Available online at www.sciencedirect.com
SCIENCE DIRECT
Information and Software Technology 47 (2005) 357–364
INFORMATION AND SOFTWARE TECHNOLOGY
www.elsevier.com/locate/infsof

Implementation of fuzzy classification in relational databases using conventional SQL querying

Yauheni Veryha*

ABB Corporate Research, Integrated solutions, Wallstadter Str. 59, Ladenburg 68526, Germany

Received 7 December 2003
Available online 24 November 2004

Abstract

In this paper, a framework for implementing fuzzy classifications in information systems using conventional SQL querying is presented. The fuzzy classification and use of conventional SQL queries provide easy-to-use functionality for data extraction similar to the conventional non-fuzzy classification and SQL querying. The developed framework can be used as data mining tool in large information systems and easily integrated with conventional relational databases. The benefits of using the presented approach include more flexible data analysis and improvement of information presentation at the report generation phase. To confirm the theory, a prototype was developed based on the stored procedures and database extensions of Microsoft SQL Server 2000.

Keywords: Information system; Fuzzy classification; SQL query; Relational database; Data mining

1. Introduction

In typical data classification in information systems, there appear some types of uncertainty, for instance, when the boundaries of a class of objects are not sharply defined [1–3]. The most common, useful and widely accepted solution is the introduction of fuzzy sets [4–8]. Fuzzy sets provide mathematical meanings to the natural language statements and become an effective solution for dealing with uncertainty. In most practical data classification situations, more than one attribute or criteria must be considered simultaneously.

Criteria are usually measured with different scales. To combine different criteria into one general performance measure is not a simple task. The approach of fuzzy set theory with its membership functions is widely used to form a realistic description of the evaluation [9,10]. Different criteria with separate scales and optimization objectives can be combined into a joint response measure called the aggregated value of the membership [8,11,12].

A number of different schemes and tools for the implementation of fuzzy sets in database management systems have been proposed in recent years, such as fuzzy querying [4,13], fuzzy extension of SQL (Structured Query Language) [2,3,6,14] and fuzzy object oriented database schemes [1,7]. There are also commercial tools available on the market, like FIDE (Fuzzy Inference Development Environment for the development of fuzzy logic-based systems) of Aptronix Inc, FLINT (Fuzzy Logic Interfacing Toolkit that makes fuzzy logic technology and fuzzy rules available within a sophisticated programming environment) of Logic Programming Associates and FQS (Fuzzy Query System that allows the user to conduct database queries using the power and semantic flexibility of Fuzzy SQL) of Sonalysts Inc.

The largest majority of the above-mentioned fuzzy methods for data mining and commercial tools require to change the conventional relational database structure or to add special features to the database management tools, for example, to modify the functionality of the conventional SQL [2,3,6] or to extract fuzzy functionality into a separate application as it was implemented in FQS of Sonalysts Inc.

Despite large importance of data mining using fuzzy technology, fuzzy methods are not widely used in

* Corresponding author. Tel.: +49 6203 716 024.
E-mail address: yauheni.veryha@de.abb.com.

0950-5849/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2004.09.007

relational database systems in practice and the main reason is that most of database system owners and users do not want to switch to fuzzy database structures or use third-party applications. As a result, they usually do not use fuzzy querying in their information systems. To attract more attention to fuzzy applications from database system owners and users we propose to use a framework based on the fuzzy classification [12,15] and conventional SQL queries. The main benefit of this framework is that there is no need to modify the functionality of the conventional relational databases and SQL. All manipulations can be done as an extension of the database schema by applying fuzzy data classification. Benefits of using fuzzy sets and fuzzy classification in data mining, like user-friendly data presentation, precision of the data classification, use of linguistic variables instead of numeric values and easy-to-use facilities for querying the extended database schema become available for users of relational databases as well.

2. Conventional data classification in relational databases

This paper is concentrated on comparison of implementation in relational databases of conventional data classification and fuzzy data classification using atomic values of attributes.

