A Knowledge-Based Fuzzy Query and Results Ranking Approach for Relational Databases

Xiangfu MENG¹,†, Zongmin MA², Xuguang ZHU¹

¹ College of Electronic and Information Engineering, Liaoning Technical University, Huludao 125105, China
² College of Information Science and Engineering, Northeastern University, Shenyang 110004, China

Abstract

Users often have vague or imprecise ideas when searching the databases and thus may not know how to precisely formulate queries that lead to satisfactory answers. Based on the fuzzy set theory, this paper proposes a fuzzy query approach which can support the expression of fuzzy queries and translate the fuzzy query into the precise query in order to provide relevant answers to the users. The fuzzy queries mean that the query condition consists of complex fuzzy terms as the operands and complex fuzzy relations as the operators in a fuzzy query. With the knowledge base related to the application domain and the different thresholds that the user chooses for the fuzzy query, the user fuzzy queries can be translated into precise queries for classical relational databases. For the the fuzzy query results, they are finally ranked according to their satisfaction degree to the original fuzzy query and the user preferences. The efficiency and effectiveness of our approach is also demonstrated by experimental results.

Keywords: Relational Database; Fuzzy Query; Knowledge Base; User Preferences; Ranking

1. Introduction

Nowadays, database query processing models have always assumed that the user knows what he/she want and they supported only a Boolean matching model. However, the user’s query intentions are usually vague or imprecise and thus the user may like to issue the fuzzy queries which consist of fuzzy terms or fuzzy relations for possibly retrieving. Moreover, the user also has insufficient knowledge about the database structure and contents, thus frequently obtaining empty answers and having to reformulate the query several times. Therefore, providing some flexibility to the query processing model can help users to improve their interaction with the databases.

Related work. Since Zadeh introduced the fuzzy sets theory [18] fuzzy values have been employed to model and handle imprecise information in databases. Several researches have been proposed to handle the fuzzy queries over the classical relational databases [1], [3], [5], [6], [14] and [16]. The success of these approaches depends on the utilization of the fuzzy sets theory. Tahani [16] firstly advocated the use of fuzzy sets for querying conventional databases, where imprecise conditions inside queries were seen as fuzzy sets. SQLf language proposed by Bosc and Pivert [1], which is a fuzzy extension to SQL, represents a synthesis of the characteristics and functionalities suggested in other previous proposals of flexible query

† Corresponding author.

Email addresses: marxi@126.com (Xiangfu MENG), zongmin_ma@yahoo.com (Zongmin MA).
in classical databases, such as Tahani [16], Bosc et al. [3], Wong and Leung [17], and Nakajiam et al. [14]. Also there are some extensions to SQLf [4], [9] and [10]. The translation rules of fuzzy queries were presented in [8] and [12]. But these methods are lack of domain knowledge and user preferences for constructing the membership functions of fuzzy sets, and they also do not discuss the solutions for fuzzy query results ranking. Recently, literature [6] proposed a fuzzy query and relaxation approach for relational databases, where the relaxation mechanism rests on a transformation that consists in applying a tolerance relation to fuzzy predicates contained in the query. However, it should be pointed out that the relaxation mechanism proposed in [6] assumes that all fuzzy criterions involved in fuzzy query are of the same importance for the user while in fact the user often has different preferences on different criterions.

2. Fuzzy Query

2.1. Definition of Fuzzy Query

Definition 1. A fuzzy query contains one or several fuzzy basic conditions, with the form \( Q = C_1 \land C_2, \ldots, \land C_k \), where each fuzzy basic condition can be represented by means of a fuzzy set and modeled gradual property.

