ABSTRACT
Analysis of new trends in databases development shows that promising solution in databases gathering a huge quantities of information is approximate retrieval. This work presents the fuzzy queries processing with a special attention to context-dependent and multidimensional membership functions.
In the first part, the analysis of occurrence of fuzzy values in SQL queries directed to the databases is shown. Next, the process of interpretation of filtering conditions in SQL queries is discussed and condition on compatibility degree is introduced to SQL queries notation.
The main part of the paper describes application of context-dependent and multidimensional membership functions in fuzzy SQL queries as two different approaches, and a whole process of their usage interpretation. All issues considered in this work are illustrated by appropriate examples.

KEY WORDS
Fuzzy Databases, Fuzzy SQL Queries, Membership Functions

1. Introduction

In outward things, there are many domains where informatics systems collecting huge quantities of various data are applied. Efficient retrieval of needed information can be a critical problem in that systems. In traditional databases, classical retrieval algorithms assure searching only these objects which attributes are precisely consistent with conditions in a given query. Objects which are not precisely consistent with the given conditions are not located. Analysis of new trends in database development shows that promising solution in database gathering a huge quantities of information is approximate retrieval [1, 2]. The main assumption of approximate retrieval methods is that the answer on the given query may contain set of the objects from database which are consistent with criteria defined in the query with the given degree. Queries of this type require defining characteristic function, which determines in what degree searched object is consistent with criteria defined in the query and threshold value, which allows qualify objects that ought to occur in the resultant answer. For the fuzzy queries a membership function is the characteristic function. Its value defines matching degree of searched object in database with criteria specified in the query [1, 2].
This work presents the fuzzy queries processing that point to the context-dependent and multidimensional membership functions. In the first part, the analysis of occurrence of fuzzy values in SQL queries directed to the databases is shown. Next, the process of interpretation of filtering conditions in SQL queries is discussed and condition on compatibility degree is introduced to SQL queries notation. The main part of the paper describes application of context-dependent and multidimensional membership functions in fuzzy SQL queries as two different approaches, and a whole process of their usage interpretation. All issues considered in this work are illustrated by appropriate examples. All of the presented queries will be directed to the database called DEPARTMENTS which structure is presented in Fig.1.

![Figure 1. The structure of the Departments database.](image)

2. Fuzzy Values in SQL Queries

Queries, the user directs to the database are often first formulated in the natural language. In these queries imprecise, fuzzy terms may appear [3]. The conditions in such
formulated queries are usually transposed on filtering conditions and therefore they should be analysed very precisely. Considering the places where filtering conditions in the `SELECT` statement of SQL language may occur, these conditions can be divided on [4, 5]:

- occurring in `WHERE` phrase (respecting particular rows),
- occurring in `HAVING` phrase (respecting selected groups of rows).

Except for places mentioned above, the fuzzy values can appear in the `SELECT`, `GROUP BY`, `ORDER BY` phrases of `SELECT` statement.

### 2.1 Interpretation of Filtering Conditions in SQL Queries

Let’s consider the simple query formulated in SQL language:

```sql
SELECT A1, ..., Ak
FROM T
WHERE C;
```

where `C` is a filtering condition, which has the following form: `X is A` (X – column of T table in database, A – given value). The retrieval process of rows fulfilling the condition given in the query can be treated as a conclusion process, where for each row in the T table (on the base of comparison of value of the X column with the given A value) it is gathered if the row can be accepted to the resultant set or not.

Taking into consideration the nature of the X and the A values, the conditions may be divided on [2, 6]:

1. classical, where both X and A assume crisp (precise) values,
2. X assumes crisp values whilst A assumes fuzzy values represented by membership functions
3. X assumes fuzzy values represented by membership functions whilst A assumes crisp values,
4. both X and A assume fuzzy values represented by membership functions.

The analysis of each of the particular cases is proceeded in a different manner. However, the result has the same general form: for each row (determining X value), value of a compatibility degree of the X value with the A value should be computed (a compatibility degree which the X value fulfils the given condition with). The different manners of the compatibility degree calculation are illustrated in Fig. 2 and in Fig. 3, and precisely are described in [2, 6, 7].

![Figure 2](image-url)  
Figure 2. Intersection between the A about 20 fuzzy value with the X=17 crisp value [8, 9].