To show the difference in the data presentation before and after use of conventional data classification, the following simple example can be considered. An exemplary database includes OFFERS table that contains data about suppliers (Supplier column), materials (Material column), material quality (Quality column), and delivery delay (Delay column). An exemplary data relation is presented in Table 1. The most common method of querying relational database tables is through SQL. To query the presented data and produce a list of 'good' suppliers, one can use a typical SQL query, such as

Table 1
OFFERS table

Material	Supplier	Quality	Delay
802.025	BAW	sufficient	8
802.025	DG	average	5
802.025	KBA	sufficient	7
802.025	MD	sufficient	9
802.025	MTX	high	2
802.025	MAM	low	7
840.024	KBA	low	9
840.024	ZT	average	2
840.024	MTX	high	4
840.024	MAM	average	6
809.200	MD	average	4
809.200	DG	high	8
809.200	KBA	sufficient	3
809.200	ZT	high	9

> SELECT Supplier, Material, Quality, Delay FROM Offers WHERE (Quality = "high" or Quality = "average") and (Delay < 5),

assuming that 'good' suppliers are those who provide a material of 'high' or 'average' quality with delay of less than five hours. For small data tables, this approach is acceptable. However, if the number of attributes is more than five and ranges of data in columns are significantly larger, queries become much more complex. To simplify data mining in large databases, one may use conventional data classification [10].

For simplicity, we will go on with introduction of conventional data classification for simple OFFERS table. To define classes, we introduce atomic values for Delay and Quality attributes. We define 'acceptable' and 'not-acceptable' atomic values for Delay attribute. The atomic value of 'acceptable' is assigned to the interval of (1–5) and the atomic value of 'not-acceptable' is assigned to the interval of (6–10). The atomic value of 'good' is assigned to 'high' and 'average' Quality attribute values. Similarly, 'bad' atomic value is assigned to 'sufficient' and 'low' Quality attribute values. As soon as the definition of atomic values is finished, it is easy to define all classes for suppliers:

> C1–Extend Relationship Class with atomic values ('acceptable', 'good');
> C2–Complain About Delay Class with atomic values ('not-acceptable', 'good');
> C3–Improve Quality Class with atomic values ('acceptable', 'bad');
> C4–Evaluate Relationship Class with atomic values ('not-acceptable', 'bad').

A distribution of suppliers from OFFERS table among the introduced classes is graphically presented in Fig. 1.

To find all members of 'Complain About Delay' class C2 (see Fig. 1), one can query initial OFFERS table with simple SQL query

> SELECT Supplier, Material, Quality, Delay FROM Offers WHERE (Quality = "high" or Quality = "average") and (Delay > 5).

The conventional non-fuzzy data classification is easy to implement using native SQL queries. The main problem of the presented conventional data classification is that the information about suppliers is not sufficient, for example, some suppliers, like DG and KBA (see Fig. 1), belong to different classes and it is hard to distinguish to which class the given supplier belongs more and to which class less because of a precise boundary definition of atomic values. Thus, the generated reports based on this simple data classification are not precise enough. To improve

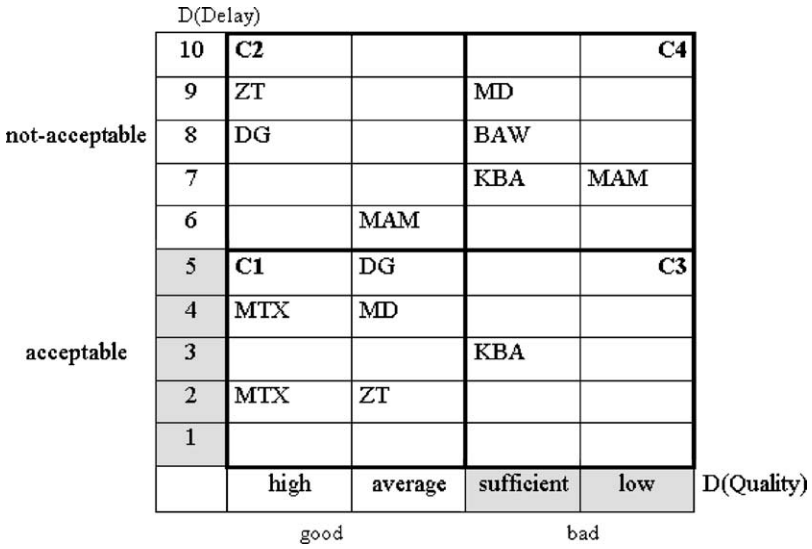

Fig. 1. Conventional non-fuzzy classification of suppliers.

a precision of data classification, we will use fuzzy data classification methodology of [15] as most widely used in practice. Other fuzzy data classification methodologies [8,9] could be also potentially used in this work.

3. Fuzzy data classification in relational databases

To get more detailed information about supplier membership in classes, one can introduce fuzzy classification of data [11,15] and linguistic variables [15]. As an example, we can consider Delay attribute as a linguistic variable. Area of definition of the linguistic variable is domain *D*(Delay). Similarly to the assignment of the atomic value, Delay linguistic variable possesses a set of terms $T(\text{Delay}) = \{\text{acceptable, not-acceptable}\}$ with the verbal terms 'accep-acceptable' and 'not-acceptable' which define appropriate equivalence classes (1–5) and (6–10).

