Acta Informatica Pragensia
2(2), 2013, 25–38, DOI: 10.18267/j.aip.21
Online: aip.vse.cz

Section:
Peer-reviewed papers

Flexible data queries

Miroslav Hudec[1]

[1] Faculty of Economic Informatics, University of Economics in Bratislava,
Dolnozemská cesta 1, 852 35 Bratislava, Slovak Republic

miroslav.hudec@euba.sk

Abstract: The Structured Query Language (SQL) has been established as a standard for querying relational databases. However, users face the problem how to define their requirements for data by the exact query conditions. This work examines advantages of fuzzy queries, which provide a better representation of the user requirements by expressing imprecise conditions through linguistic terms. Further, the paper discusses solving empty and overabundant answer problems, revealing similarities in database entities and applying preferences in query conditions. Finally, paper discusses practical realisations of fuzzy queries.

Keywords: Fuzzy logic, SQL, Benefits of fuzzy queries

1 INTRODUCTION

A query against a collection of data (in database) provides a formal description of the items of interest to the user posing this query [12]. For querying relational databases the Structured Query Language (SQL) has been developed. SQL queries use two-valued Boolean concepts (logical conditions) to describe the entities users are looking for e.g. ***select*** *municipalities* ***where*** *altitude above sea level is greater than 1000 m*. From the computational point of view this is powerful way for selecting a sub set of entities from a database.

On the other side of a database are people which use the natural language in communication, searching for useful information and reasoning. Human approximate reasoning, although without precise measurements, is a very powerful way for finding solutions of various problems. Therefore, for people it is more convenient to use the concept based on natural language in data selection processes.

"Computing, in its usual sense, is centred on manipulation of numbers and symbols." [20] In contrast, computing with words is inspired by the singular human capability to perform a wide variety of tasks without precise measurements and computations [20]. These terms include a certain vagueness or uncertainty that information systems operated on the two-valued logic {true, false} do not understand and therefore cannot use [6]. When an entity with attribute value of r_a satisfies a condition, then we expect that an entity having attribute value close to r_a should also satisfy the condition, but with a slightly lower satisfaction degree.

In these cases the uncertainty is not based on randomness, it cannot be presented as a precise numeric value. This type of uncertainty is called fuzziness [23]. It implies that fuzzy set and fuzzy logic theory [21] is a rational option which may offer the solution for above mentioned tasks. The fuzzy logic directly employs calculations with linguistic terms in order to solve data selection (retrieval) problems.

Fuzzy logic reduces the complexity of mathematical analysis of problems in comparison with the classical approaches [16]. By the two-valued logic, two elements of the analysed universe (e.g. customers or municipalities) can be discerned only if one has and another does not have the examined property. Therefore, the number of necessary properties (attributes in queries) increases. Contrary, in fuzzy logic the complexity of the problem can be reduced by including the intensity of the examined property (e.g. query conditions). This allows us to distinguish elements with the same property, based on the intensity of matching it.

The paper has two objectives. The first intent is to present advantages of fuzzy queries for researchers and practitioners, which are highly depended on data (analysts, decision makers…) and are not experts in fuzzy logic and databases. The second intent is discussing issues and solutions of flexible queries such as empty and overabundant answers, similarities, preferences and practical realisations. Section 2 discusses limitations of SQL queries. Introduction to the fuzzy logic is provided in Section 3. Section 4 illustrates fuzzy queries. Section 5 is focused on practical realisations. Section 6 is dedicated to particular problems which ca be efficiently solved by fuzzy queries. Finally, concluding remarks are drawn in Section 7.

2 LIMITATIONS OF SQL

The SQL has been introduced in [11]. Since then the SQL has been used in variety of relational database management systems. The applicability of the SQL may be regarded as one of the mayor reasons for the success of relational databases in the commercial world [18].

Let R be a table in a relational database. A set of tuples t (entities) is then defined as relation on Cartesian product:

$$R \subseteq \{t \mid t \in Dom(A_1)\, x\, Dom(A_2)\, x \cdots x\, Dom(A_n)\} \quad \textbf{(1)}$$

where A_i is the entity's attribute and $Dom(A_i)$ is its associated domain.

