DOI:10.2298/CSIS100102010S

Towards the Methodology for Development of Fuzzy Relational Database Applications

Srđan Škrbić[1], Miloš Racković[1], and Aleksandar Takači[2]

[1] Faculty of Science, Trg Dositeja Obradovića 3,
21000 Novi Sad, Serbia
{shkrba, rackovic}@uns.ac.rs
[2] Faculty of Technology, Bulevar Cara Lazara 1,
21000 Novi Sad, Serbia
stakaci@tf.uns.ac.rs

Abstract. In this paper we examine the possibilities to extend the relational data model with the mechanisms that can handle imprecise, uncertain and inconsistent attribute values using fuzzy logic and fuzzy sets. We present a fuzzy relational data model which we use for fuzzy knowledge representation in relational databases that guarantees the model in 3rd normal form. We also describe the CASE tool for the fuzzy database model development which is apparently the first implementation of such a CASE tool. In this sense, this paper presents a leap forward towards the specification of a methodology for fuzzy relational database applications development.

Keywords: fuzzy database, fuzzy-relational data model, fuzzy-relational CASE tool

1. Introduction

Relational model's disability to model uncertain and incomplete data can be viewed as its disadvantage in some applications. The idea to use fuzzy sets and fuzzy logic to extend existing database models to include these capabilities has been utilized since the 1980s. Although this area has been researched for a long time, concrete implementations are rare. Methodologies for fuzzy-relational database applications development are nonexistent.

However, literature contains references to several models of fuzzy knowledge representation in relational databases, as well as different approaches to fuzzy-relational database querying mechanisms. Some good overviews of these research areas can be found in [1,2,3].

In the next section of this paper we give a detailed description of a number of references that describe research in the use of fuzzy logic in relational databases. This overview stretches from the very beginnings of this idea, to the most recent approaches. Third section contains a description of our approach to fuzzy-relational data modelling. We give a detailed description of our fuzzy meta model and investigate its theoretical value. Fourth section contains a description of the fuzzy-relational data modelling CASE tool, the first of its kind. We compare our approach to some previous approaches in the fifth section. At the end, we give the conclusion.

2. Related Work

One of the early works, the Buckles-Petry model [4] is the first model that introduces similarity relations in the relational model. This paper gives a structure for representing inexact information in the form of a relational database. The structure differs from ordinary relational databases in two important aspects: components of tuples need not be single values and a similarity relation is required for each domain set of the database. Zvieli and Chen [5] offered a first approach to incorporate fuzzy logic in the ER (Entity-Relationship) model. Their model allows fuzzy attributes in entities and relationships. It defines three levels of fuzziness. At the first level, entity sets, relationships and attribute sets may be fuzzy, i.e. they have a membership degree to the model. The second level is related to the fuzzy occurrences of entities and relationships, and on notion which instances belong to the entity or relationship with different membership degrees. Finally, the third level is concerned with the fuzzy values of attributes of special entities and relationships.

Fuzzy functional dependencies and fuzzy normal forms, as well as algorithms for dependency preserving and lossless join decompositions of fuzzy relations in specific fuzzy extensions of relational model are investigated in [6,7].

Umano and Fukami proposed FREEDOM-O, a fuzzy database system which is an extension of relational model of data [8]. This system supports a fuzzy data model, and querying. It is the first implementation of a fuzzy database system. After that result, other researchers have proposed similar fuzzy extensions to the relational model in such as in [9,10,11,12].

Another serious attempt to implement a fuzzy database system is given in [13,14]. Authors propose fuzzy extensions of the classical SQL and implement a system that allows using fuzzy conditions in place of Boolean ones.

The GEFRED (Generalized Model of Fuzzy Relational Databases) model [15] is a probabilistic model that refers to generalized fuzzy domains and admits the possibility distribution in domains. This is a fuzzy relational database model that has representation capabilities for a wide range of fuzzy information. In addition, it describes a flexible way to handle this information. Also, it contains the notion of unknown, undefined and null values. The GEFRED model experienced subsequent expansions, such as [16,17,18,19].

Chen and Kerre [20,21] introduced the fuzzy extension of several major EER (Extended Entity-Relationship) concepts. Fuzzy logic was applied to some of the basic EER concepts connected to the notion of subclass and super class. Chaudhry, Moyne and Rundensteiner [22] proposed a method for designing fuzzy relational databases following the extension of the ER model of Zvieli and Chen. They also proposed a design methodology for FRDBs (Fuzzy Relational Databases), which contains extensions for representing the imprecision of data in the ER model, and a set of steps for the derivation of a FRDB from this extended ER model.