For example, consider a realtor database consisting of a single table HouseDB with attributes (Price, City, Bedrooms, Location, Schooldistrict, SqFt, Buildyear…). Each tuple represents a house for sale in the US. Consider a potential house buyer searching for houses in this database. She may like a comparatively new house that is priced not exceed $300000 and the living space is between 1200 and 1500 square feet. According to her needs, we can formulate the following fuzzy query:

\[ Q : - \text{HouseDB (Price at most 300000 } \land \text{SqFt between 1200 and 1500} \land \text{Buildyear = more or less recent)} \]

Obviously, there are two fuzzy basic conditions, “Price at most 300000” and “Buildyear = more or less recent” in this query, where the fuzzy term is “more or less recent” and the fuzzy relation is “at most”. However, most database systems are still based on standard SQL for querying nowadays and allow the user neither to use vague or imprecise terms nor to express his/her needs in the query. Moreover, if the query is not very selective, too many tuples may be in the answer after a fuzzy query. Thus, it is necessary to translate the fuzzy query into the precise query which DBMS can execute and to rank the answer items according to their satisfaction degree to the user’s needs and preferences.

2.2. Formulation of Fuzzy Queries

The traditional precise query of relational databases is composed of the basic condition \( A \theta Y \), where \( A \) is an attribute, \( \theta \) is the regular operator such as =, >, <, \geq, \leq, \neq, (not) between etc, and \( Y \) is the operand. Similarly, the fuzzy basic condition in a fuzzy query can be described by fuzzy terms or fuzzy relations. Combining some fuzzy relations with precise values, or combining fuzzy terms and various regular relations, the fuzzy basic conditions are formed.

2.2.1. Fuzzy Relations as Operators.

There are three types of fuzzy relations, which are “(not) close to”, “(not) at least” and “(not) at most”, have been identified in [12]. Using these fuzzy relations and precise values, the fuzzy basic condition with fuzzy operators, which has the form \( A \theta Y \), is formed. Here, \( \theta \) is a fuzzy relation, \( Y \) is a precise value, and \( \theta Y \) is a fuzzy number with membership function.
2.2.2. Fuzzy Terms as Operands

Three kinds of fuzzy terms have been identified: simple (atomic) fuzzy term, modified (composite) fuzzy term, and compound fuzzy term [3].

Simple (atomic) fuzzy term [8]. A simple fuzzy term such as “young” or “tall” is defined by a fuzzy number with membership function.

Modified (composite) fuzzy term. A modified fuzzy term such as “very young” or “more or less tall” is described by a fuzzy number with membership function. Note that its membership function is not defined but computed through the membership function of the corresponding simple fuzzy term.

Compound fuzzy term. A compound fuzzy term such as “young ∪ very young” is represented by simple fuzzy terms or modified fuzzy terms connected by union, intersection or complementation connectors.

Using the fuzzy terms above and traditional regular operators, the fuzzy query condition with fuzzy operands, which has the form \( A \theta ^\gamma \), is formed [12]. Here \( ^\gamma \) is a fuzzy terms given above as the operands.

2.2.3. Fuzzy Interval as Operands

For a range query condition, “\( A \) (not) between \( Y_1 \) and \( Y_2 \)”, where \( A \) is the attribute, \([Y_1, Y_2]\) is a numerical interval. In fact, although the user specifies a precise range query, they will also be satisfied by the numerical values nearby the numerical interval of the query. So the precise numerical interval should be expanded. In this paper, the interval \([Y_1, Y_2]\), which is expanded from \([Y_1, Y_2]\) is called as the fuzzy interval.

3. Translation and Relaxation of Fuzzy Query

The idea of translating and relaxing the fuzzy query is to convert each fuzzy basic condition of the fuzzy query into an enlarged precise one by using the knowledge base, membership functions and \(\alpha\)-cut operation of fuzzy set, and then these conditions are combined to formulate an enlarged precise query.

3.1. The Knowledge Base

The knowledge base (KB) includes the values for building membership functions and other information such as the relaxation directions according to the attributes semantics. The KB is maintained by domain experts [13]. The information stored in KB is the priori domain knowledge, so we call it knowledge base. The KB in this paper is consisted of the following four relational tables.

