Yugoslav Journal of Operations Research
21 (2011), Number 2, 239-251
DOI: 10.2298/YJOR1102239H

FUZZY IMPROVEMENT OF THE SQL

Miroslav HUDEC
Institute of Informatics and Statistics,
Bratislava, Slovakia
hudec@infostat.sk

Received: December 2007 / Accepted: July 2011

Abstract: Structured Query Language (SQL) is used to obtain data from relational databases. Fuzzy improvement of SQL queries has advantages in cases when the user cannot unambiguously define selection criteria or when the user wants to examine data that almost meet the given criteria. In this paper we examine a realisation of the fuzzy querying concept. For this purposes the fuzzy generalized logical condition for the *WHERE* part of the SQL is created. It allows users to create queries by linguistic terms. The proposed model is an extension of the SQL so that no modification inside databases has to be undertaken.

Keywords: SQL, fuzzy query, generalized logical condition, database.

MSC: 94D05, 68P99, 90B99.

1. INTRODUCTION

SQL is a standard query language for relational databases. The SQL was initially presented by Chamberlin and Boyce [3]. Since then, it has been used for data selection in many relational databases and information systems. SQL is one of the key factors for accepting relational databases and for their use in storing and retrieving vast amount of data. Its advantages among others are: optimised work with relational database management systems and understandable interpretation for users.

The SQL uses two-valued logic (crisp logic) in querying process. It means that a small error in data values or in cases when user cannot unambiguously define the criteria by crisp values, may involve some inadequately selected data.

This limitation of the SQL can be avoided by fuzzy logic. The fuzzy set theory was initiated by Zadeh [12]. Since then, many researches and applications have been published. Fuzzy queries have appeared in the last 30 years to cope with the necessity to soften the Boolean logic in database queries. “A fuzzy query system is an interface to users to get information from database using (quasi) natural language sentences.” [2] In this period many fuzzy query implementations were proposed, resulting in slightly different languages. Brief analysis of these proposals and their variations can be found in [2]. This area is still interesting because there are possibilities for the improvement of existing approaches and for creating new ones. Cox [4] discussed applications of fuzzy set theory in several areas including database queries in his book. Issues and perspectives of fuzzy querying are discussed by Kacprzyk et al. [8]. Fuzzy querying realisations can be also found in [1], [10] and [11].

The goal of this paper is to present the fuzzy query approach which can be easily implemented and used. The Generalized Logical Condition (GLC) formula capable to capture linguistic expressions into the *WHERE* part of the SQL was initially created in [7]. Our approach is demonstrated on a case study with a database from the official statistics. This case study analysis was chosen because practical applications of fuzzy queries in statistics are rarely used. Linguistic expressions like: high rate of unemployment or medium precipitation are used very often, and statistical data are often collected with some errors and vagueness. In this paper we integrated ideas and results from the last mentioned paper in order to present concept of easy-to-use data selection approach for end-users.

This paper is organised as follows: The limitation of the SQL is briefly presented in section 2. Section 3 explains the fuzzy query approach and the GLC developed for this purpose. This querying process is described by the case study in section 4. Section 5 discusses some aspects concerning realisation of suggested fuzzy querying concept in the object-oriented programming environment. Finally, some conclusions are drawn.

2. THE SQL’S WHERE CLAUSE LIMITATION

The SQL uses crisp logic in querying process. The constraints of the crisp logic are explained on the following example:

```
select attribute_1,…,attribute_n
from T                                              (2.1)
where attribute_p > P and attribute_r < R.
```

The result of the SQL query is depicted in graphical mode in figure 1. Values *P* and *R* delimit the space of the relevant data. Small squares in the graph show database records. The user cannot obtain any information about records that are close to meet the query criterion (areas marked with grey hues). The area marked with the darkest hue contains records that almost meet the intent of the query.

The SQL makes crisp selection. It means that the record would not be selected even if it is extremely close to the intent of the query. This is the penalty paid for using crisp logic in selection criteria.