The most important feature of linguistic variables is that every term of a linguistic variable represents a fuzzy set. The membership function of the fuzzy set is defined over the domain of the corresponding attribute. For instance, in Fig. 2, the delay of five days is at the same time acceptable and not acceptable; in both cases the membership function μ has the value 0.5 (i.e. $\mu_{\text{acceptable}} = 0.5$ and $\mu_{\text{not-acceptable}} = 0.5$). The membership of an object in a specific class can be calculated by an aggregation over all terms of the linguistic variables that define the class. Terms 'acceptable' and 'good', for

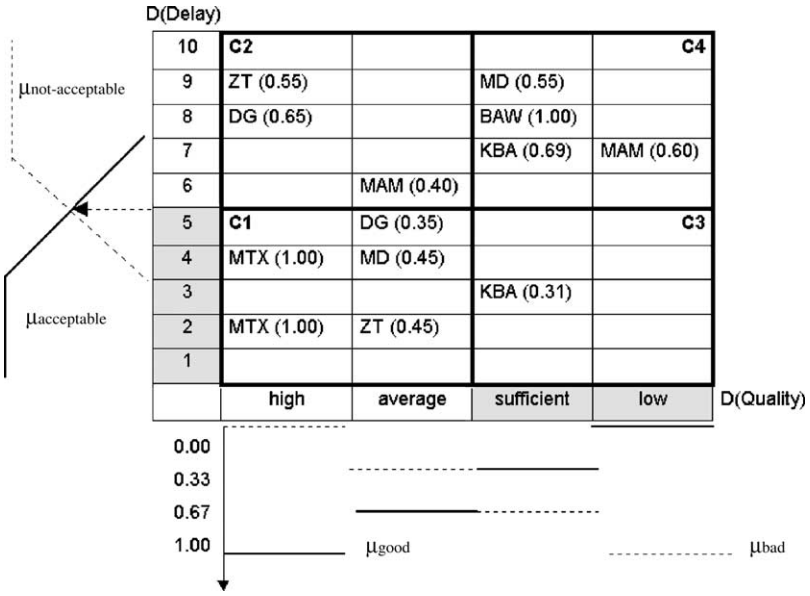

Fig. 2. Fuzzy classification of data with appropriate membership functions.

instance, describe class C1. The membership is a conjunction of the corresponding values of the membership functions $\mu_{\text{acceptable}}$ and μ_{good}.

There exist a number of operators that can be used to calculate conjunctions of membership function values [11,12]. For example, one can apply the γ-operator which is used as a 'compensatory AND' and was empirically tested in [11]. The membership $\mu_{\tilde{A}_i,\text{comp}}(x)$ of x object with m linguistic variables to the given classes can be calculated based on the following equation [12]

$$\mu_{\tilde{A}_i,\text{comp}}(x) = \left(\prod_{i=1}^{m}\mu_i(x)\right)^{(1-\gamma)}\left(1-\prod_{i=1}^{m}(1-\mu_i(x))\right)^{\gamma}, \quad x \in X,\ 0 \le \gamma \le 1$$

where γ is control parameter with default value of 0.5 [12], $\mu_i(x)$ is the membership value of x object to a particular linguistic variable, m is the number of linguistic variables. Based on γ-operator, values of supplier membership are calculated and presented in Fig. 2.

The main benefit of fuzzy data classification becomes clear when one compares the presentation of C1, C2, C2 and C4 classes in Figs. 1 and 2. As it is shown in Fig. 2 with fuzzy data classification, one cannot only determine members of the given class but also compare the level of the membership to the particular classes. For example, based on the fuzzy classification, one can say that MAM supplier belongs more to C4 class than to C2 class (see Fig. 2). Membership values can be very useful for searching the best suppliers in the given class. They become important criteria for evaluating suppliers; one only needs to compare membership values of suppliers of a particular class. For example, if two suppliers offer the same quality and delivery delay for the given material, one can compare their membership values. The supplier with a higher membership value to C1 class can be considered to be better.

Membership value is similar to the deviation in the statistics and shows the distribution of the supplier membership to different classes. For example, to get suppliers that belong to C2 class (see Fig. 2), one could get data output as database view (see Table 2) after fuzzy classification. Using given data view, one can say that DG supplier belongs to C2 class more than MAM and ZT suppliers because of higher value of membership function.