Galindo, Urrutia and Piattini [23] describe a way to use the fuzzy EER model to model the database and represent modelled fuzzy knowledge using relational database in detail. This work gives insight into some new semantic aspects and extends EER model with fuzzy capabilities. The model is called FuzzyEER model. Also, a way to translate FuzzyEER model to the FIRST-2, a database schema that allows representation of fuzzy attributes in relational databases is given. The FIRST-2 schema introduces a concept of Fuzzy Meta-knowledge Base (FMB). For each attribute type, it defines how to represent values and what information about them has to be stored in the FMB. In addition, in this work, authors introduce and describe specification and implementation of the FSQL - an SQL language with fuzzy capabilities in great detail. This language is an extension of the SQL language that allows users to write flexible conditions in queries, using all extensions defined by the FuzzyEER model.

We conclude that the current state of the art in this area includes mature fuzzy EER model extensions that describe a wide range of modelling concepts for full flavoured fuzzy database modelling. These conceptual models are supported by robust models for fuzzy data representation in relational databases, such as the FIRST-2. The possibilities to translate conceptual models to the relational-based ones are also studied in detail. In addition, the FSQL is the first implementation of fuzzy database query language that incorporates the majority of fuzzy logic concepts.

In [24,25,26] authors have studied the possibilities to extend the relational model with the fuzzy logic capabilities. The subject was elaborated in [27,28], where a detailed model of Fuzzy Relational Databases (FRDB) was given. One of the main features of the model is that it allows any fuzzy subset of the domain to be the attribute value which was not the case in previous FRDB models.

Moreover, using the concept of the Generalized Priority Constraint Satisfaction Problem (GPFCSP) from [29] and [30], authors have found a way to introduce priority queries into FRDB, which resulted in the PFSQL query language [31,32]. In [33] authors introduce similarity relations on the fuzzy domain which are used to evaluate the FRDB conditions. The PFSQL allows the conditions in the WHERE clause of the query to have different priority i.e. importance degree. It is one of the first languages with such capabilities. The GPFCSP gives the theoretical background for the implementation of priority queries.

In this paper, we focus on an innovative fuzzy relational data model designed to include a more detailed structure and allow better performance. We analyze concordance of this model to theoretical concepts of relational model, especially normal forms. In addition, we describe the CASE tool that allows development of fuzzy databases using our model. This appears to be the first implementation of such a CASE tool. Proposed fuzzy relational data model and the CASE tool that supports it give a good foundation for development of the database part in fuzzy relational database (FRDB) applications. We discuss the possibilities to define a complete methodology for FRDB applications development and describe steps that need to be made in that direction.

3. Fuzzy Relational Data Model

In this section we describe our relational model extensions that constitute our variant of fuzzy relational data model. Our model stores crisp values in the same way as relational model does, while, for fuzzy values, we define fuzzy meta data model. In addition, here we provide an insight into the process of transformation of an example of the classical relational model with fuzzy attributes to the corresponding fuzzy relational data model.

If we wish to store a fuzzy value, we need to find a way to store data about its characteristic function. Theoretically, in this way, we could store any fuzzy value. But, in practice, only a handful of characteristic functions are in use. Let us name them fuzzy data types from this aspect. That is why we cover only a limited number of fuzzy data types and obtain an efficient and relatively simple data model.

An example relational model shown at Fig. 1 contains tables Worker and Car as well as an intersection table Uses that we use to model this many- to-many relationship. Tables Worker and Car have two fuzzy attributes each.

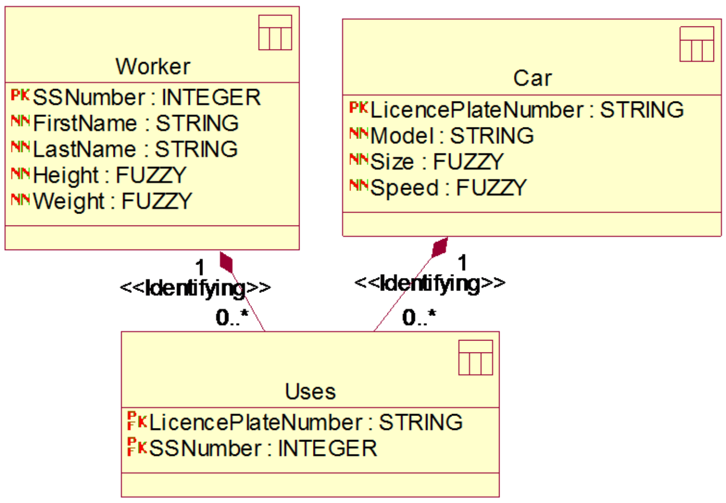


Fig. 1. Example fuzzy data model

The corresponding fuzzy-relational data model is shown at Fig. 2. The tables Worker, Car and Uses are shown at the top of the figure. They are the same as they were before except for the data type of fuzzy columns. In this model, they are of type INTEGER. Moreover, they became foreign keys that originate from the attribute *ValueID* in the table FuzzyValue. In order to represent these fuzzy values in the database, we extend this model with some additional tables that make the fuzzy meta data model.