![Figure 3](image-url)  
Figure 3. Intersection between the A about 20 fuzzy value defined by triangular membership function and the X fuzzy value defined by trapezoidal membership function [8, 9].

The mentioned cases refer to the simple condition comparing one value with the second one. The `WHERE` clause in classical `SELECT` statement in the SQL language can contain many comparison conditions joined by conjunction AND and disjunction OR operators [4, 5]. The columns can be preceded by negation operator (NOT). Generally, we can say, that in the `WHERE` clause containing fuzzy conditions, values of fuzzy conjunction or disjunction can be calculated on the different ways. The simplest method is to apply the Zadeh norm [8, 9]: t-norm for conjunction (minimal value of compared compatibility degrees) and s-norm for disjunction (maximal value of compared compatibility degrees). Another t-norm and s-norm operators may be applied, too [8, 9]. In situation, when the column is preceded by negation operator (NOT), the value of the compatibility degree is named e.g. by subtraction initial value of the compatibility degree from 1 value. The operators for complex fuzzy conditions in `SELECT` statement are presented in table 1 [9, 10].

<table>
<thead>
<tr>
<th>WHERE phrase</th>
<th>AND</th>
<th>t-norm</th>
<th>OR</th>
<th>s-norm</th>
<th>NOT</th>
<th>1-µ or another</th>
</tr>
</thead>
</table>

To describe notation of a fuzzy conditions in formal way, we define the grammar fragment in the BNF. It has the following form [2, 6]:

```plaintext
...<complex_fuzzy_cond> ::= <simple_fuzzy_cond> | <complex_fuzzy_cond><oper><simple_fuzzy_cond>
<simple_fuzzy_cond> ::= <argument><relational_operator><argument> | <argument><fuzzy_operator><argument> | <argument> ::= <crisp_value> | <fuzzy_value> | <relational_operator> ::= < > | <= | => | <> | in | more_or_less
<fuzzy_operator> ::= AND | OR | AND NOT | OR NOT ...
```

This example illustrates, how the given query in the natural language with imprecise term is transposed into SQL statement extended with simple fuzzy condition:

“Find names of employees, who are about 50 years old.”

```sql
SELECT LastName
FROM employees
WHERE age is about 50;
```
2.2 Introduction Condition on Compatibility Degree to SQL Queries Notation

The introduction of condition on compatibility degree to SQL query notation affects the process of query interpretation. In the process, for all searched rows, the compatibility degrees with criteria given in query must be determined [2, 11]. Then, the problem of selection of appropriate rows, which should come into the resultant answer, must be resolved. A few methods can be given for the problem:

- selection of rows with maximal value of a compatibility degree with criteria given in the query,
- selection of rows which a compatibility degree value is greater than the average value of a compatibility degree,
- selection of rows for which a compatibility degree is greater than a given threshold value.

The threshold value of a compatibility degree can be introduced in DBMS on two different ways by [11]:

- initial establishing of a constant threshold value,
- establishing of condition on compatibility degree in SQL query notation.

The first approach seems to be less complicated in the implementation, but delimits the query construction in certain way. The second one is more flexible, but more difficult in the implementation.

To describe the notation of condition on a threshold value of compatibility degree in formal way, we define the following grammar fragment [11] in the BNF:

```
<extended_cond> ::= (<complex_fuzzy_cond>)<threshold_cond>
<threshold_cond> ::= <relational_operator><threshol_value>
<relational_operator> ::= = | < | > | <= | >= |<> 
<threshol_value> ::= <decimal number with range of [0, 1]>
```

The example from point 2.1 with consideration of the condition on the threshold value of the compatibility degree is the following:

"Find names of employees, who are about 50 years old, with compatibility degree greater than 0.7 (so in the answer should be included only these rows which compatibility degree with query criteria is greater than 0.7)"

```
SELECT LastName
FROM employees
WHERE (age is about 50) > 0.7;
```

2.3 Numerical Examples

Example 1

The user directs the following query to the database:

"Display names of employees who are about 50 years old and have seniority about 20 years, in the answer display only these rows which compatibility degree is greater than 0.6"

This query written in SQL language can have the following form [2, 7]:

```
SELECT LastName
FROM employees
WHERE ((age is about 50)
AND (seniority is about 20)) > 0.8;
```

Table 2. The Employees table

<table>
<thead>
<tr>
<th>EmpNo</th>
<th>LastName</th>
<th>Age</th>
<th>Seniority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Kowalski</td>
<td>48</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>Nowak</td>
<td>38</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Sowa</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Sroka</td>
<td>53</td>
<td>22</td>
</tr>
<tr>
<td>5</td>
<td>Maj</td>
<td>47</td>
<td>8</td>
</tr>
</tbody>
</table>

The age about 50 and seniority about 20 membership functions are defined by the Gaussian membership functions [8, 9] (Fig. 4 and Fig. 5).