It is easy for users to decide which columns (attributes) of records they want to select. Boundary values separate relevant records from not relevant ones, and are defined in the *WHERE* clause of a query. In many cases it is not simple to know or to identify these values. This is usually the case when the boundary values are numeric values. In cases when the user can not unambiguously separate data (he is interested in from those he is not interested in) by sharp boundaries, or when the user wants to obtain data that are very close to satisfy queries, it is necessary to adapt the SQL to these requirements.

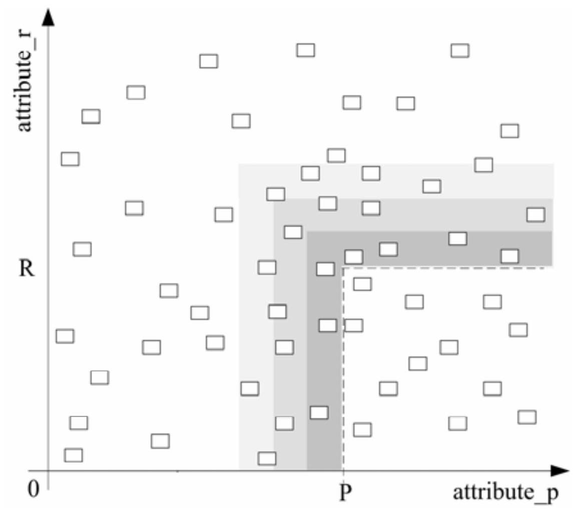


Figure 1: The result of classical query

The great number of software applications created incalculable accesses to wide variety of data. Actually, the classical selection of data is simply not enough in many cases. The improvement of the access to data consists in the language and the methods in "communication" with data. The language of the query is presented by linguistic terms. Fuzzy sets and fuzzy logic allow "communication" with data on a higher level than the crisp logic. In the next chapters, we present one realisation of the mentioned improvement.

3. SQL IMPROVED WITH FUZZY COMPONENT

3.1 The preliminary of proposed approach

For the further reading it is important to define the Query Compatibility Index (QCI). The QCI is used to indicate how the selected record satisfies a query criterion. The QCI has values from the [0, 1] interval with the following meaning: 0 – record does not satisfy a query, 1-record fully satisfies a query, interval (0, 1) – record partially satisfies a query.

All records from database tables are usually analysed for the QCI calculation. In case when linguistic terms represented by fuzzy sets are asymptotic to the zero line this is an acceptable method. When linguistic expressions presented by triangular, trapezoidal or bell shaped fuzzy sets are used, this approach has no advantages. All records from the database are examined. In many cases only a part of all records have QCI>0. It means that in client/server applications a large number of data uselessly flows across the net.

The goal of this research is to change values in the *WHERE* clause (*P* and *R* from query (2.1)) with linguistic terms and to calculate the lower bound of QCI from these linguistic expressions. Thus calculated lower bound is used as a parameter for database queries to select only records that have QCI>0. In the next step appropriate t-norms or t-conorms are used to calculate QCI values for all retrieved records. Figure 2 shows the steps and the modules of this approach. This approach decreases the amount of transferred data across nets and the QCI value is calculated only for the selected data.

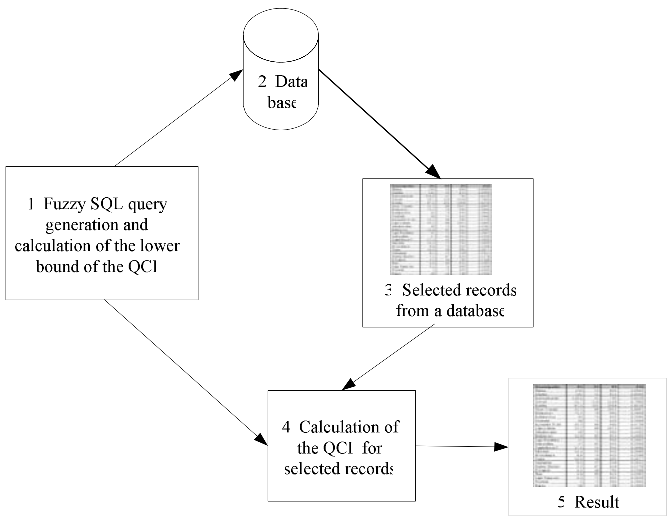

2 Data base
1 Fuzzy SQL query generation and calculation of the lower bound of the QCI
3 Selected records from a database
4 Calculation of the QCI for selected records
5 Result

