSQL)

정은영[1] · 박순철[2] · 이상범[3*]

Abstract A fuzzy query language, called FSQL, in the relational databases is introduced in this paper. In general, database systems have query systems which are able to retrieve and manipulate precise data. However, such queries are hard to operate on the real world applications since their queries are often imprecise or incomplete. Recently, considerable attention has been given to research dealing with vagueness of the query in relational database systems. In this paper we have suggested an effective method of accepting vagueness of the query in data processing. The syntax of FSQL is formally defined with EBNF, and an interpreter of FSQL has been implemented as a prototype.

요 약 본 논문에서는 관계형 데이터베이스에서 운영될 수 있는 퍼지질의어인 FSQL를 소개하였다. 일반적으로 데이터베이스 시스템은 질의시스템을 갖고 있는데 이는 정확한 데이터를 추출하고 처리할 수가 있다. 하지만 실세계에서는 정확한 데이터를 처리하는 경우보다는 부정확하거나 불명확한 질의를 처리하는 경우가 많다. 최근에 관계형 데이터베이스 시스템에서의 애매모호한 질의를 처리할 수 있는 연구가 관심을 끌고 있다. 본 고에서는 데이터처리에 있어 애매모호한 질의를 처리할 수 있는 효과적인 방법을 제안하였다. 퍼지질의어인 FSQL의 구문은 EBNF로 정의되었고 FQSL를 SQL로 전환하여 처리할 수 있는 인터프리터를 시제품으로 개발하였다.

Key Words : Fuzzy query, Relational Databases, Fuzzy sets

1 Introduction

The relational data model has been extensively studied and widely used because it works very simply compared to other types of data models, and it also has a reliable mathematical basis. This model usually can manipulate only well-defined and precise queries. However, real world applications often require imprecise or incomplete queries which cannot be manipulated by the relational data systems. For example, a query such as "*retrieve a set of single women who are young and tall*", is hardly processed in traditional relational database systems. In order to deal with this kind of imprecise query, fuzzy database systems have been widely studied [1-12]. However, there are only few references that implement fuzzy databases yet.

We propose, in this paper, a query language called FSQL (Fuzzy Structured Query Language) that can deal with vague queries in relational database systems. The syntax of FSQL is formally defined by EBNF and an FSQL interpreter is implemented to prove its power and expressiveness. An input query statement of FSQL is converted to SQL form by this interpreter and processing is performed by referring metadata in a relational database system.

This paper is organized as follows. Section 2 presents the definition of FSQL and the EBNF form of FSQL. Section 3 introduces the implementation of an FSQL interpreter and shows the results. In sec-

[1]LG Electronics Ltd. Co. Researcher
[2]Dept. of Information and Communication Eng. Chonbuk National University
[3]Dept. of Electronics and Computer Science Dankook University
*Corresponding author: Sang Bum Lee (sblee@dankook.ac.kr)

tion 4, conclusions are discussed.

2. FSQL overview

2.1 FSQL Process

The major features of FSQL are a superset of SQL, supporting vagueness query and running in a normal relational database. <Fig. 1> shows the process flow of FSQL operation suggested in this paper. The basic processing is as follows: when an FSQL source file inputs, then SQL code generator converts that the source code to a normal SQL code. To convert the SQL code, metadata is used for defining the fuzziness of data. Some new variables may be generated for easy SQL processing. The converted SQL statement is processed in the relational database system to produce the results. The final results will be filtered for providing more proper results.

2.2 The EBNF form of FSQL

The following EBNF format of FSQL syntax shows only extended part of SQL BNF form. The basic structure of FSQL inherits the structures of the SQL form since FSQL is the superset language of SQL. The bold words are newly defined words. It is an extension of SQL's BNF form.