In traditional SQL queries the entity in a database can either fully satisfy the intent of a query Q_s or not. Other options do not exist. Let $A(Q_s)$ be the set of answers to query Q_s defined in the following way:

$$A(Q_S) = \{t \mid t \in R \wedge \varphi(t) = 1\} \quad \textbf{(2)}$$

where $\varphi(t)$ indicates that the selected tuple t meets the query criterion.

The usefulness of the SQL for queries on relational databases is indisputable. Nevertheless, limits of using dichotomous criteria might appear. The limitation is explained on the following query:

```
select *
from [Table T]
where attribute_1 > A1 and attribute_2 < A2
```
(3)

The result of the query is depicted in the graphical mode in Fig. 1. Values A_1 and A_2 delimit the space of the relevant data. Small squares in the graph show entities in a database. From the graph it is obvious that three entities are very close to meet the query criterion (3). For example, these entities could be customers and direct marketing should also consider these customers in motivation issues.

The result is not surprising, because the SQL uses the two-valued (crisp) logic in querying. It means that the entity is not selected even though it is extremely close to meeting the intent of the query.

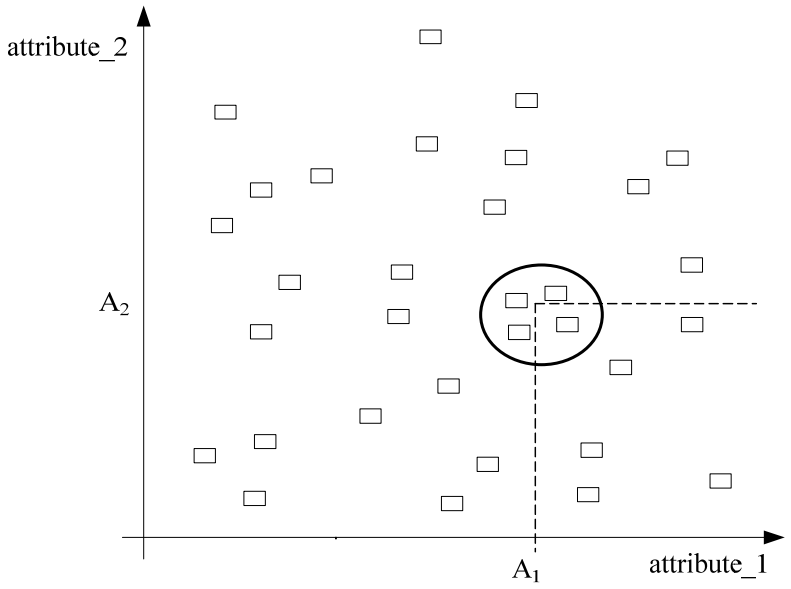

Fig. 1. The result of the classical query.

If the SQL is used for solving this problem, the SQL relaxation could be in the form [4]:

```
select *
from [Table T]
where ateribute_1 > A1-a1 and attribute_2 < A2+a2
```
(4)

where a_1 and a_2 are used to expand the initial condition to encompass entities that are near the boundary of a query. According to [4] this approach has two main disadvantages. Firstly, the meaning of the initial query is diluted in order to capture neighbouring entities, but these entities satisfy a query with the same degree as all entities meeting the initial query (3). It means that the difference between original and neighbouring data (additional records selected by the relaxed query condition) does not exist. Secondly, what about entities that are very close to satisfy the expanded query? Is it useful to make another expanding of a query? In this way more data from the database is selected, but not better data.

If we include not only the information whether or not record meet the query condition but also the intensity of satisfying the query condition then we are able to solve issue depicted in Fig. 1.

3 FUZZY SETS AND FUZZY LOGIC IN BRIEF

The concept of fuzzy sets was initially introduced in [21] where it was observed that precisely defined criteria of belonging to a set often could not be defined. In the classical set theory an element either belongs or does not belong to a set. The fuzzy set theory allows describing the intensity, uncertainty and ambiguity described by linguistic terms. The intensity is described by a membership function μ valued in the unit interval [0, 1]. Let consider a set called high unemployment rate (HUR) [8]. The HUR set can be presented by fuzzy set HUR shown in Fig. 2. User could define that the unemployment rate (UR) greater and equal than 10% fully belongs to the HUR concept, the

unemployment rate smaller than 8% definitely is not HUR and unemployment rate between 8% and 10% partially belongs to the HUR concept. Naturally, when unemployment rate is closer to 10%, then it stronger belongs to the HUR concept. The fuzzy sets approach is not limited to clear sets' boundaries.