The table IsFuzzy simply stores the information whether an attribute is fuzzy or not. All attribute names in the database are stored here, and beside the table and the attribute name (attributes *TableName* and *AttributeName*), the information whether the attribute is fuzzy (value of the attribute *IsFuzzy* is 1) or not (value of the attribute *IsFuzzy* is 0) is present.

The table FuzzyValue represents a connection between the fuzzy data model and the fuzzy data meta model. Every fuzzy value in every table is a foreign key that references attribute *ValueID* - the primary key of the table FuzzyValue. Thus, we have one record in the table FuzzyValue for every record with the fuzzy value in the database. The attribute *Code* is a foreign key from the table FuzzyType. This table stores the name of every possible type of fuzzy value allowed in the model.

These types are as follows:

- interval - fuzzy value is an interval,
- triangle - fuzzy value is a triangular fuzzy number,
- trapezoid - fuzzy value is a trapezoidal fuzzy number,
- general - fuzzy value is a general fuzzy number given by points,
- fuzzyShoulder - fuzzy value is a fuzzy shoulder,
- linguisticLabel - fuzzy value is a linguistic label,
- crisp - fuzzy value is actually a crisp value.

For every value in this list, there is a separate table in the meta model that stores data for all fuzzy values of specific fuzzy type. Every one of these tables has the attribute *ValueID*, foreign key from the table FuzzyValue. In this way, the value for the specific fuzzy attribute is stored in one of these tables depending on its type.

The attribute *ForValueID* in the table FuzzyValue is a foreign key that represents a recursive relationship and references the primary key of the FuzzyValue table. This attribute is used to represent linguistic labels. It has a value different than null if the type of the attribute that it represents is linguisticLabel. As mentioned before, linguistic labels only represent names for previously defined fuzzy values. In this fashion, if an attribute is a linguistic label, then its name is stored in the table LinguisticLabel. In this case, the attribute *ForValueID* has the value of *ValueID* of a fuzzy value that this linguistic label represents. We conclude that, in order to represent a linguistic label, two records in the table FuzzyValue are needed.