① 0, 1, and the decimal fractions between 0 and 1.
<fnumeric literal> :: = 0 | 0[<period>[<unsigned integer>]]|1

② ~operator
<SQL special character> ::= <SQL special character> | **<Tilde>**
<Tilde> :: = ~

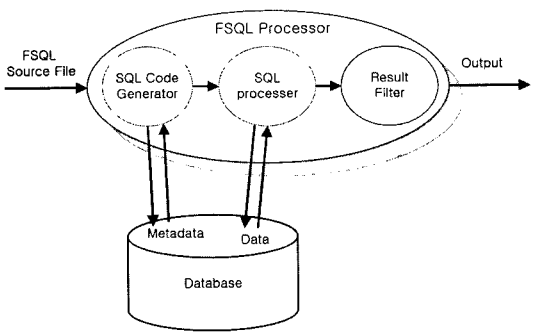


Fig. 1. Structure of FSQL Process.

③ reserved word (bound, very, little, above, below)
<reserved word> :: = <reserved word> | bound | very | little | above | below

④ bound
<query specification> ::= <query specification> | <query specification> **<bound clause>**
<where clause> ::= <where clause> | <where clause> **<bound clause>**
<bound clause> ::= bound **<fnumeric literal>**

2.3 Basic Structure of FSQL

The syntactical form of FSQL suggested in this paper is shown in <Fig. 2>. The basic structure of FSQL expression also consists of three clauses as in SQL, but FSQL has more reserved words to deal with fuzziness such as **'bound'**, **'F'**, **'Wc'** and **'op'**. These words are explained as following.

• **bound** The word 'bound' limits the lowest value from its membership function or results. It picks out exceptional documents that have a low correspondence to user need, or controls the size of

Select $A_1, A_2, \ldots, A_n$ ***bound*** K_1
From $T_1, T_2, \ldots, T_m$
Where $[x_1 \Theta F W_1 y_1$ ***bound*** $K_2]$ ***op***
$[x_2 \Theta F W_2 y_2$ ***bound*** $K_3]$ ***op***
. . .
$[x_n \Theta F W_n y_n$ ***bound*** $K_{n+1}]$

A_i : Attribute_name
T_j : Table_Name
K_k : Bound Value
x_a : Attribute_name for Searching
y_b : Value
W_c: Reserved Word(null or very, little, above, below)
Θ : Comparision Operator (<, ≤, =, ≠, ≥, >)
F : ~ Operator
op : Connection Operator (and, or, not)

Fig. 2. FSQL Form.

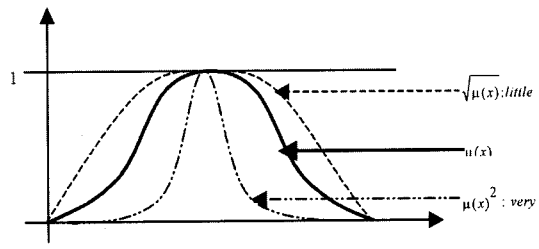


Fig. 3. Functions of 'very' and 'little'.

the output by eliminating documents below bound values.

• **null or very, little, above, below** Occasionally during query processing, we need to control a degree, such as 'find a very young woman' or 'find a young woman'. In such cases, we can change the degree of its membership function by using the reserved word (Wc), 'very' or 'little'. In this paper, we define that 'very' has a squared function, and 'small' has a square root function as shown in <Fig. 3>.

There are two more reserved words, 'above' and 'below'. From the specified base point in their membership function, these bring the right side to '1' in case of 'above', and the left side to '1' in case of 'below.' For example, the query of 'find a person taller than 150 (or above 150 cm)' brings the membership function to '1' in its right side from a base line, 150(cm), since it coincides with the user's need.

• **~Operator(F)** The operator~means 'fuzzy', and exists in front of a vague target. In this system, no other comparative operator can explain fuzziness. That is, the expression, 'where height = ~180' means '180 and more', or '180 and less'.