Table 2
Database view for C2 class members

Supplier	Class	Quality	Delay	Membership Value
MAM	C2	good	not-acceptable	0.4
ZT	C2	good	not-acceptable	0.55
DG	C2	good	not-acceptable	0.65

The use of fuzzy classification provides higher accuracy of data presentation and gives a more precise description of data in generated reports. The drawback is that membership values of suppliers have to be calculated with rather complicated mathematical formulas making SQL querying process a bit complicated.

There are different approaches for implementing the presented fuzzy classification in practice. One of the approaches is to use fuzzy querying languages as an extension of the conventional SQL [2,3,9]. However, an introduction of new clauses into the SQL syntax changes conventional SQL functionality.

To attract more attention to fuzzy applications we propose to use only conventional SQL functionality for querying data with fuzzy classification. Database users do not need to add any new clauses to the SQL syntax and can continue using a conventional syntax of SQL, as if there is no fuzzy classification behind the data. This approach provides an added value to the conventional SQL and becomes very attractive for owners and users of existing relational databases.

4. SQL for fuzzy classified data querying

The goal of SQL querying of data with fuzzy classification is to provide database views and/or reports of fuzzy classified data, similar to that presented in Table 2, using easy-to-use and familiar SQL queries. To implement the functionality of SQL for fuzzy classified data querying in relational databases, the best solution is to develop an interpreter as stored procedure that will translate conventional SQL commands into native SQL queries of a particular database. Users can formulate SQL queries with well-defined and familiar terms and do not need to know definitions of equivalence classes in details or fuzzy classification details behind the data. In addition, all complex mathematical formulas for calculating membership functions are hidden from users; the interpreter will take care for them.

In the developed prototype, the following basic functionality of SQL for fuzzy classified data was implemented

select ⟨*Object*⟩
[**into**] ⟨*View*⟩
from ⟨*Relation*⟩
[**where**] ⟨Classification_condition⟩

For example, SQL query

select Supplier **into** Suppliers **from** Offers

performs a classification (see Fig. 2) of all suppliers from OFFERS table (see Table 1) into SUPPLIERS view. SUPPLIERS view content is presented in Table 3.

Table 3
Suppliers view

Supplier	Class	Quality	Delay	Membership Value
DG	C1	good	acceptable	0.35
MD	C1	good	acceptable	0.45
ZT	C1	good	acceptable	0.45
MTX	C1	good	acceptable	1.00
MAM	C2	good	not-acceptable	0.4
ZT	C2	good	not-acceptable	0.55
DG	C2	good	not-acceptable	0.65
KBA	C3	bad	acceptable	0.31
MD	C4	bad	not-acceptable	0.55
BAW	C4	bad	not-acceptable	1.00
MAM	C4	bad	not-acceptable	0.60
KBA	C4	bad	not-acceptable	0.69

Data from SUPPLIERS view can be again queried using conventional SQL for further data analysis and report generation. Users can also use SQL query

select Supplier **from** Offers

without **into** clause to output the same results on the screen. An extended SQL query

select Supplier **from** Offers **where** (Delay = "not-acceptable" and Quality = "good")

will generate the data presented in Table 2. SQL for fuzzy classified data fully complies with the conventional SQL. The only difference is that one needs to use an interpreter that will translate above presented SQL queries into the native SQL queries of the given database management system, for example, into Transact-SQL of the Microsoft SQL Server 2000. To implement the SQL for querying of fuzzy classified data, appropriate database management system extensions will have to be added.

5. Fuzzy classification and querying implementation in relational databases

Implementation scheme of SQL for fuzzy classified data querying includes the following steps:

1. Design of database tables or views to query them later using SQL for fuzzy classified data. This step has to be carried out by database owners.
2. Design of database extensions (additional tables that contain linguistic variables, membership values and descriptions of atomic values). This step should be carried out by an expert in the given application area. Database extensions can be generated automatically (additional programming may be required in this case).
3. Design and implementation of interpreter for SQL transformation into native SQL for the given relational database management system using lexical and syntactical analysis of queries. This step should be carried out by software developer that will develop an interpreter in the form of the stored procedure for the given database management system.
4. Generation of database reports and views using SQL querying of fuzzy classified data formed on Steps 1 and 2.