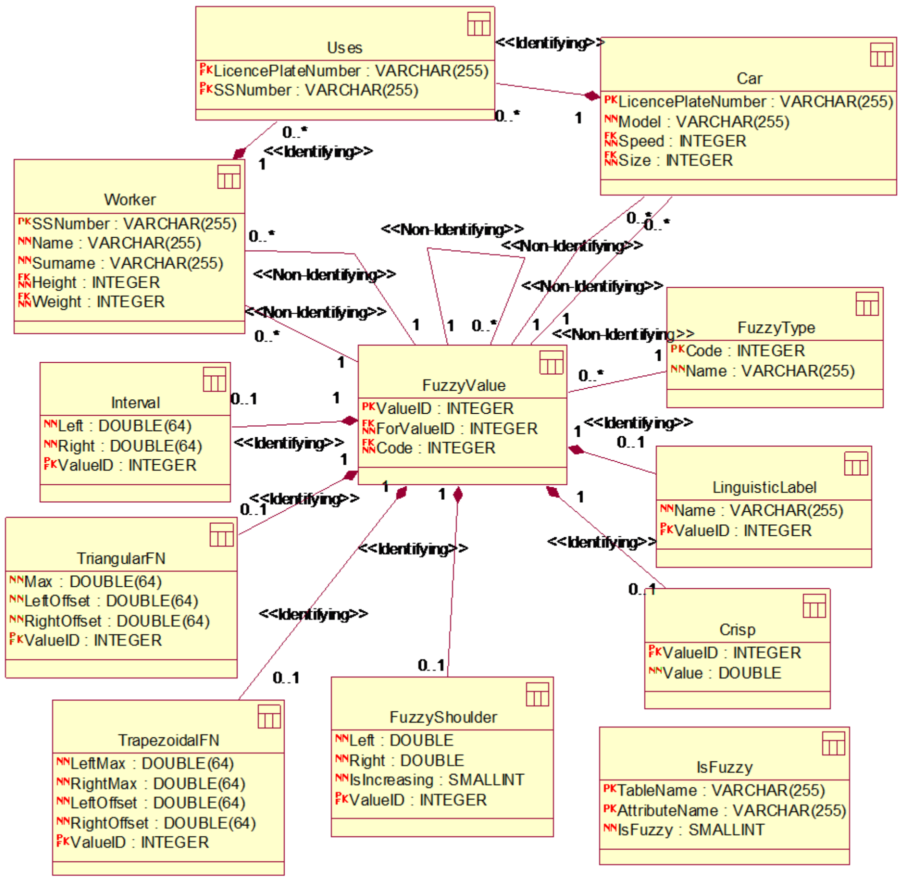


Fig. 2. Fuzzy relational data model.

For example, let us suppose that worker John Doe has a height designated with linguistic label Tall which represents the fuzzy shoulder with the membership function:

$$\begin{cases} 0, x < 180 \\ \frac{x}{20} - 9, 180 \leq x < 200 \\ 1, x \geq 200 \end{cases} \qquad (1)$$

,

and weights 80kg. In the table Worker there is a corresponding record with attribute values *SSNumber*=001, *Name*='John', *Surname*='Doe', *Height*=1, *Weight*=2. In the table FuzzyType there are three records with the following attribute values: Record T1: *Code*=1, *Name*='linguisticLabel'; Record T2: *Code*=2, *Name*='crisp'; Record T3: *Code*=3, *Name*='fuzzyShoulder'. Attributes *Height* and *Weight* in the table Worker are foreign keys that refer to the attribute *ValueID* in the FuzzyValue table. In this case, the table FuzzyValue contains three records with the following attribute values: Record

F1: *ValueID*=1, *ForValueID*=3, *Code*=1; Record F2: *ValueID*=2, *ForValueID*=NULL, *Code*=2; Record F3: *ValueID*=3, *ForValueID*=NULL, *Code*=3. Record F1 represents linguistic label Tall, Record F2 represents crisp value for weight and Record F3 represents fuzzy shoulder which corresponds to the linguistic label Tall. Further descriptions of these values are given in tables corresponding to their types described in the FuzzyType table. According to this, the table LinguisticLabel contains one record with the following attribute values: *Name*='Tall', *ValueID*=1. It describes the linguistic label Tall. The table FuzzyShoulder contains one record with the following attribute values: Left=180, Right=200, IsIncreasing=1, ValueID=3, describing the membership function given in equation (1), while the table Crisp contains a record describing the weight: *ValueID*=2, *Value*=80.

The rest of the values, for other fuzzy types, are stored in the database in a similar way. The complete description of all values and types that can be stored in the database can be found in [25, 33, 28]. The difference between the data model described there and this improved version is in structures that store fuzzy values. In the previous model, we had only two tables in the fuzzy meta data model - IsFuzzy and FuzzyValue. In that model, fuzzy values were stored in the table FuzzyValue as strings with predefined structure - one type for every type of fuzzy value that can be stored. In this way, the value of a column in a database record was not atomic, so it had to be decomposed in order to be used. That implies that our previous model was not even in the first normal form. Here we use one table for every fuzzy data type and have atomic values in the whole database.

Presented fuzzy meta model has been put through the synthesis algorithm [34] that guarantees that resulting model conforms to the 3rd normal form. Of course, fulfilment of theoretical conditions for the 3rd normal form relational model depends on the ground database model that we are creating too. In any case, this feature guarantees that if a database model is at least in the 3rd normal form, then the addition of the presented fuzzy meta model will result in a complete model at least in the 3rd normal form. In this way, presented fuzzy meta model significantly improves its theoretical and practical performance. The main reason for insisting on 3rd normal form in this model is the efficiency of the complete software system. In the previous case, when the values were not atomic, the system parsed the strings which correspond to the appropriate fuzzy values. Now, all information is obtained by following the primary-foreign key pairs, which results in better overall performance.

4. The CASE Tool

In this section we give an overview of the CASE tool for fuzzy relational model development. The application is implemented using Java programming language and Swing platform for the GUI (Fig. 3). It can be downloaded together with the source code and accompanying UML model from http://www.is.pmf.uns.ac.rs/fuzzydb.

4.1. Requirements

Requirements set in the process of modelling of the CASE tool include functions for simplified building of a fuzzy relational data model, as well as functions for its transformation to the SQL script.