Figure 2: Structure of the SQL with fuzzy component

3.2 Fuzzy SQL realisation

Conditions in queries contain these basic comparison operators: >, <, and = when numerical attributes in query conditions are used. These crisp comparison operators are adapted for fuzzy queries in the following way: operator > was improved with fuzzy set High value, operator < was improved with fuzzy set Small value and operator = was improved with fuzzy set About value. In addition, Operator ≠ is the negation of the operator = so this operator is not further analysed. Similar statement is valid for the operator **between** (it is similar to the operator =), from the fuzzy point of view.

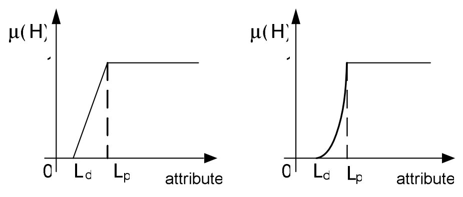
Crisp values in the *WHERE* clause (*P* and *R*) (2.1.) are replaced with linguistic terms in the following way:

```
select attribute_1,..., attribute_n
from T                                                    (3.1)
where attribute_p is High and attribute_r is Small.
```

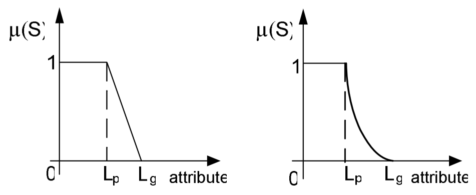
According to the above analysis, three types of linguistic terms in this research are supported: high value, small value and about value of attribute. All types are shown in figure 3. The figure shows two kinds of fuzzy sets for each linguistic term. The shapes of fuzzy sets could be broadened to allow better interpretability of a particular data selection task. The lower and the upper limit (L_d, L_g) have the same meaning in all fuzzy sets. In the QCI calculation step, the differences between fuzzy set shapes become important.

In case when it is required to find records which have high value of attribute, the query has the form (3.2), and its graphical presentation is given in figure (3.a).

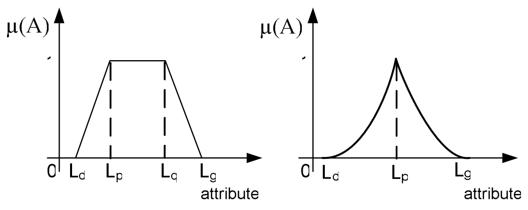
```
select attribute_1,…,attribute_n
from T                                              (3.2)
where attribute_p is High.
```



a) High value



b) Small value



c) About value
Figure 3: Fuzzy sets

The user determines the shape of the fuzzy set, lower limit of fuzzy set (L_d) and the value of full query satisfaction (L_p). The lower limit becomes a part of the *WHERE* clause. This clause access the database and selects records that have QCI>0. This query has the form shown in (3.3).

```
select attribute_1,…,attribute_n
from T                                                    (3.3)
where attribute_p > Ld
```

In case when it is required to find records with small attribute value (figure 3.b)), the query has the following structure:

```
select attribute_1,…,attribute_n
from T                                                    (3.4)
where attribute_p < Lg
```

In case when it is required to find records that have attribute value about the exact value and when fuzzy sets from figure 3.c are used, the query has the following structure:

```
select attribute_1,…,attribute_n
from T                                                    (3.5)
where attribute_p > Ld and attribute_p < Lg
```

Values L_d, L_p, L_q, L_g are used to define the analytical form of the fuzzy set. According to this analytical form, the QCI values are calculated for all selected records. The creation of the GLC is explained in the next subchapters.