The interpretation process of this query is as follows: for the age and seniority columns for each row of the Employees table compatibility degrees with query criteria are calculated (table 3). In the next step the total compatibility degree is calculated with the use of Zadeh t-norm (minimal value of compared compatibility degrees for each row) [8] (table 4).
Only these rows for which total compatibility degree is greater than 0.6 are included in the resultant answer (shadowed rows in the table 4).

**Example 2**

In the example, the DEPARTMENTS database contains fuzzy values and the specified query has the fuzzy conditions. In the database the Requirement table exists and includes estimated information about year-long needs for paper, toner and CDs (appropriate columns are implemented). In these columns the fuzzy values defined by proper membership functions are stored [6, 12].

The query directed to database has the following form:

> Display names of departments that turned requirement for quite a lot of toner and not many reams of paper at the same time. In the end answer only rows with compatibility degree greater than 0.7 can be included."

In this query both values (value determined in query criteria and column value) are fuzzy.

In SQL language this query can be written in the following way:

```
SELECT DepNo
FROM requirement
WHERE ((toner is quite_a_lot) AND (paper is not_many)) > 0.7;
```

The process of total compatibility degree calculation is presented in [6, 11]. General conception of compatibility degree calculation in case the database column and query criteria are represented by fuzzy sets is shown in Fig. 3.

**Example 3**

In the query examples presented so far, the fuzzy conditions were applied only in the WHERE phrase. The following example shows that the fuzzy values can take place in the other phrases, too [13]. The following query is directed to the DEPARTMENTS database:

> Display name of these departments, which turned year-long requirement for about 100 CDs in 2004 year and their maximal consumption for CDs was about 120. In the answer should be consider only these rows, which fulfil given criteria with the compatibility degree at least 0.8."

In SQL language this query can be written in the following way:

```
SELECT d.DepNo, DepName
FROM departments d JOIN requirements r
ON d.DepNo = r.DepNo JOIN consumption c
ON r.CDs = c.CDs
WHERE r.year = '2004' AND
(r.CDs is about 100) AND
(max(c.CDs) is about 120) >= 0.8
HAVING (max(c.CDs) is about 120) >= 0.8;
```

In the interpretation process of above query, the following steps may be separated [2]:

- first (on the filtrating stage – WHERE phrase) the rows accomplishing the crisp condition (concerning 2004 year) are chosen, and then, only for these rows the compatibility degree for the fuzzy condition is calculated: (r.CDs is about 100),
- then, only the rows, for which the compatibility degree is at least 0.8 are grouped,
- in turn, for created groups in the HAVING phrase the maximal values are calculated, and again, for the fuzzy condition in HAVING phrase the compatibility degrees are determined: (max(c.CDs) is about 120).

The resultant answer is created from these groups for which the compatibility degree fulfils the threshold value condition (at least 0.8) [2].

### 3. Context-Dependent Membership Functions

In the previous sections, each fuzzy value was determined by exactly one membership function. There are more complicated situations where this assumption can be the excessive simplification [9]. Let’s assume, that the following tables exist in the database:

- Marks: (StudentNo, SubjectNo, Mark)
- Students: (StudentNo, MajorNo, LastName)

The following query may be considered:

> "For the particular major, find these students, whose average is high."

This query in the SQL language notation can have the following form:

```
SELECT LastName
FROM students s JOIN marks m
ON s.StudentNo = m.StudentNo
GROUP BY MajorNo, LastName
HAVING AVG(Mark) is high;
```

The execution of this query requires the prior definition of the high average fuzzy set, determined by the appropriate membership function.