Our intention was to implement a CASE tool capable for visual modelling and easy administration of all components of a fuzzy relational data model - tables, attributes, fuzzy data types and relationships. The CASE tool's GUI works in the similar way as in all modern tools of this type that allow modelling of classical relational models. So we do not describe the details here. In the model building process all the automation related to the migration of keys through relationships is included. This feature includes cascade deletion process, migration of keys during the relationship creation process and circular reference detection and prevention.

In addition, the CASE tool is required to allow easy SQL script generation for the specific database management system. In this sense, capabilities to specify the data types used by the DBMS and rules for mapping of the types used in the model (together with fuzzy data types) to these types had to be included.

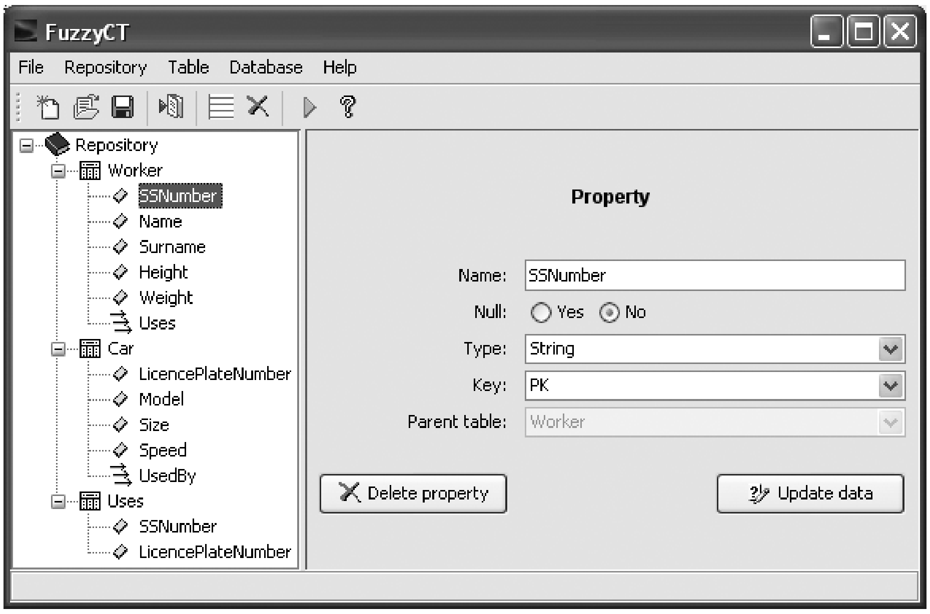
The complete UML model that includes the use case diagrams, the static system model (class diagrams) and the dynamic model of the main processes, as mentioned before, is available for download.

4.2. Elements

The main window consists of five parts shown at Fig. 3: menu, toolbar, navigation tree, main panel and status bar.

The menu and the toolbar contain all of the commands available in the CASE tool. Using those commands user can manage data model, specify data types in the model and their mapping to SQL data types. The only data type that exists by default is Fuzzy. Crisp data types have to be explicitly defined together with their mapping to SQL data types that are specific to the DBMS used.

The navigation tree is a visual representation of data about the model that exists in the repository. The repository can contain a set of tables, while the tables contain attributes and relationships (Fig. 3). The main panel is located to the left of the navigation tree. Its content depends on the element selected at the navigation tree. For every type of element at the navigation tree, the panel that allows editing of that element is defined. The status bar with some useful information is located at the bottom of the application window.



Name: SSNumber
Null: Yes [✓ No]
Type: String
Key: PK
Parent table: Worker

Fig. 3. The main window

4.3. Generating the SQL DDL script

This section provides insight into the process of generation of the SQL DDL (Structured Query Language Data Definition Language) script based on the fuzzy relational data model.

Activity diagram at Fig. 4 models the actions that need to be conducted in order to generate data definition SQL code from the model existing in the CASE tool. The first activity in the code generation is related to the FuzzyValue table definition. As mentioned above, this table is essential in our model, and presents a link between the model and the fuzzy meta model. The rest of the fuzzy meta model is defined in the similar way.

After that, the SQL code for the tables in the model is created. This activity has three sub activities. At first, for every table in the model we open a CREATE TABLE clause. After that, based on the table attributes, SQL code describing those attributes is generated inside the clause. At the end, the primary key constraint is added.

When all tables are created, foreign key constraints that connect tables need to be added. At first, code that links tables in the model to the fuzzy meta model via FuzzyValue table is created. Then the rest of the foreign key constraints are generated. Result of this process is a sequence of SQL clauses written into the file system as a DDL (Data Definition Language) text file.