3.2.1. Where clause connected with and operator

In order to find generalized formula which describes fuzzy queries, it is necessary to analyse query with all above mentioned logical conditions:

```
where a is High and b is Small and c is About.
```

In case when two attributes are in a *WHERE* clause all t-norms can be used. However, for fuzzy SQL, it is required to use more than two attributes in the *WHERE* clause of a query is required. According to the associative rule of t-norms, only a few t-norms can be easily aggregated for more than two attributes. These t-norms are as follows [9]:

- minimum:

$$\mathrm{QCI} = \min(\mu(x_i)) \quad i = 1,\dots, n \ , \tag{3.6}$$

- product:

$$\mathrm{QCI} = \prod_{i=1}^{n} \mu(x_i) \ , \tag{3.7}$$

- bounded difference (BD):

$$\mathrm{QCI} = \max(0, \sum_{i=1}^{n} x_i - n + 1) \quad . \tag{3.8}$$

The QCI, for the logical conditions mentioned above, has the following forms: for minimum (3.6):

$$\mathrm{QCI} = \min(\mu(\mathrm{a}), \mu(\mathrm{b}), \mu(\mathrm{c}) \quad , \tag{3.9}$$

for product (3.7):

$$\mathrm{QCI} = \prod(\mu(\mathrm{a}), \mu(\mathrm{b}), \mu(\mathrm{c})) \, , \tag{3.10}$$

for BD (3.8):

$$\mathrm{QCI} = \max(0, \mu(\mathrm{a}) + \mu(\mathrm{b}) + \mu(\mathrm{c}) - \mathrm{n} + 1) \quad , \tag{3.11}$$

where *μ(a)*, *μ(b)* and *μ(c)* denotes membership degrees of fuzzy sets respectively.

Minimum (3.6) determines the lowest value of membership degrees for all selected attributes. Product (3.7) takes into account values of all attributes. BD (3.8) satisfies the law of contradiction.

If record membership degree is zero for only one attribute, then the record QCI is zero. This fact is the principle for extraction records that have the QCI values in interval (0, 1] only. Records that have membership degrees for all linguistic terms greater than zero are selected from a database.

If for attribute **a** lower limit is L_d, for attribute **b** upper limit is L_g , and for attribute **c** limits are L_d and L_g, then *WHERE* clause becomes as following:

where a > L_d and b < L_g and (c > L_d and c < L_g). (3.12)

3.2.2. Where clause connected with or operator

For logical condition:

```
where a is High or b is Small or c is About ,
```

selection of appropriate records and calculation of the QCI is analogous to the procedure in the previous chapter. The difference is in logical operator and in adequate choice for t-conorms. Maximum and bounded sum t-conorms could be easily aggregated in case of n attributes [9]:

- maximum:

$$\mathrm{QCI} = \max(\mu(\mathrm{x}_{\mathrm{i}})) \quad \mathrm{i} = 1, \ldots, \mathrm{n} \ , \tag{3.13}$$

- bounded sum (BS):

$$\mathrm{QCI} = \min(1, \sum_{i=1}^{n} x_i) \ . \tag{3.14}$$

The QCI for logical conditions has the following forms:
for maximum (3.13):

$$\text{QCI} = \max(\mu(\text{a}), \mu(\text{b}), \mu(\text{c})), \tag{3.15}$$

for BS (3.14):

$$\text{QCI} = \min(1, \mu(\text{a}) + \mu(\text{b}) + \mu(\text{c})) \quad , \tag{3.16}$$

where *μ(a)*, *μ(b)* and *μ(c)* have the same meaning as in formulas for the *and* operator.

In case when logical *or* operator is used, the record's QCI is zero when the record membership degree is zero for all the attributes. The records that have membership degrees for at least one linguistic term greater than zero are selected from the database.