- **AttDescription** (Attribute, Table, Llimit, Rlimit, LImportance)
- **LImportance** (LImportance, Attribute, Table, Memdegree)
- **MemFunction** (FuncName, Attribute, Table, Predicate, para1, para2, para3, para4, Satisfy)
- **AttRelaxation** (Attribute, Table, Predicate, Directionrel, Lreldeg, Rreldeg, Lsatisfy, Rsatisfy)

The table **AttDescription** describes the relaxable attributes, i.e., the attributes specified by fuzzy basic conditions of fuzzy query. The attribute **Limit** (resp. **RLimit**) is the limit value of the left relaxation (resp. right). The attribute **LImportance** is a linguistic label reflecting the importance of the associated attribute in comparison with the other attributes of the database. This label also expresses the importance of fuzzy basic condition involving the associated attribute and it can be specified by users according to their preferences.

In the table **LImportance**, the attribute **Memdegree** refers to the membership degree of the linguistic label **LImportance** in the fuzzy set modeling the importance of the attribute.
The table MemFunction provides the values for constructing membership functions. The attribute FuncName denotes the name of the membership function representing the associated fuzzy predicate Predicate. The membership functions representing the fuzzy predicates have been pre-defined in the system without materializing. The attributes para1, para2, para3 and para4 represent the values of corresponding parameters of the membership function, respectively. The attribute Predicate represents the fuzzy relations or fuzzy terms associated to the attributes. Note that, if the Predicate is “between”, the core of which is \([para1, para2]\). Satisf represents the threshold of membership function which indicates that the query condition must be satisfied with minimum degree threshold in \([0, 1]\).

The table AttRelaxation provides the information for relaxing the fuzzy basic conditions, which includes, the relaxable attribute and fuzzy basic condition, the relaxation direction (left and/or right), the relaxation degree and the satisfaction type (decreasing or non-decreasing). For a fuzzy basic condition “Price close to 300000”, if the relaxation direction is “left”, it means that for the relaxation of this condition, it only considers values less than 300,000.

3.2. The Algorithm of Fuzzy Query Translation

According to the formulation of fuzzy query and the knowledge base, the fuzzy query translation step can be described by algorithm 1.

\textbf{Algorithm 1.} Fuzzy query translation algorithm

\textbf{Input:} fuzzy query \(Q\), knowledge base \(KB\), threshold \(\lambda\)

\textbf{Output:} the enlarged precise query \(Q'\)

\textbf{Method:}

1. \(Q' \leftarrow \emptyset\)
2. for each fuzzy basic condition \(C_i\) of \(Q\) do
3. \(w \leftarrow \) the importance of \(C_i\)
4. \(ER \leftarrow \text{MembershipFunctionExpandingRange}(\text{relaxation direction, satisfaction type})\)
5. if (the relaxation degree \(\neq\) null) then
6. \(MF \leftarrow \text{MembershipFunctionConstruct1 (FuncName, parameter values, relaxdegree, w)}\)
7. else
8. if (\(\exists\) a predicate \(P\) \& (\(\text{core}(P) = \) set of values satisfying \(C_i\) \& \(\text{ExpandingRange}(P) = ER\) then
9. \(MF \leftarrow \text{MembershipFunctionConstruct2 (FuncName, P, parameter values, w)}\)
10. else
11. if (\(\exists\) limit values associated to the attribute involved in \(C_i\) then
12. \(MF \leftarrow \text{MembershipFunctionConstruct3 (FuncName, limit values, w)}\)
13. \(C_i' \leftarrow \text{MembershipFunctionCompute (MF, \lambda)}\)
14. \(Q' \leftarrow Q' \cup C_i'\)
15. return \(Q'\)

By using the algorithm 1, a fuzzy query can be translated into a enlarged precise one and the relevant answer items can be retrieved. For a large size database, however, a fuzzy query may result in too many answers and these fuzzy query results should be ranked according to their satisfaction degree.
4. Fuzzy Query Results Ranking

In this section we first propose the membership degree measuring method and then describe the relevance degree measuring method. These two methods are finally used to rank the fuzzy query results.