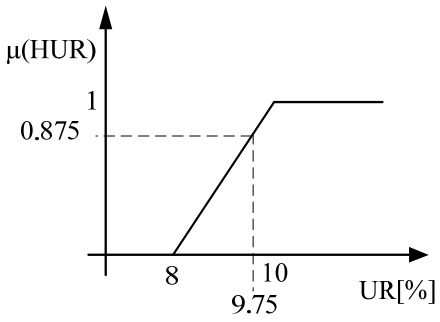

Fig. 2. Fuzzy set for the high unemployment rate concept [8].

A query usually contains several atomic (elementary) conditions in the overall query condition. For example, query (3) contains two elementary conditions merged by the *and* logical operator. Let's consider the condition: *unemployment is high and migration is small.* The satisfaction degree for each elementary condition takes value from the [0, 1] interval. For example the territorial unit satisfies the high unemployment with 0.8 and the second elementary condition with 0.65. We should apply aggregation functions dealing with membership degrees. For the logical *and* operator, these functions are called t-norms [13]. The min t-norm:

$$\mu(t) = \min(\mu_i(\mathrm{a_i})), \qquad \mathrm{i} = 1,...,\mathrm{n} \tag{5}$$

where $\mu_i(a_i)$ denotes the membership degree of the attribute a_i to the i-th fuzzy set and $\mu(t)$ denotes satisfaction degree to the overall condition for entity t is often used because of its simplicity. Discussion about other t-norm functions can be found in e.g. [4], [5] and [10]. Interesting introduction into fuzzy logic and its practical applicability presented in a popular way can be found in e.g. [15].

4 FLEXIBLE QUERIES

In queries based on fuzzy logic the entity in a database can satisfy the intent of a query Q_F with the particular membership degree. Let $A(Q_F)$ be the set of answers to query Q_F defined in the following way:

$$A(Q_F) = \{(t, \mu(t)) \mid t \in R \wedge \mu(t) > 0\} \tag{6}$$

where $\mu(t)$ indicates how the selected entity t meets the query criterion. If $\mu(t) = 1$, the entity fully meets the query criterion. Value of $\mu(t)$ in the interval (0, 1) means that the entity t partially satisfies the query criterion.

In fuzzy queries, instead of numbers in query conditions (e.g. unemployment rate > 5%) and traditional comparison operators (>, =, <) we can apply fuzzy sets. The operator > (greater than) was in flexible queries improved with the fuzzy set *high value* (Fig. 1) to catch terms like *unemployment rate is high*. The operator = (equal) was fuzzified through the use of the term *more or less* equal to value *a*, where *a* is a real number. It can be described as a trapezoidal or triangular (a = B = C) fuzzy set *about value* (Fig. 3). In the same way other operators like < or *between* can be fuzzified.

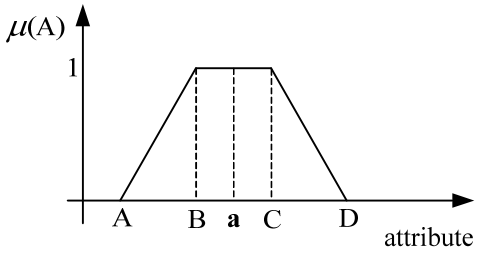

Fig. 3. Fuzzy set About (more or less equal to).

According to the above mentioned facts, the example of a fuzzy query e.g. find appropriate areas for tourism has the following form:

```
select municipality
from [Table T]
where air pollution is Small and                    (7)
number of sunny days is High and
altitude above sea level is about 1500 meters
```

The meaning of a fuzzy query is obvious at the first glance because it is expressed with the linguistic terms. User is not forced to create the query criterion by numbers. Because linguistic terms have different meaning for different users and contexts, user should define parameters of these terms. For example the term high temperature has different meaning for temperatures describing activities inside the Sun or for selecting the appropriate tourism destination. Moreover, for different tourists the term high temperature has different meaning. Therefore, the parameters of membership functions should be adjusted according to user's requirements. For the example (7) *small*, *high* and *about* fuzzy sets and their respective parameters are depicted in Fig. 4. The result of this query is illustrated in the Table 1.