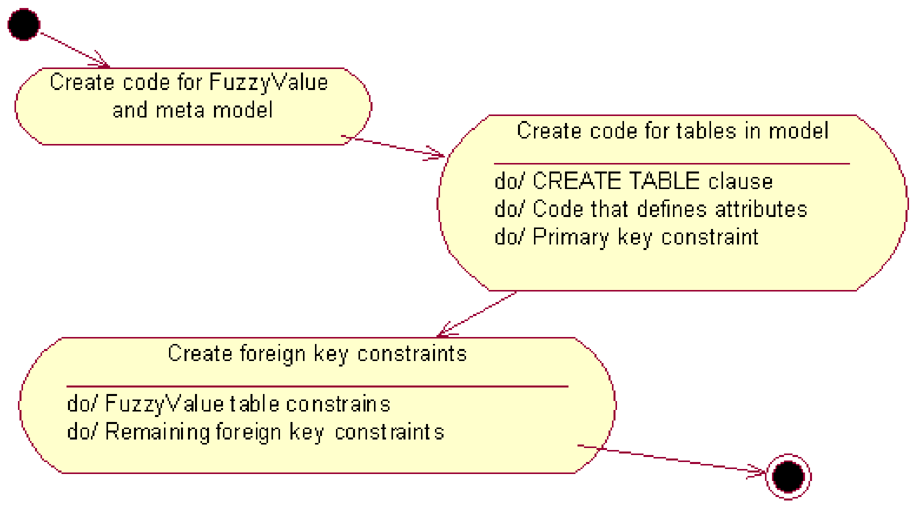

Create code for FuzzyValue and meta model
Create code for tables in model
do/ CREATE TABLE clause
do/ Code that defines attributes
do/ Primary key constraint
Create foreign key constraints
do/ FuzzyValue table constrains
do/ Remaining foreign key constraints

Fig. 4. The SQL DDL code generation process.

5. Motivation and Comparison

Here we give a comparison of our data model to the most advanced fuzzy relational data model available today - the FIRST-2 [6]. Our conclusion is that there are several similarities between them. Although the methods for fuzzy value representation are completely different, functionally, our model is a subset of the FIRST-2 model. Our intention was to define the simplest possible model that supports most widely used fuzzy concepts, and stores values as effectively as possible without too much overhead.

Fuzzy attributes of the type 1 in the FIRST-2 model are crisp values that our model also supports. Fuzzy types that our model covers are a subset of those represented by the fuzzy attributes type 2 and 3. Null values, intervals and trapezoidal fuzzy numbers in the FIRST-2 are represented by the structures that have these same names. A subset of the set of triangular fuzzy numbers, isosceles triangle, is represented by the approximate value with explicit margin in the FIRST-2 model. All other types of triangular fuzzy numbers, as well as fuzzy quantities can be represented by the possibility distributions with 2 and with 4 values in the FIRST-2, although these distribution types are more general.

The general fuzzy number from our model is known as the fuzzy attribute type 3 in the FIRST-2 model. Moreover, the FIRST-2 model describes a wider range of other possibilities for fuzzy values and combines atomic values according to their respective structure. In this paper we described an advanced version of our model that treats fuzzy values similarly. Although, functionally, our model is a subset of the FIRST-2, it gives theoretical contribution in modelling from the aspect of relational model theory because it conforms to the 3rd normal form. The basic disadvantage of the FIRST-2 model is non conformance even to the 1st normal form.

The fuzzy database query language FSQL is built on top of the FIRST-2 model using Oracle DBMS and PL/SQL stored procedures [23]. Similarly, we used the fuzzy-relational data model described in this paper to build an interpreter for the PFSQL language. We have developed the PFSQL query language from ground up, extending the features of SQL into the fuzzy domain. The PFSQL language is an extension of the SQL language that allows fuzzy logic concepts to be used in queries. Among other features described in [27,31,32] in detail, this query language allows priority statements to be specified for query conditions. For calculating the membership degree of query tuples when priority is assigned to conditions, we use the GPFCSP systems mentioned in introduction. Although the FSQL language has more features than the PFSQL, it does not allow usage of priority statements. The PFSQL is the first query language that does. Moreover, the PFSQL is implemented using Java, outside the database, which makes our implementation database independent.

Following this idea, we implemented the CASE tool described here in order to ease the fuzzy-relational model development and its usage in the real world applications with the PFSQL.

6. Conclusion

In this paper we present an innovative fuzzy-relational data model and a unique CASE tool for FRDB model development. Presented data model extends the relational model with capabilities to store fuzzy values and supports the execution of PFSQL queries. The CASE tool for fuzzy relational model development has been implemented using the Java programming language and the Swing platform for the graphic user interface. To the best of our knowledge, this is the only CASE tool with such capabilities in existence today.