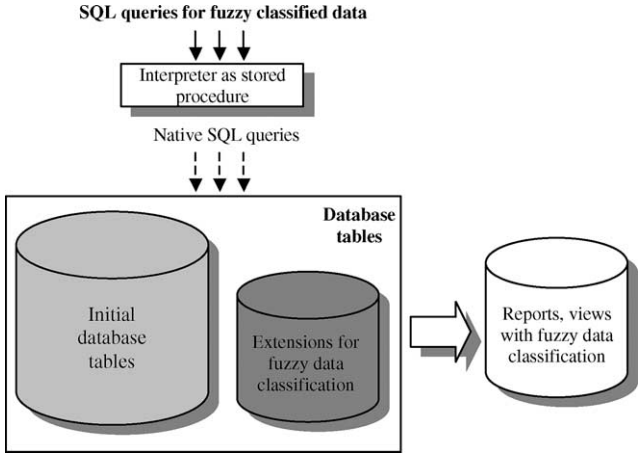

Fig. 3. Scheme of fuzzy classification framework implementation in relational database.

The scheme of the developed framework implementation in relational database is shown in Fig. 3.

Processing of SQL queries by the interpreter includes two main stages:

- Syntactical analysis of SQL string clauses for fuzzy classified data querying;
- Execution of native SQL sub-queries (grouping of objects into classes, calculation of 'compensatory AND' membership of objects to the classes and calculation of normalized membership of objects to classes) with parameters from SQL clauses for fuzzy classified data querying.

The developed fuzzy classification and querying framework implementation are illustrated by simple database example that is similar to that presented in the beginning of the article. Microsoft SQL server 2000 database management system was chosen for fuzzy classification implementation because of its large popularity among database users. The embedded Transact-SQL language of the Microsoft SQL Server 2000 provided all required functionality to develop interpreter as the stored procedure.

The interpreter was developed with the assumption that similar interpreter could be further developed for other platforms, like Oracle or SyBase, using their embedded SQL versions. The interpreter was realized as a stored procedure on the Microsoft SQL Server 2000 that could

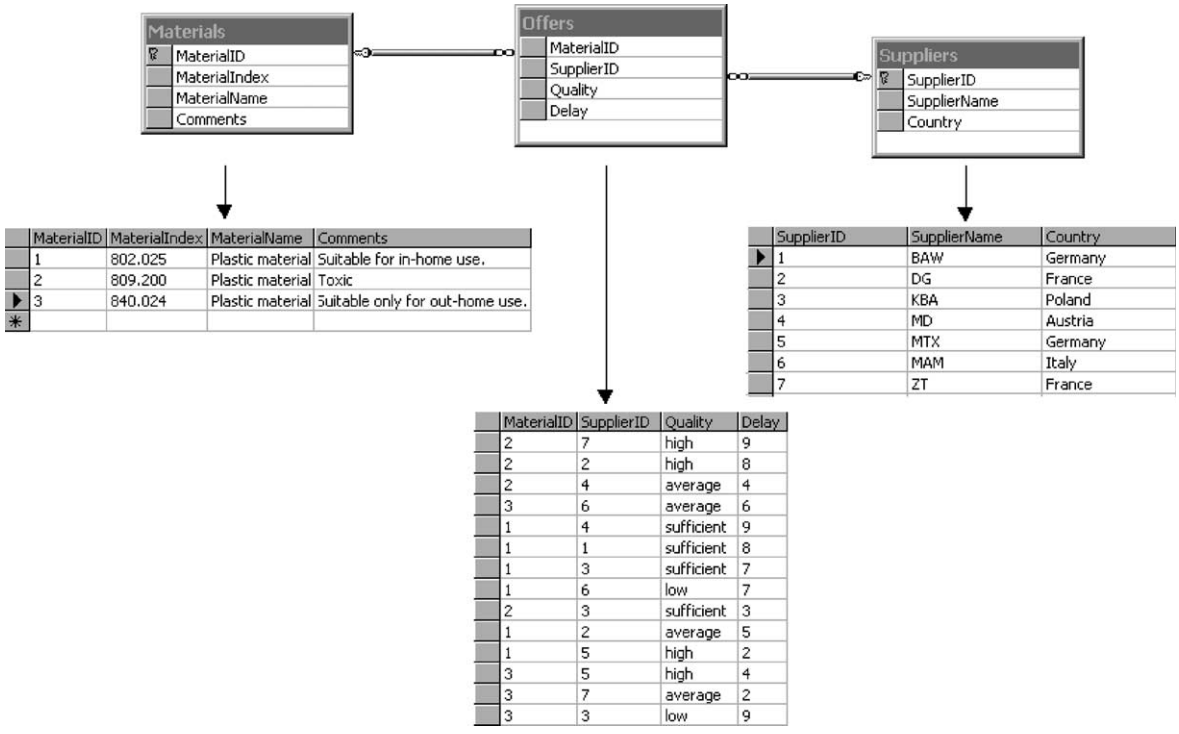

Fig. 4. Database schema example with content and relationships.