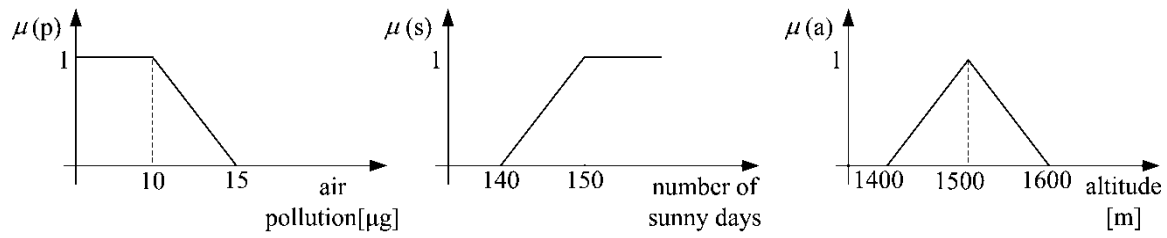

Fig. 4. Fuzzy set for the example (7).

Municipality	Pollution (P)	μ(P)	Number of sunny days (S)	μ(S)	Altitude (A)	μ(A)	Satisfaction degree	
M7	**8.2**	**1.00**	**161**	**1.0**	**1500**	**1.00**	**1**	
M5	10.2	0.96	149	0.9	1495	0.95	0.9	←
M2	9.5	1.00	147	0.7	1520	0.80	0.7	←
M3	11.0	0.80	145	0.5	1510	0.90	0.5	
M8	14.1	0.18	160	1.0	1430	0.30	0.18	

Table 1. Areas suitable for the tourism activities.

In Table 1 *μ(P)* denotes the membership degree to the small pollution concept, *μ(S)* denotes the membership degree to the high number of days with sunshine and *μ(A)* denotes the membership degree to the altitude about 1500 m concept. The satisfaction degree, calculated by the minimum t-norm (5) represents membership degree to the *small air pollution and high number of sunny days and altitude about 1500 meters* concept. If tourist cannot travel to *M7*, the next choice is the *M5* that almost satisfies the intent of the query. Again, if it is not possible to realise tourism activities in *M5*, the next choice is *M2* and so on. It is important to emphasize that ranking is not done by one indicator, their combination by weighted coefficients, etc. All indicators have the same importance and ranking is done according to the satisfying the concept created in the query criterion. If SQL were used, this additional valuable information would remain hidden.

The aim of this paper is not to examine the technical realisation of fuzzy queries on relational databases (transformation from linguistic terms to the SQL *where* clause, selection all candidates and calculate their membership degrees). Concerning this topic, readers can find more in e.g. [1], [9], [19] and [22].

5 PRACTICAL REALISATIONS

Issues of practical realisation are usually divided into two main areas: realisation of an application layer and intuitive design of a user interface. The former should support imprecise queries and keep a relational database structure unchanged. An illustrative example of the realisation of fuzzy query by the fuzzy Generalized Logical Condition [9] is shown in Fig. 5. In the first step users defines all required parts of queries (attributes of interest and conditions). In the next step application layer converts fuzzy queries into the SQL query in a way that all candidates (entities which satisfy query condition with degree greater than 0) are selected from a relational databases. In the reverse direction, membership degrees to each elementary condition and to the overall query are calculated and result is provided on a user interface.