In addition, we give a comparison between this model and the more general FIRST-2 model. It is our conclusion that this model represents a significant improvement compared to our previous model given in [28]. On the other hand, it is a functional subset of the FIRST-2 model, although the methods for the fuzzy value representation are completely different. Its compliance to the 3rd normal form makes it better theoretically founded than other known models of this kind.

In an effort to ease the PFSQL usage further, we implemented a fuzzy JDBC driver [24,27,32] that allows easy PFSQL statement execution within the Java environment. This set of tools supports the idea to specify the complete methodology for fuzzy-relational database applications development. A working version of our model, the CASE tool and the PFSQL is available for download from http://www.is.pmf.uns.ac.rs/fuzzydb.

References

1. Bosc, P., Kraft, D., Petry, F.: Fuzzy Sets in Database and Information Systems Status and Opportunities. Fuzzy Sets and Systems, Vol. 156, No. 3, 418–426. (2005)
2. Ma, Z.M., Mili, F.: Handling Fuzzy Information in Extended Possibility-Based Fuzzy Relational Databases. International Journal of Intelligent Systems, Vol. 17, No. 10, 925–942. (2002)
3. Prade, H., Testemale, C.: Fuzzy Relational Databases: Representational Issues and Reduction Using Similarity Measures. Journal of the American Society for Information Science, Vol. 38, No. 2, 118-126. (1987)
4. Buckles, B., Petry, F.: A Fuzzy Representation of Data for Relational Databases. Fuzzy Sets and Systems, Vol. 7, No. 3, 213-226. (1982)
5. Zvieli, A., Chen, P.: ER Modelling and Fuzzy Databases. In Proceedings of the 2nd International Conference on Data Engineering, Los Angeles, CA, 320-327. (1986)
6. Bahar, O., Yazici, A.: Normalization and Lossless Join Decomposition of Similarity-Based Fuzzy Relational Databases. International Journal of Intelligent Systems, Vol. 19, No 10, 885 – 917. (2004)
7. Raju, K., Majumdar, A.: Fuzzy functional Dependencies and Lossless Join Decomposition of Fuzzy Relational Database Systems. ACM Transactions on Database Systems, Vol. 13, No. 2, 129-166. (1988)
8. Umano, M.: Freedom-O: A Fuzzy Database System. Fuzzy Information and Decision Processes. In: Gupta, Sanchez (eds.): Fuzzy Information and Decision Processes, North-Holand, Amsterdam. (1982)
9. Kacprzyk, J., Zadrozny, S., Ziolkowski, A.: Fquery III+: A “Human-Consistent” Database Querying System Based on Fuzzy Logic with Linguistic Quantifier. Information Systems, Vol. 14, No. 6, 443–453. (1989)
10. Sanchez, E.: Importance in Knowledge Systems. Information Systems, Vol. 14, No. 6, 455–464. (1989)
11. Vandenberghe, R., Schooten, A.V., Caluwe, R. D., Kerre, E.: Some Practical Aspects of Fuzzy Database Techniques: An Example. Information Systems, Vol. 14, No. 6, 465–472. (1989)
12. Wong, M., Leung, K.: A Fuzzy Database-Query Language. Information Systems, Vol. 15, No. 5, 583–590. (1990)
13. Bosc, P., Pivert, O., Farquhar, K.: Integrating Fuzzy Queries into an Existing Database Management System: An Example. International Journal of Intelligent Systems, Vol. 9, No. 5, 475–492. (1994)
14. Bosc, P., Pivert, O.: SQLf: A Relational Database Language for Fuzzy Querying. IEEE Transactions on Fuzzy Systems, Vol. 3, No. 1, 1-17. (1995)
15. Medina, J.M., Pons, O., Vila, M.A.: GEFRED: A Generalized Model of Fuzzy Relational Databases. Information Sciences, Vol. 76, No. 1-2, 87-109. (1994)
16. Galindo, J., Medina, J.M., Aranda, M.C.: Querying Fuzzy Relational Databases Through Fuzzy Domain Calculus. International Journal of Intelligent Systems, Vol. 14, No. 4, 375-411. (1999)
17. Galindo, J., Medina, J.M., Aranda-Garrido, M.C.: Fuzzy Division in Fuzzy Relational Databases: an Approach. Fuzzy Sets and Systems, Vol. 121 No. 3, 471–490. (2001)
18. Galindo, J., Medina, J.M., Cubero, J.C., Garcia, M.C.: Relaxing the Universal Quantifier of the Division in Fuzzy Relational Databases. International Journal of Intelligent Systems, Vol. 16, No. 6, 713-742. (2001)