translate SQL queries for fuzzy classified data into normal Transact-SQL queries. The syntactical analysis (typical extraction of clauses from query string) of SQL queries for fuzzy classified data was implemented using Transact-SQL functions (SUBSTRING, LTRIM, RTRIM, LEFT and PATINDEX) for strings and embedded Microsoft SQL Server 2000 stored procedure (SP_EXECUTESQL). The database scheme of the above-presented example in the normalized form is presented in Fig. 4.

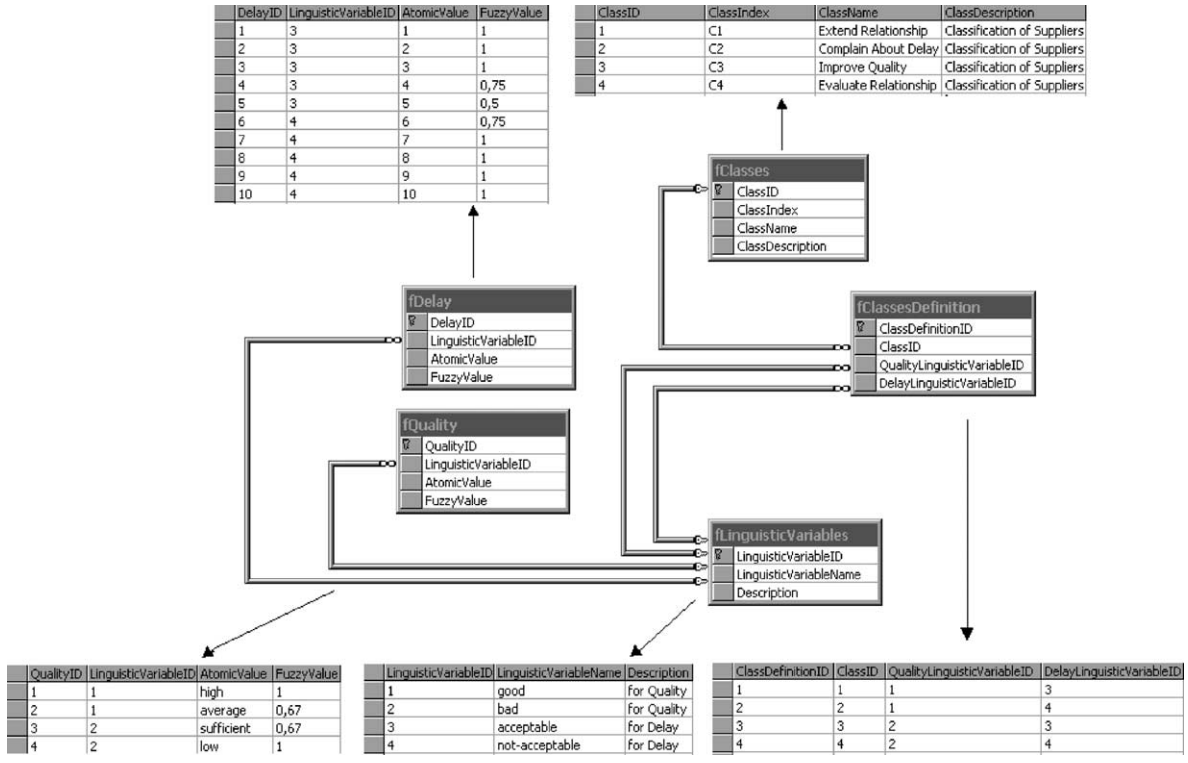

Fig. 5. Database extension tables and relationships for fuzzy classification.

```
SELECT i.Class, i.Supplier, i.Quality, i.[Delay], CONVERT (char, i.Total/(SELECT j.TotalSum FROM (SELECT h.Supplier
AS Supplier, SUM(h.Total) AS TotalSum FROM (SELECT g.Class, g.[Supplier Name] as Supplier, (SELECT LinguisticVariableI
FROM fLinguisticVariables WHERE LinguisticVariableID = AVG(g.Quality)) AS Quality, (SELECT LinguisticVariableName FRO
fLinguisticVariables WHERE LinguisticVariableID = AVG(g.[Delivery Delay])) AS [Delay], SUM(g.[Fuzzy Compensatory Valu
AS Total FROM (SELECT a.OfferID as [ID], (SELECT ClassIndex FROM fClasses WHERE ClassID = d.ClassID) AS Class,
b.SupplierName as [Supplier Name], c.MaterialIndex AS Material, (SELECT LinguisticVariableID FROM fLinguisticVariables
WHERE LinguisticVariableID = e.LinguisticVariableID) AS Quality, (SELECT LinguisticVariableID FROM fLinguisticVariable
LinguisticVariableID = f.LinguisticVariableID) AS [Delivery Delay], e.FuzzyValue AS [Quality Fuzzy Value], f.FuzzyVal
AS [Delay Fuzzy Value], (SELECT POWER(e.FuzzyValue*f.FuzzyValue, 0.5)*POWER((1-(1-e.FuzzyValue)*(1- f.FuzzyValue)), 0
AS [Fuzzy Compensatory Value] FROM Offers a, Suppliers b, Materials c, fClassesDefinition d, fQuality e, fDelay f
WHERE (a.SupplierID = b.SupplierID AND a.MaterialID = c.MaterialID AND a.Quality = e.AtomicValue
 AND a.[Delay]= f.AtomicValue AND d.QualityLinguisticVariableID = e.LinguisticVariableID AND
d.DelayLinguisticVariableID = f.LinguisticVariableID AND d.QualityLinguisticVariableID = e.LinguisticVariableID AND