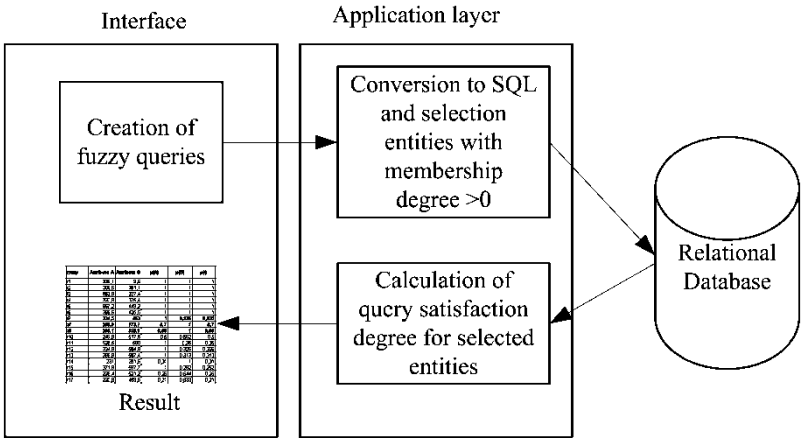

Fig. 5. Fuzzy query realisation.

SQL query is not so easy to use for novice users. The same holds for fuzzy queries. Although fuzzy queries have significant advantages, an interface should be as intuitive as possible to help users to formulate their queries. The PostgreSQL_f approach [17] offers the solution. Because, all the functionalities created in the PostgreSQL_f cannot be easily managed by users, the ReqFlex was created to offer the intuitive construction of flexible queries.

User should have two options: (i) choose parameters of fuzzy sets according his/her preferences and (ii) a fuzzy query application calculates these parameters either from users' previous preferences or by mining from the current database content. Concerning the (i) both above mentioned approaches ([9], [17]) offer the solution. Regarding (ii), [17] is able to offer parameters from users previous preferences whereas approach [9] can mine parameters from the current database content by adding methods discussed in [10]. Ideally, all discussed ways should be integrated.

6 BENEFITS OF FUZZY QUERIES

In this section, some benefits of fuzzy queries in comparison with traditional SQL queries are examined, namely solving empty and overabundant answer, membership degrees, similarities and preferences.

6.1 TREATMENT OF EMPTY AND OVERABUNDANT ANSWER PROBLEMS

The empty answer means that no entity meets a query condition. At the first glance, this may not be a problem. It is also an informative answer. But, when the user wants to know why an empty answer has appeared or how close entities to match he query condition are, user have to modify parameters in query conditions and process another, similar query. In theory this is known as "*empty answer problem*", that is, the problem of providing some alternative data when there is no data fitting the query [3].

Let look at the query (7) from the traditional SQL perspective:

```
where air pollution < 10 and number of sunny days > 150
and altitude above sea level = 1500;
```

and imagine that the *M7* does not exist in database (Table 1).

As a consequence, no entity from our database is selected. Moreover, it is not simply to detect that a small relaxation of the condition like:

```
where air pollution < 10.3 and number of sunny days > 148
and altitude above sea level = 1495;
```

will retrieve one entity.

Let consider the query from fuzzy logic perspective again. The solution of fuzzy query shown in Table 1, when excluding M7 (for the purpose of demonstration, M7 does not exist in a database), tells us that no municipality fully meets the query condition, but also tell us that several municipalities partially meet the query. In this case, no further adjustment and query processing is required.

Fuzzy queries requires more computational time due to transformation from fuzzy to SQL conditions, selection all candidates and calculate their membership degrees. On the other hand, in case of empty answer problems, one fuzzy query could solve this issue and therefore, leads to less computational and user burden. However, empty answer could also appear in fuzzy queries [3]. In order to solve this issue relaxation of the initial query has been suggested in [3]. The relaxation is a continuous and time consuming process. When query contains several fuzzy elementary conditions, then each elementary condition should be relaxed and tested whether empty answer disappeared. The querying process is repeated until the answer is not empty or the modified query becomes semantically far from the original one [3].

The opposite situation appears when query retrieves overabundant number of records, which fully meets the query condition (if large number of entities partially satisfies the query condition, then applying threshold by α-cut is an effective and fast solution). Overabundant answer problem is discussed in [3] where predicates are intensified in order to obtain stronger restriction. Another solution is adding additional elementary condition (predicate) to the initial query. In practice, it is not an easy task for users to find the most appropriate predicate which can effectively solve this problem. To solve this issue an approach which selects a set of predicates according to their degree of semantic correlation with the initial query is suggested in [2]. In both approaches we have to avoid a deep modification of the query (semantically far from the original one).