19. Galindo, J., Medina, J.M., Pons, O., Cubero, J.C.: A Server for Fuzzy SQL Queries. Lecture Notes in Computer Science, Vol. 1495, 164-174. (1998)
20. Chen, G., Kerre, E.: Extending ER/EER Concepts Towards Fuzzy Conceptual Data Modelling. In Proceedings of the IEEE International Conference on Fuzzy Systems. Anchorage, AK, 1320-1325. (1998)
21. Kerre, E., Chen, G.: Fuzzy Data Modelling at a Conceptual Level: Extending ER/EER Concepts. In: O. Pons (ed.): Knowledge Management in Fuzzy Databases, Physica Verlag, Heidelberg, 3-11. (2000)
22. Chaudhry, N., Moyne, J., Rundensteiner, E.: A Design Methodolgy for Databases with Uncertain Data. In Proceedings of the 7th International Working Conference on Scientific and Statistical Database Management. Charlottesville, VA, 32-41. (1994)
23. Galindo, J., Urrutia, A., Piattini, M.: Fuzzy Databases: Modeling Design and Implementation. IDEA Group Publishing, Hershey, USA. (2006)
24. Škrbić, S., Takači, A.: On Development of Fuzzy Relational Database Applications. In Proceedings of the 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Malaga, Spain, 268-273. (2008)
25. Takači, A., Škrbić, S.: How to Implement FSQL and Priority Queries. In Proceedings of the 3rd Serbian-Hungarian Joint Symposium on Intelligent Systems, Subotica, Serbia, 261-267. (2005)
26. Takači, A.: Towards a Priority Based Logic. In Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty, Paris, France, 247-252. (2006)
27. Škrbić, S.: Fuzzy Logic Usage in Relational Databases. Doctoral dissertation, Faculty of Science, Novi Sad. (2009) (in Serbian)
28. Takači, A., Škrbić, S.: Data Model of FRDB with Different Data Types and PFSQL. In: Galindo, J. (ed.): Handbook of Research on Fuzzy Information Processing in Databases, IGI Global, Hershey, PA, 403-430. (2008)
29. Luo, X., Lee, J., Leung, H., Jennings, N.: Prioritized Fuzzy Constraint Satisfaction Problems: Axioms, Instantiation and Validation. Fuzzy Sets and Systems, Vol. 136, No. 2, 151-188. (2003)
30. Takači, A.: Schur-concave Triangular Norms: Characterization and Application in PFCSP. Fuzzy Sets and Systems, Vol. 155, No. 1, 50-64. (2005)
31. Škrbić, S., Racković, M.: PFSQL: a Fuzzy SQL Language With Priorities. In Proceedings of the 4th International Conference on Engineering Technologies, Novi Sad, Serbia, 58-63. (2009)
32. Škrbić, S., Takači, A.: An Interpreter for Priority Fuzzy Logic Enriched SQL. In Proceedings of the 4th Balkan Conference in Informatics, Thessaloniki, Greece, 96–100. (2009)
33. Takači, A., Škrbić, S.: Measuring the Similarity of Different Types of Fuzzy Sets in FRDB. In Proceedings of the 5th Conference of the EUSFLAT, Ostrava, Czech Republic, 247-252. (2007)
34. Bernstein, P.A.: Synthesizing Third Normal Form Relations From Functional Dependencies. ACM Transactions on Database Systems, Vol. 1, No. 4, 277-298. (1976)

Srđan Škrbić received the PhD degree in 2009 from the Department of mathematics and informatics, Faculty of Science at University of Novi Sad. He is currently an assistant professor at the same department. His research interests include intelligent database systems and fuzzy database modeling. He has published more than 30 scientific papers.

Miloš Racković is a full professor at the Department of mathematics and informatics, Faculty of Science at University of Novi Sad. He received the PhD from the same department in 1996. He works in the fields of computer science, artificial intelligence and databases. Professor Racković has published 82 scientific papers, two monographs and three books. He mentored more than 200 Bsc theses, 20 Msc theses and two PhD theses.

Aleksandar Takači, received the PhD degree in 2006 from the Department of mathematics and informatics, Faculty of Science at University of Novi Sad. He is currently an assistant professor at the Faculty of Technology, University of Novi Sad. His research interests include fuzzy systems and fuzzy logic. He has published more than 30 scientific papers.

Received: January 02, 2010; Accepted: April 13, 2010.