• **The Coupled Operator (op)** The FSQL operators 'Fuzzy and' and 'Fuzzy or' are logical operators similar to those in general SQL language however, the FSQL operators are somewhat different from those in SQL. Let's suppose that some person has a membership value of '0.9' as the condition of a tall person, and a membership value of '0.5' as the condition of a young person. This person must have one membership value for the case of each query '*a young and tall man*' and the other query 'a young or tall man'. Many research projects about this are currently in progress. In this paper, we use 'Fuzzy and', and 'Fuzzy or' operators [9-12] that solve 'Single Operand Dependency Problem' and 'Negative Compensation Problem'. <Table 1> shows the 'Fuzzy and' and 'Fuzzy or' operation method.

Table 1. Fuzzy and, Fuzzy or operations

Fuzzy and	$\gamma MIN(x,y) + \frac{(1-\gamma)(x+y)}{2},\ 0 \le \gamma \le 1$
Fuzzy or	$\gamma MAX(x,y) + \frac{(1-\gamma)(x+y)}{2},\ 0 \le \gamma \le 1$

***Select* Name, Address *bound* 0.8**
***From* Background**
***Where* age=~25 *and* height = *above* 150**

Fig. 4. An example of the FSQL form.

<Fig. 4> shows the example of the 'Fuzzy and' operation. The FSQL statement in <Fig. 4> says: 'selects people who are about 25-years old and taller than 150 with above the membership value of 0.8.'

3. Implementation of the FSQL Processor

The FSQL processor consists of three components. The first is 'fsqlparser' to analyze a FSQL sentence. Second is 'fsql2sql' to convert the FSQL form to the SQL form, referring metadata. Last is the 'fsqlQuery_application' to get the results produced by the converted SQL. <Fig. 5> shows an example of the FSQL statement and the converted SQL statement. It assumes that there are persons who are from 0 to 100 years old. In this example,

***select* manager**
***from* employee**
***where* age = ~old *bound* 0.8**
→
***select* manager**
***from* employee**
***where* 45 <= age < 145 *and* age_old_fvalue > 0.8**

Fig. 5. Example of FSQL processing.

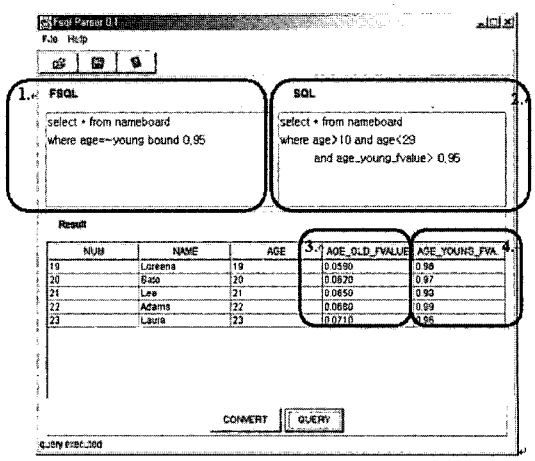


Fig. 6. FSQL Process window.

age_old_fvalue is a newly generated attribute for the fuzzy variable of age.

<Fig. 6> shows the user interface of the FSQL query system. An FSQL statement is input through the 'FSQL' window. If the CONVERT button is pushed, the FSQL statement is converted into the standard SQL statement. Once it presses the QUERY button, the result of output data is shown in the lower window. This FSQL processor is implemented by JavaCC, a Java parser generator built by Sriram Sankar, Rob Duncan, and Sreenivasa Viswanadha [13, 14].

4 Conclusions

Database systems that can manipulate vague information are needed. We have proposed a query language called FSQL that can deal with vagueness in the relational database system. It offers an easy user interface, and it doesn't need crisp knowledge of the keys to retrieve data stored in a database.

In our model, FSQL has a special function. First, 'bound' controls the amount of results that it limits to a membership function. Second, 'reserved word (very, little, above, below) controls the shape of the membership function. Third, '~operator' means 'fuzzy'. Finally, 'and' and 'or' operator computes a union value of membership values.

To develop this model for a wider FSQL system, it is important to process a lot of metadata. And it is not very easy to the personal user unless it doesn't unite with speech recognition. The Fuzzy database model in this paper is a vague model, the database of which stores only crisp values. Since it is exactly same as those of a normal relational database system, our FSQL can be applied to the normal relational database systems.