Approaches focused on solving empty and overabundant answer problems [2], [3] operates with fuzzy sets parameters defined “a priori”. We would like to mention another solution for these issues: construction of fuzzy sets parameters from the current database content [10]. In this way a user obtains

suggested parameters of fuzzy sets before running a query that may reduce both issues: empty and overabundant answer.

6.2 MEMBERSHIP DEGREES TO THE QUERY CONDITION

Fuzzy queries make distinction between selected entities by membership degree. In this way selected entities are ranked downwards from the best to the worst according to the membership degree, as was demonstrated in previous sections. Let's now examine the condition:

```
where attribute_a = v
```

where v is a real number.

There is higher jeopardy of empty answer. It could happen that no entity have the exact value of v for the examined attribute, although there might exist many entities with similar values. Traditional queries could use intervals in the following way:

```
where attribute_a > v1 and attribute_a < v2
```

where $v_1 < v < v_2$ and v_1 and v_2 are also real numbers.

If the interval is longer, more entities will be selected. But, an entity near the edge of the interval satisfies the query condition in the same way as entity near the middle of the interval. A small interval might lead to an empty answer. Therefore, the question is how to choose optimal values of v_1 and v_2. In case of the fuzzy query this question is irrelevant. We can create triangular fuzzy set on the examined interval. In this way entities near the edges of the interval have lower membership degrees than entities near the middle of the interval. In an extreme situation the interval can be the whole attribute's domain. Changing values v_1 and v_2 will lead to different number of selected entities, but their ranking will not be changed. Applying the threshold value [19], we are able to provide only entities that significantly meet the query condition. Anyway, reasonable values of v_1 and v_2 should be chosen in order to avoid longer evaluation of the fuzzy query. Sharp interval and fuzzy set are depicted in Fig. 6.

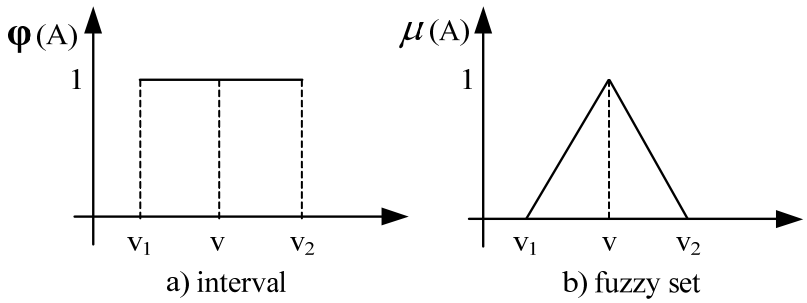

Fig. 6. An interval and a fuzzy set.

6.3 EVALUATION OF SIMILAR ENTITIES IN A DATABASE

Linguistic term *more or less equal to* is suitable for finding entities in a database with the same or similar values of chosen indicators. In this example we are interested to find whether municipalities with similar values of three indicators (altitude, land area and population density) as the municipality *M* exist. The suggested procedure works in the following way: the user chooses one municipality (in our case municipality M) and relevant indicators. Consequently, the application finds values of selected indicators for the municipality *M* and creates triangular fuzzy sets (Fig. 3; B = C) for each indicator. The result is presented in Table 2 [7]. Table shows that no municipality has the same values as municipality *M*, but there are two municipalities (*M1* and *M2*) which are very similar to the municipality *M*. Other selected municipalities are less similar. More about this issue can be found in e.g. [7].

Municipality	Altitude [m]	Area [km^2]	Popul. Density [inhab/km^2]	Similarity
M1	125	15.78	68.32	0.8189
M2	125	15.22	70.64	0.7477
M3	120	18.05	69.53	0.4821
M4	116	18.13	75.66	0.3688
M5	126	13.39	75.65	02505
M6	132	14.77	75.15	0.1885
M7	132	18.71	72.73	0.1885

Values for the municipality *M* are: altitude: 123 m; land area: 16.14 km^2;
population density: 74 inhabitants / km^2

Table 2. Similarity.