The assumption, only one membership function exists, can be an excessive simplification, because the definition of high average can vary for the different majors. The high average fuzzy value can be represented by the trapezoidal membership function (Fig. 6), which domain is a range [k, p] and the trapezium is defined by {(a, 0), (b, 1), (c, 1), (d, 0)} point set [2]. The form of this function is determined by the various set of {a, b, c, d} parameters for each major.

The question formulated in the beginning of this section is extended to the following form:

> "For the particular major, find these students, whose average is high. Display only rows with compatibility degree greater than 0.8."

The previous database is expanded by adding the Average table:

- Average: (MajorNo, HighAvg)
- Students: (StudentNo, MajorNo, LastName)
- Marks: (StudentNo, SubjectNo, Mark)
where the HighAvg column of the Average table stores membership function parameters for HighAvg fuzzy sets for the particular major.

\[
\mu(x)
\]

**Figure 6.** The parameterized trapezoidal membership function.

In this work, the following solution for queries of this type is offered:

```sql
SELECT LastName
FROM students s JOIN marks m
ON s.StudentNo = m.StudentNo
GROUP BY s.StudentNo, LastName
HAVING AVG(m.Mark) =
(SELECT (HighAvg)*>0.8
FROM Average a
WHERE a.MajorNo = s.MajorNo);
```

Based on conditions specified in the query, the parameterized membership function is transformed into the proper membership function. The appropriate membership function parameters are chosen according to major number (MajorNo column).

It is possible to apply linguistic values e.g. good, big, high etc. that can be interpreted in the context of data stored in the database. It implies the expansions of the database by adding a table, which describes the relationships between the linguistic layer and the data layer. The SQL query must use this table explicitly.

For example, in the query searching the high students at the University, very important attribute distinguishing membership function is gender, because usually men are higher than women. Therefore, term high has different meaning for men (Fig. 7) and women (Fig. 8).

**Figure 7.** The family of the membership functions for the linguistic values of the Height of Men linguistic variable (the high value is marked).

**Figure 8.** The family of the membership functions for the linguistic values of the Height of Women linguistic variable (the high value is marked).

### 4. Multidimensional Membership Functions

In the context-dependent approach, presented in the previous section, values of parameters describing the membership function are driven by data retrieved in the query process. In the case, additional table is created to store the context data. However, if the granularity of context data is huge or data has continuous nature, the creation of tables reflecting multiple values of the context is very difficult to implement or impossible. In such situations, the proper approach seems to be by increasing dimensionality of the membership function (see Fig. 9).

**Figure 9.** Multidimensional time-dependent membership function for men’s high height.

Constructing such a multidimensional membership function allows to formulate queries without the necessity of creating membership function for each value of context data. Final shape of the multidimensional function is estimated based on several sample measurements or statistical data, and can be prepared by experts of the appropriate domain. Fig. 9 presents how the height of men changes at the turn of decades. The \( t \) dimension represents polyvalent set of one context data (time), which in the case can be discrete, in other continuous. The \( x \) dimension represents the height of men. A tendency to widen the range of \( x \) va-
lus by the membership function and to change the angles of the trapezium through the decades may be observed. At the turn of the years the meaning of the men’s high height term changes – e.g. in the 1970 it is described by the range (170, 180) with compatibility degree 0.8, and in the 1990 by the range (174, 186) with the same degree (Fig. 10). Based on the function, the meaning of the term may be predicted in the future, e.g. in the 2010 the predicted range covers (178, 190) with the degree 0.8.

Figure 10. Shape of the membership function describing the men’s high height in the 2D view for the year 1990.

Considering the height of a high man living at the turn of the years specified in Fig. 11, changes of the compatibility degree and potential membership changes may be investigated (Fig. 12). The approach allows to analyze the qualification of the selected element of the entire population to appropriate fuzzy set described by the characteristic function.

Figure 11. The intersection between the membership function and the height of the chosen man (who is 174 cm high) through the years 1970-2010.

Presented example demonstrates concurrently that spatial membership function may be context–dependent, as well (Gender parameter defines the context).

5. Conclusions

The paper introduces the extension of the SQL language notation for fuzzy query processing with the special attention for context-dependent and multidimensional membership functions. In the first part the investigation of occurrence of fuzzy values in SQL queries directed to the databases was carried out. All techniques presented in the paper were implemented in the real database systems as additional operators or functions (e.g. PostgreSQL, Oracle platform). Given query examples were executed against sample data. However, the analysis of context-dependent

References