If boundaries for all attributes are the same as for the logical *and* operator, then the *WHERE* clause has the following structure:

$$\texttt{where a > } L_d \texttt{ or b < } L_g \texttt{ or (c > } L_d \texttt{ and c < } L_g\texttt{).} \tag{3.17}$$

3.2.3. Generalisation

Let symbol $\otimes$ has the following structure:

$$\otimes = \begin{cases} \mathit{and}, & \text{for min, product and BD} \\ \mathit{or}, & \text{for max and BS} \end{cases} \quad . \tag{3.18}$$

The *WHERE* clause of a query, in consonance with equations (3.12) and (3.17) gets the following structure:

$$\text{where a>L}_\text{d} \otimes \text{b<L}_\text{g} \otimes (\text{c> L}_\text{d} \text{ and c<L}_\text{g}) . \tag{3.19}$$

Generally, the structure of *WHERE* clause can be written in the form:

$$\text{where } \bigotimes_{i=1}^{n} (a_i \circ L_{xi}) \quad , \tag{3.20}$$

where n denotes the number of attributes with fuzzy constraints in the *WHERE* clause of a query and

$$a_i \circ L_{xi} = \begin{cases} a_i > L_{di}, & \text{a}_\text{i} \text{ is High} \\ a_i < L_{gi}, & \text{a}_\text{i} \text{ is Small} \\ a_i > L_{di} \text{ and } a_i < L_{gi}, a_i \text{ is About} \end{cases} \cdot \tag{3.21}$$

The (3.20) structure can be entitled as the GLC for fuzzy queries.

3.2.4. Calculation of the QCI

All selected records from a database have the QCI value greater than zero. The QCI values for these records are calculated as the next step. The membership degrees of fuzzy sets for each attribute are calculated (for example for trapezoidal fuzzy set values L_d, L_p, L_q and L_g are used to obtain the membership degree). On the basis of the selected t-norm or t-conorm, the QCI for each record is calculated at the end of this process.

The differences between t-norms are important when the correlation between attributes is taken into account. In case when correlation is unknown or does not exist, minimum or product t-norms are used. In other cases, BD t-norm is used. The similar is valid for t-conorms. If a correlation exists, the BS t-conorm is used, and in other cases maximum t-conorm is used.

4. CASE STUDY

This fuzzy query approach is under the pilot development for information systems in official statistics. Statistics have become one of potential applications of fuzzy query techniques, due to great number of data available to be used. The objectives of use of statistical data are various - from making state administration policy to decisions of investments. Statistics also works on data dissemination for various kinds of statistical data users. Crisp tools also enable useful data dissemination, but the fuzzy approach satisfies the demand on data in human understandable form, and supports obtaining data using linguistic terms.

This system is tested on data taken from the Information system for support regional development in the Slovak Republic [6]. In this case study, districts with high unemployment rate and small area size are sought. The high unemployment density is analysed as an illustrative example. The simplified query has the following form (the simplification is for better comprehensibility; in the *WHERE* and *FROM* part of the query the names of tables and joins conditions connected by primary and foreign key are skipped):

```
select district, unemployment, area
from T
where unemployment is High and area is Small.
```

Unemployment is represented by the High value fuzzy set with these parameters L_d=8% and L_p =10% and the shape as in figure 3.a on the left side. The Small value fuzzy set with parameters L_p=300km^2 and L_g =650km^2 and the shape as in figure 3.b on the left side describes the area of district attribute. According to these parameters the query has the following form:

```
select district, unemployment, area
from T
where unemployment>8 and area<650.
```

The query selects 26 of 79 districts from the database. Because of non existence of correlation between these two attributes the minimum (3.6) and product t-norms (3.7) are used to calculate QCI values. The result of the query is presented in table 1. The value of minimum t-norm is used for district ranking.

The result of the query is not ranked as in results from a crisp SQL query by one of the selected attributes. The result is also not ranked by weighted coefficients of the selected attributes. In a fuzzy query all attributes have equal importance, and their ranking is done by the degree of matching the query criterion. With fuzzy querying it is possible to treat each record according to its QCI value. This enables decision makers to allocate resources more precisely according to the value of the query satisfaction for example.