6.4 FUZZY PREFERENCES IN QUERY CONDITIONS

Fuzzy preferences allow expressing that some elementary conditions are more relevant than others. For example, query for the selection of an appropriate hotel would be as follows: *select hotels where price is low, distance to the beach is low and distance to the city centre is low where low distance to the city centre has lower priority*.

Weights for the first two indicators are equal to 1 and the weight for the third indicator is equal to 0.5. The solution is calculated using the Kleene-Dienes implication [22]:

$$\mu(a^*) = \max(\mu(\mathrm{a}), 1 - \mathrm{w}) \qquad \textbf{(8)}$$

where a is a database attribute and $w \in [0,1]$ is an importance weight of an attribute a. Applying the Kleene-Dienes implication (8) and min t-norm (5) the query satisfaction degree is calculated in the following way:

$$\mu(t) = \min_{i=1,\ldots,n}(\max(\mu_i(a_i), 1 - w_i)) \qquad \textbf{(9)}$$

The membership degrees to the sets low price, low distance to the city centre and low distance to the beach as well as the solution are depicted in Table 3.

Hotels	μ(Low price)	μ(Low distance to beach)	μ(Low distance to city centre)	Matching Degree μ(t)
H1	1	1	1	1
H2	1	1	0.9	0.94
H3	0.9	0.9	0.7	0.7
H3	0.9	0.9	0.1	0.5
H4	0.9	0.9	0.5	0.5
H5	1	0.33	0.3	0.33

Table 3. Evaluation of appropriate hotels.

More about theory of fuzzy preferences can be found in e.g. [22] and about practical issues in [14].

7 CONCLUSION

The usefulness of the SQL for data selection from relational databases has been proven in many information systems. When sharp values in query criteria are required, the SQL is the optimal solution. However, when sharp constraints cannot appropriately describe the intent of a query, queries based on the fuzzy logic offer the solution.

It is obvious that processing of fuzzy queries introduces an additional computation burden due to the substantial amount of calculations concerning data [12]. We need to point out that this additional amount of the calculation burden is balanced with the additional valuable information mined from a database. The main goal of flexible queries is not to select more but better data for users.

The paper contributed with an overview of advantages of fuzzy queries in a legible way for data users, which are not familiar with mathematics of fuzzy logic and relational databases. Furthermore, paper discussed several issues that could be efficiently solved by fuzzy logic and fuzzy queries. To summarize, main characteristics of a fuzzy query approach are as follows:

- The query condition is expressed by the linguistic terms, so the meaning of the query is understandable at the first glance and therefore, query is easily readable and modifiable;
- Retrieved entities are ranked downward from the best to the worst according to the satisfaction degree of the overall query condition;
- Fuzzy queries reduce the risk of empty answers providing some data that partially meet the query condition when no data fully meets the query condition;
- An efficient solution for end-users could be obtained by integration of the several fuzzy query approaches.

8 REFERENCES

[1] BOSC, P., PIVERT, O. *SQLf Query Functionality on Top of a Regular Relational Database Management System*. In Knowledge Management in Fuzzy Databases. Physica Publisher, Heidelberg, pp. 171-190, 2000.

[2] BOSC, P., HADJALI, A., PIVERT, O., SMITS, G. Trimming Plethoric Answers to Fuzzy Queries: An Approach Based on Predicate Correlation. In *Computational Intelligence for Knowledge-Based Systems Design*. Springer, Heidelberg, volume 6178, pp. 595-604, 2010.

[3] BOSC, P. HADJALI, A., PIVERT, O. Empty versus overabundant answers to flexible relational queries. *Fuzzy Sets and Systems*. Vol. 159, No, 12, 2008, pp. 1450-1467.

[4] COX, E.: *Fuzzy modeling and genetic algorithms for data mining and exploration*. Morgan Kaufman, San Francisco, 2005. ISBN-10: 0121942759.

[5] DETYNIECKI M. *Fundamentals on Aggregation Operators*. In AGOP International Summer School on Aggregation Operators, Asturias, 2001. [cited 2013-7-7]. Available from:http://www-poleia.lip6.fr/~marcin/papers/Detynieck_AGOP_01.pdf

[6] GALINDO J., URRUTIA A., PIATTINI M. *Fuzzy databases: Modeling, Design and Implementation*. Idea Group Publishing Inc., Hershey, 2006. ISBN 1-59140-324-3

[7] HUDEC, M .Fuzzy database queries in official statistics: Perspective of using linguistic terms in query conditions. *Statistical Journal of the IAOS* (in press).