d.DelayLinguisticVariableID = f.LinguisticVariableID)) g GROUP BY g.[Supplier Name], g.Class) h GROUP BY h.Supplier)
WHERE j.Supplier=i.Supplier), 0) AS [Fuzzy Membership]FROM (SELECT g.Class, g.[Supplier Name] AS Supplier, (SELECT Li
LinguisticVariableID = AVG(g.Quality)) AS Quality, (SELECT LinguisticVariableName FROM fLinguisticVariables WHERE
LinguisticVariableID = AVG(g.[Delivery Delay])) AS [Delay], SUM(g.[Fuzzy Compensatory Value]) AS Total FROM (SELECT a
(SELECT ClassIndex FROM fClasses WHERE ClassID = d.ClassID) AS Class, b.SupplierName as [Supplier Name],
c.MaterialIndex AS Material, (SELECT LinguisticVariableID FROM fLinguisticVariables WHERE
LinguisticVariableID = e.LinguisticVariableID) AS Quality, (SELECT LinguisticVariableID FROM fLinguisticVariables WHE
```

Class	Supplier	Quality	Delay	Fuzzy Membership
C1	DG	good	acceptable	0.345931
C1	MD	good	acceptable	0.453412
C1	MTX	good	acceptable	1
C1	ZT	good	acceptable	0.450107
C2	DG	good	not-acceptable	0.654069
C2	MAM	good	not-acceptable	0.404408
C2	ZT	good	not-acceptable	0.549893
C3	KBA	bad	acceptable	0.310396
C4	BAW	bad	not-acceptable	1
C4	KBA	bad	not-acceptable	0.689604
C4	MAM	bad	not-acceptable	0.595592
C4	MD	bad	not-acceptable	0.546588

Fig. 6. Screen of native Transact-SQL querying of fuzzy classified data.

To extend the current database scheme for working with the fuzzy classified data, the following database schema extensions (i.e. tables and relationships), were provided for fuzzy classification, as shown in Fig. 5. In addition to initial OFFERS, MATERIALS and SUPPLIERS tables (see Fig. 4), the following classifications tables were added (see Fig. 5): fDELAY, fQUALITY, fCLASSES, fCLASSES-DEFINITION and fLINGUISTICVARIABLES tables.

Additional tables include the definition of classes, linguistic variables and their dependencies. To query the given database and receive a fuzzy classified data report similar to that shown in Table 3, one may have to execute very large native Transact-SQL query without interpreter, as shown in Fig. 6. Instead of this, one can execute a simple SQL query with the help of the developed interpreter and receive the same results, as it is shown in Fig. 7.