Table 1: Result of the fuzzy query

District	Unemployment [%]	Area [km^2]	μ (unemp)	μ (area)	QCI (min)	QCI (prod)
Bánovce nad Bebravou	8,01	462	0,003	0,538	0,003	0,002
Ružomberok	9,63	647	0,813	0,009	0,009	0,007
Košice II	8,07	80	0,034	1,000	0,034	0,034
Stará Ľubovňa	8,92	624	0,461	0,074	0,074	0,034
Topoľčany	9,23	598	0,617	0,149	0,149	0,092
Spišská Nová Ves	13,92	587	1,000	0,179	0,179	0,179
Krupina	14,48	585	1,000	0,186	0,186	0,186
Gelnica	16,99	584	1,000	0,187	0,187	0,187
Košice III	8,44	17	0,220	1,000	0,220	0,220
Svidník	12,62	550	1,000	0,287	0,287	0,287
Sobrance	20,69	538	1,000	0,319	0,319	0,319
Zlaté Moravce	9,97	521	0,986	0,368	0,368	0,363
Žiar nad Hronom	12,58	518	1,000	0,378	0,378	0,378
Dolný Kubín	9,17	492	0,584	0,452	0,452	0,264
Sabinov	16,20	483	1,000	0,476	0,476	0,476
Poltár	18,59	476	1,000	0,497	0,497	0,497
Detva	14,74	449	1,000	0,574	0,574	0,574
Medzilaborce	14,84	427	1,000	0,636	0,636	0,636
Kysucké Nové Mesto	9,32	174	0,661	1,000	0,661	0,661
Turčinske Teplice	11,22	393	1,000	0,735	0,735	0,735
Stropkov	11,44	389	1,000	0,746	0,746	0,746
Levoča	13,22	357	1,000	0,836	0,836	0,836
Šaľa	10,69	356	1,000	0,840	0,840	0,840
Bytča	9,77	282	0,883	1,000	0,883	0,883
Partizánske	9,85	301	0,924	0,997	0,924	0,921
Banská Štiavnica	13,88	292	1,000	1,000	1,000	1,000

4.1. Multiple usage of the same query

In case of multiple usage of the same query the meaning of the query remains stable. Only parameters of the fuzzy sets are changeable. As an example, the query from the case study can be used: **User wants to select territorial units with high unemployment rate (a_1) and small area (a_2)**. This linguistic expression is the base for multiple usages, for example in two different directions:

- User wants to select territorial units which satisfy mentioned linguistic terms in different time periods. Parameters and shapes of fuzzy sets could be changed to satisfy new requirements. In this case, the state in labour market or territorial unit structure can be changed and implemented into fuzzy sets.
- The same query could be used for territorial unit selection in other regions with different labour market situation and territory division. Parameters of fuzzy sets are also changed to better describe the situation.

Combination of both ways is also possible. This small discussion shows the importance of queries based on linguistic expressions. Table 2 shows the evolution of one fuzzy query. Linguistic meaning of the query remains unchanged. Parameters of Small and High fuzzy sets are changeable to allow query adaptation to new requirements.

Table 2: Linguistic meaning of a query is not changed

Period/ Region	Period 1	…	Period n
Territory 1	select * from T where a_1 is High and a_2 is Small		select * from T where a_1 is High and a_2 is Small
…			
Territory n	select * from T where a_1 is High and a_2 is Small		select * from T where a_1 is High and a_2 is Small

In the next chapter, the implementation of the GLC (3.20) into object-oriented programming is discussed.

5. THE SOFTWARE REALISATION OF THE GLC

The GLC can be designed as a generic class in an object-oriented programming language. The inheritance and replacement are used to obtain particular queries from the generic one. In this work the GLC is under development in Microsoft Visual Basic programming language. In order to allow users to work with this system, the fuzzy module, as well as modules for databases connections, and for providing results in a usable and understandable form, in e.g. print-ready tables or in xls format for further use are under development. For statistical information systems, providing the result in a thematic map is very useful too.

The fuzzy SQL is an independent module and it can be used when the user can not unambiguously define selection criteria, or when the user wants to examine data that almost meet the given criteria. In other cases, the classical SQL gives satisfying results and the request for the fuzzy SQL does not exist.