References

[1] L. A. Zadeh, "Fuzzy Sets", Information and Control, 1965, pp.338-353.

[2] Deyi Li and Dongbo Liu, "A Fuzzy Prolog Database System," Research Studies Press LTD. 1989.

[3] M. H. Wong and K.S. Leung, "A Fuzzy Database Query Language', Information Systems, 1990, pp. 583-590.

[4] Y. Takahashi, "Fuzzy Database Query Languages and Their Relational Completeness Theorem", IEEE Trans. Knowledge and Data Engineering, 1993, pp.122-125.

[5] Takahiko Nomura and Toshiyuki Odaka, "Generating Ambiguous Attributes for Fuzzy Queries", 0-7803-0236-2192 IEEE, 1992, pp.753-760.

[6] Janusz Kacprzyk, Bill P. Buckles, Frderick E. Petry, "Fuzzy Information and Database Systems", Elsevier Science Publishers B.V., 1990, pp.133-135.

[7] M. Umano, "Retrieval from Fuzzy Database by Fuzzy Relational Algebra", IFAC Fuzzy Information, 1983, pp.1-6.

[8] A. Motro, "VAGUE:A User Interface to Relational Databases That Permit Vague Queries", ACM Trans. On Office Information Systems, 1988, pp.187-214.

[9] Doheon Lee, Hyung Lee-Kwang and Myoung Ho Kim, "A Study on the Fuzzy Selective Relational Algebra(FSRA)", the 2nd International Conference on Fuzzy Logic & Neural Networks, 1992, pp.353-356.

[10] S. C. Park, C. S. Kim and D. S. Kim, "Fuzzy Logic and its Applications to Engineering Information Sciences and Intelligent Systems", KLUWER ACADEMIC PUBLICSMERS, 1995, pp407-415.

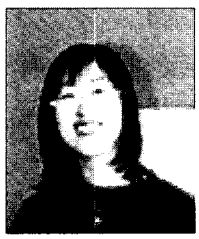
[11] A. Motro, "Accommodating Imprecision in Database Systems: Issues and Solutions", Data Engineering, 1990, pp29-34.

[12] M. Unamo and Y. Ezawa, "Implementation of SQL-type Data Manipulation Language for Fuzzy Relational Databases", IFSA Brussels, 1988.

[13] http://pepe.uta.edu/sohn/JavaCC/index.html

[14] http://www.metamata.com/javacc/docs

Eun-Young Jung **[Regular member]**



- Feb. 1999 : Chonbuk University, B.S. in Information and Telecomm. Engineering
- Feb. 2001 : Chonbuk University, M.S. in Information and Telecomm. Engineering
- March 2001-Present : LG Electronics Co. Researcher

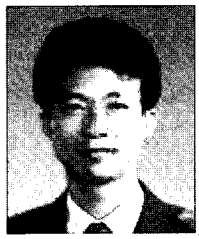
<Research Interests>
Database, Network Programming

Soon Cheol Park **[Regular member]**

- Feb. 1979 : Inhwa University, B.S.
- Dec. 1991 : Louisiana State University, Ph.D. in Computer Science
- Dec. 1991-Sept. 1993 : ETRI Senior Researcher
- Sept. 1993-Present : Professor, Dept of Electronical and Information Engineering, Chonbuk Univerisy

<Research Interests>
Databases, Information Retrieval, Data Mining

Sang-Bum Lee **[Regular member]**



- Feb. 1983 : HanYang University, B.S. in Mechanical Eng.
- Dec. 1989 : Louisiana State University, M.S. in Computer Science
- Aug. 1992 : Louisiana State University, Ph.D. in Computer Science
- Sept. 1992-Sept. 1993 : ETRI Senior Research
- Oct. 1993-Present : Associate Professor, Dept. of Computer Science, Dankook University

<Research Interests>
Software Enigneering, Information Retrieval