[8] HUDEC, M. Exploration of applying fuzzy logic for official statistics. *Conference of New Techniques and Technologies for Statistic*. European Commission, Brussels, 2011. [cited 2013-5-10]. Available from: http://www.cros-portal.eu/content/ps1-poster-9-ntts-2011-s1

[9] HUDEC, M. An Approach to Fuzzy Database Querying, Analysis and Realisation. *Computer Science and Information Systems*, Vol. 6, No. 2, 2009, pp. 127-140.

[10] HUDEC, M., SUDZINA, F. Construction of fuzzy sets and applying aggregation operators for fuzzy queries. *14th International Conference on Enterprise Information Systems*, Wroclaw, Poland, ICEIS, 2012, Proceedings volume 1, pp. 253-257.

[11] CHAMBERLIN, D., BOYCE R.SEQUEL: A Structured English Query Language. In *ACM SIGMOD Workshop on Data Description, Access and Control*, Ann Arbor, USA, 1974, pp. 249-264.

[12] KACPRZYK J., PASI G., VOJTÁŠ P., ZADROZNY S. Fuzzy querying: Issues and perspectives. *Kybernetika,* Vol. 36, No. 6, 2000, pp. 605-616.

[13] KLIR G., YUAN B. *Fuzzy sets and fuzzy logic, theory and applications*. Prentice Hall, New Jersey, 1995. ISBN-10: 0131011715.

[14] KĽÚČIK, M, HUDEC, M., JURIOVÁ, J. *Final report on the case study results on usage of IT tools and procedures developed for data collection (Soft computing tools for Official Statistics)*. Deliverable 5.2, Blue-ETS Project, 2012. [cited 2013-03-10]. Available from: http://www.blue-ets.istat.it/index.php?id=7&tx_wfqbe_pi1 [showpage][1]=2

[15] MCNEILL F.M., THRO E. *Fuzzy logic - a practical approach*. Academic Press Inc., Boston. 1994. ISBN-10: 0124859658

[16] RADOJEVIĆ, D. *Interpolative realization of Boolean algebra as a consistent frame for gradation and/or fuzziness.* In Forging New Frontiers: Fuzzy Pioneers II Studies in Fuzziness and Soft Computing. Springer-Verlag, Berlin / Heidelberg, pp. 295-318, 2008.

[17] SMITS, G., PIVERT, O., GIRAULT, T. (2013). ReqFlex: Fuzzy Queries for Everyone. *The VLDB Endowment*, Riva del Garda, Italy, Vol. 6, No. 12. pp. 1206-1209, 2013.

[18] URRUTIA, A., PAVESI, L. Extending the capabilities of database queries using fuzzy logic. *Proceedings of the Collecter-LatAm Conference*, Santiago, Chile, 2004. [cited 2011-5-10]. Available from: http://www.collecter.org/archives/2004_October/06.pdf.

[19] WANG ,T.C., LEE, H.D., CHEN, C.M. Intelligent Queries based on Fuzzy Set Theory and SQL. *Joint Conference on Information Science*, Salt Lake City, USA, 2007.pp. 1426-1433.

[20] ZADEH L.A. From computing with numbers to computing with words - from manipulation of measurements to manipulation of perceptions. *International Journal of Applied Mathematics and Computer Science*, Vol. 12, No. 3, 2002, pp. 307–324.

[21] ZADEH L. A. Fuzzy sets. *Information and Control*, Vol. 8, 1965, pp. 338–353.

[22] ZADROŻNY S., DE TRÉ G., DE CALUWE R. KACPRZYK J. *An overview of fuzzy approaches to flexible database querying*, In Fuzzy Information Processing in Databases, IGI Global, London, pp. 34-54, 2008.

[23] ZIMMERMANN, H-J. *Fuzzy Set Theory: And Its Applications*. Kluwer Academic Publishers, London, 2001. ISBN-10: 0792374355.