The presented example shows that the use of SQL with the developed interpreter is much easier and user-friendlier than the direct use of native Transact-SQL. The drawback of the presented solution is that it is not flexible enough for possible database structure changes and may require additional programming to implement other operators (currently, it supports only one ‘compen-compensatory AND’ operator for computing membership function).

```
Execute SQLinterpreter 'SELECT Supplier FROM Offers'
```

Class	Supplier	Quality	Delay	Fuzzy Membership
C1	MTX	good	acceptable	1
C1	MD	good	acceptable	0.453412
C1	ZT	good	acceptable	0.450107
C1	DG	good	acceptable	0.345931
C2	DG	good	not-acceptable	0.654069
C2	ZT	good	not-acceptable	0.549893
C2	MAM	good	not-acceptable	0.404408
C3	KBA	bad	acceptable	0.310396
C4	BAW	bad	not-acceptable	1
C4	KBA	bad	not-acceptable	0.689604
C4	MAM	bad	not-acceptable	0.595592
C4	MD	bad	not-acceptable	0.546588

```
(12 row(s) affected)
```

Fig. 7. Screen of fuzzy classified data querying using SQL and interpreter as stored procedure.

6. Conclusions

Fuzzy classification framework for relational databases using SQL querying has been presented. The fuzzy classification and use of conventional SQL queries provide easy-to-use functionality for data extraction similar to the conventional non-fuzzy classification and SQL querying. The main benefits of using fuzzy classification and SQL querying are the following:

1. Good integration with conventional databases and high flexibility for data analysis.
2. Data presentation for report generation phase with linguistic variables and fuzzy values for more descriptive queries in data analysis phase.
3. Users or database administrators do not have to change underlying databases, which are, in practice, very large. With the developed approach, one operates on the database schema level by defining appropriate fuzzy classifications with linguistic variables.
4. Fuzzy classification scheme provides additional data security features because of the introduction of an additional view-based data layer that hides numerical values from users.

For the implementation of fuzzy classification framework, the prototype (data mining tool) based on Microsoft SQL Server 2000 was developed. The use of the interpreter for SQL queries of fuzzy classified data on the server side as the stored procedure gave the following benefits:

- simple querying using conventional SQL,
- close integration with the database and easy distribution,
- availability of stored procedure on the server for all users of the database.

Future works will include testing of the developed framework on the large database from industry and comparison of framework implementation with different operators for calculating conjunctions of membership function values in fuzzy classifications.

References

[1] G. Borgodna, A. Leporati, D. Lucarella, G. Pasi, The fuzzy object-oriented database management system, Studies in Fuzziness and Soft Computing 53 (2000) 209–236.
[2] J. Kacprzyk, S. Zadrozny, Data mining via fuzzy querying over the internet, Studies in Fuzziness and Soft Computing 39 (2000) 211–233.
[3] J. Kacprzyk, S. Zadrozny, On combining intelligent querying and data mining using fuzzy logic concepts, Studies in Fuzziness and Soft Computing 53 (2000) 67–84.
[4] M. Bellma, N. Vojdani, Fuzzy prototypes for fuzzy data mining, Studies in Fuzziness and Soft Computing 39 (2000) 175–286.
[5] I. Blanco, J. Cubero, O. Pons, A. Vila, An implementation for fuzzy deductive relational databases, Studies in Fuzziness and Soft Computing 53 (2000) 183–208.
[6] P. Bosc, O. Pivert, SQLf query functionality on top of a regular relational database management system, Studies in Fuzziness and Soft Computing 39 (2000) 171–191.
[7] D. Dubois, M. Nakata, H. Prade, Extended divisions for flexible queries in relational databases, Studies in Fuzziness and Soft Computing 39 (2000) 105–121.
[8] L. Zadeh, Knowledge representation in fuzzy logic, IEEE Transactions on Knowledge and Data Engineering 1 (1989) 89–100.
[9] G. Chen, Fuzzy Logic in Data Modeling—Semantics, Constraints and Database Design, Kluwer, London, 1998.
[10] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, New York, 2001.
[11] H. Zimmermann, Fuzzy Set Theory—and Its Applications, Kluwer, London, 1992.
[12] H. Zimmermann, P. Zysno, Latent connectives in human decision making, Fuzzy Sets and Systems 4 (1) (1980) 37–51.
[13] S. Abdennadher, H. Schuetz, Flexible Query Language, Proceedings of Flexible Query Answering System Conference, Roskilde, Denmark, 1998 pp. 1–14.
[14] S. Zadrozny, J. Kacprzyk, Implementing fuzzy querying via the internet/www: java applets active X controls and cookies, Proceedings of Flexible Query Answering System Conference, Roskilde, Denmark 1998; 382–392.
[15] G. Schindler, Fuzzy Datenanalyse durch kontextbasierte Datenbankanfragen, DUV/Gabler, Wiesbaden, Germany, 1998.