4.1. Membership Degree Measuring

Let $Q$ be a fuzzy query and $t$ be an answer tuple for $Q$. Also let the set of attributes $X = \{X_1, \ldots, X_i\} \subseteq A$ be the set of attributes specified by the fuzzy basic conditions in $Q$. Then, the membership degree of the answer tuple $t$ to the fuzzy query $Q$ can be defined as

$$ D(t, Q) = \sum_{i=1}^{k} W(X_i) \times \mu_C(v_i) $$

Here $k$ is the number of fuzzy basic conditions in $Q$, $X_i$ is the attribute specified by the fuzzy basic condition $C_i$ in $Q$, $W(X_i)$ denotes the membership degree of the label associated to $C_i$ in the fuzzy set modeling the importance of the attribute $X_i$, and $\mu_C(v_i)$ is the membership degree of the value $v_i$ of attribute $X_i$ to the fuzzy set representing $C_i$. However, it is insufficient by only using the tuple’s membership degree to rank the fuzzy query results, this is because several tuples may tie for the same membership degree in the answer and thus get ordered arbitrarily. Therefore, it is necessary to look the unspecified attributes in the query and to estimate the tuple’s relevance degree to the user preferences for ranking.

4.2. Relevance Degree Measuring

In this section, we propose how to speculate the user preferences on values of unspecified attributes and use these implicit preferences to rank the answer tuples sharing the same membership degree.

4.2.1. IDF Weight

The well-known IDF method has been used extensively in IR to suggest that commonly occurring words convey less information about user’s needs than rarely occurring words, and thus should be weighted less. $IDF(w)$ of a word $w$ is defined as $\log(n/F(w))$ where $n$ is the number of documents, and $F(w)$ is the number of document in which $w$ appears. If the database only had categorical attributes, each tuple can be treated as a small document. Thus, we can mimic this method for our problem.

**IDF weight for categorical attribute values.** For a categorical attribute value $v_i$ of attribute $A_i$, $IDF(v_i)$ is defined as $\log(n/F(v_i))$ which represents the importance of attribute value $v_i$ in the database, where $n$ is the number of tuples in the database and $F(v_i)$ is the frequency of tuples in the database where $A_i = v_i$.

**IDF weight for numerical attribute values.** The adaptation of IDF for estimating a numerical value $v_i$ of attribute $A_i$ has been proposed in [1]. We briefly describe it as follows. Let $T = \{v_1, v_2, \ldots, v_n\}$ be the values of numerical attribute $A$ that occur in the database. For any numerical value $v_i$, $IDF(v)$ be defined as shown in Equation (2) (where $h$ is the bandwidth parameter and is illustrated in the following).

$$ IDF(v) = \log \left( \frac{n}{\sum_{i=1}^{n} e^{\frac{1}{2} \left( \frac{v - v_i}{h} \right)^2}} \right) $$

(2)
Intuitively, the denominator in Equation (2) represents a numerical extension of the concept of “frequency” of $v$, i.e. the sum of “contributions” to $v$ from every the other point $v_i$ in the database. These contributions are modeled as (scaled) Gaussian distributions, so that the further $v$ is from $v_i$, the smaller is the contribution from $v_i$. A popular estimate for the bandwidth is $h = 1.06\sigma n^{-1/5}$, where $\sigma$ is the standard deviation of $\{v_1, v_2, \ldots, v_n\}$.

### 4.2.2. IDF-based Relevance Degree Estimation

Consider a fuzzy query $Q$ specified a set of attribute values, we assume that $X$ is the set of attributes specified in the query, and $Y$ is the remaining set of unspecified attributes. Based on this assumption, for every value $v_j$ in the domain of unspecified attribute $Y_j$, the relevance score of value $v_j$ can be defined by

$$\text{RelScore}(v_j) = \text{IDF}(v_j)$$  \hspace{1cm} (3)

In Equation (3), if $Y_j$ is a categorical attribute, $\text{IDF}(v_j)$ is defined as $\log(N/F(v_j))$, where $N$ is the number of tuples in the database, $F(v_j)$ is the frequency of occurrence of value $v_j$ of attribute $Y_j$ in the database; if $Y_j$ is a numerical attribute, $\text{IDF}(v_j)$ is defined as Equation (2), where $n$ is the number of tuples in the database.