The modularity of this approach supports changes and improvements for each module independently. For example, fuzzy module can be improved while other modules remain the same.

The state of the art of this approach depends also on theoretical and practical development of fuzzy database systems. More about fuzzy database systems can be found in [5]. All statistical databases known to the author are developed in classical relational databases. This trend dominates in development of new statistical information systems and databases. The fuzzy improvement of the SQL presented in this work has the significant perspective for use in official statistics.

6. CONCLUSION

SQL is used in all major database applications. The suggested fuzzy query improvement allows using linguistic terms and do not change the structure and the concept of obtaining data from databases.

The GLC was created to improve the *WHERE* clause of a query with fuzzy logic, and to keep accessing relational databases in the unchanged way. The query technique presented in this paper enables an effective data selection. Only the records that have QCI>0 are selected from the database. In the client/server applications, this approach decreases the amount of data transferred from the server to the client side of an information system. Moreover, users do not need to learn new query language and execute additional calculation steps.

This system is under development primarily for statistical information systems. Expressions like: high rate of unemployment, high migration level, etc are very often used in statistics. The goal of this system is to capture these expressions and make them suitable for database queries. In many cases (not only for statistic data), user cannot unambiguously select relevant data by sharp boundaries, or he cannot unambiguously prove that the chosen boundary value is the best one. Users also want to examine data that almost meet the given criteria and to know the distance to the full query satisfaction.

The information system at users' disposal usually contains some software for their daily work (e.g. database updating, reports creation, software for work with geographical data on thematic maps). The fuzzy SQL tool could be used to improve their work in wide area of tasks because it creates simple and easy-to-use querying and data mining tool.

The fuzzy SQL is in this approach an independent module, and it can be applied when the user wants to use linguistic terms in queries. The research done in this work could be continued in searching the methods for fuzzy query improvements.

REFERENCES

[1] Bosc, P., and Pivert, O., "SQL query functionality on top of a regular relational database management system", in: O. Pons, M.A.Vila, and J. Kacprzyk (eds.), *Knowledge Management in Fuzzy Databases*, Physica Publisher, Heidelberg, 2000, 171-190.

[2] Branco, A., Evsukoff, A., and Ebecken, N., “Generating fuzzy queries from weighted fuzzy classifier rules“, *ICDM workshop on Computational Intelligence in Data Mining*, (2005) 21-28.
[3] Chamberlin, D., and Boyce R., “SEQUEL: A structured english query language”, *ACM SIGMOD Workshop on Data Description, Access and Control*, (1974) 249-264.
[4] Cox E., *Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration*, Morgan Kaufman, San Francisco, 2005.
[5] Galindo, J., Urrutia, A., and Piattini, M., *Fuzzy Databases, Modeling, Design and Implementation*, Idea Group Publishing, London, 2006.
[6] Hudec, M., and Priehradníková, L., "Significance of the information system for support of regional development in the Slovak Republic, *INFO-M*, 17 (2006) 4-10.
[7] Hudec, M., “Fuzzy improvement of the SQL”, *8th Balkan Conference on Operational Research*, (2007) 255-265
[8] Kacprzyk, J., Pasi, G., Vojtáš, P., and Zadrożny, S., “Fuzzy querying: Issues and perspectives”, *Kybernetika*, 36 (2000) 605-616.
[9] Siler, W., and Buckley, J., *Fuzzy Expert Systems and Fuzzy Reasoning*, John Wiley & Sons, Inc., New Jersey, 2005.
[10] Wang, T.C., Lee, H.D., and Chen, C.M., "Intelligent queries based on fuzzy set theory and SQL" *Joint Conference on Information Science*, (2007) 1426-1432.
[11] Werro, N., Meier, A., Mezger, C., and Schindler G., “Concept and implementation of a fuzzy classification query language”, *DMIN* (2005) 208-214.
[12] Zadeh, L. “Fuzzy sets”, *Information and Control*, 8 (1965) 338-353.