Based on the relevance score of each unspecified attribute value, we can define the tuple’s relevance degree to the user preferences. Let $Q$ be a fuzzy query and $t$ be an answer tuple for $Q$. Also let the set of attributes $X = \{X_1, \ldots, X_s\} \subseteq A$ is the set of specified attributes, while the set $Y = A - X$ is the set of unspecified attributes. Then, the relevance degree of the answer tuple $t$ to the user preferences can be defined as:

$$S(t, Q) = \sum_{j=1}^{m} W(Y_j) \times \text{RelScore}(v_j)$$ \hspace{1cm} (4)

Where, $W(Y_j)$ is the membership degree of the linguistic label associated to attribute $Y_j$ in the fuzzy set modeling the importance of the attribute $Y_j$, $\text{RelScore}(v_j)$ is the relevance score of value $v_j$ of unspecified attribute $Y_j$. The relevance coefficient of unspecified attribute value is defined as the product of relevance score and the corresponding attribute weight, and the relevance degree of each answer tuple to the user preferences is simply the sum of corresponding relevance coefficient over all unspecified attributes.

## 5. Experiments

### 5.1. Experimental Setup

We used Microsoft SQL Server 2005 RDBMS on a P4 3.2-GHz PC with 1 GB of RAM for our experiments. We implemented all algorithms in C# and SQL language, and connected to the RDBMS through ADO. For our evaluation, we set up two databases. The first database is a real estate database HouseDB (City, Price, Location, Bedrooms, Bathrooms, SqFt, Schooldistrict, View, Neighborhood, BuildYear) containing 237,620 tuples extracted from Yahoo!RealEstate and the total data size is 53.75MB. The second database is a used car database CarDB (Make, Model, Year, Color, Engine, Price, Mileage) containing 1000,000 tuples extracted from Yahoo!Autos and the total data size is 123.05MB.

### 5.2. Efficiency of Fuzzy Query Translation and Relaxation

In order to evaluate the efficiency of our algorithm—FQRR (fuzzy query & results ranking), we used the standards recall and precision evaluations. Especially, we computed the precision and recall at various cut-off points, where the precision is determined at various recall levels.
It is trivial that high recall is obtained at the cost of lower precision. Likewise, a high precision can be attained at the cost of recall. The problem is to find a good balance between recall and precision [7]. Fig. 1 shows the precision evaluation when taking 10 recall levels from 0 to 100%; that is, given ranked results of the fuzzy query, the subject checks whether the first ranked tuple is truly relevant, if so, it is associated 100% precision level, and 0 otherwise, then checks the second ranked tuple, and so forth. The numerical values pointed out in Fig. 1 were obtained using an average of 10 queries over HouseDB (resp. CarDB), these test queries were provided by 10 subjects.

![Fig.1 Precision and Recall Curves using FQRR Algorithm over HouseDB and CarDB](image)

5.3. Performance Report

Fig. 2 shows the query execution time of the queries over HouseDB (resp. CarDB) as a function of the number of tuples in the query result, using both FQRR query model and Logical query model. It can be seen that the execution time of FQRR grows almost linearly with the number of tuples in the query result.

![Fig.2 Execution Times for Different Numbers of Query Results for HouseDB and CarDB](image)

From Fig. 2 we can also see that the computation time of FQRR, which is almost three times that of the classical Logical query model. But it should be noticed that the large increase of such computational time is not entirely related to the fuzzy translation and ranking algorithm themselves, but partly to annexed implementations such as interface. However, the computation time is usually not a primary issue as the relevance is deemed much more important from the user’s perspective viewpoint.
6. Conclusions

This paper presented FQRR—a knowledge-based approach for answering fuzzy queries over relational databases. The main contributions are summarized as: (1) proposed a fuzzy query translation method, which considers both the query criterion importance and domain knowledge when translating the fuzzy query into the precise query, and (2) proposes a fuzzy query results ranking method, which considers both the membership degree to the fuzzy query and the relevance degree to the user preferences of answers. How to make the knowledge base running more efficient without experts’ assistants is our further work.

Acknowledgments.

This work is supported by the National Natural Science Foundation of China (60873010).

References