Imam Mohammad Bin Saud Islamic University
College of Computer and Information Sciences

Human-like Flexible Queries of Relational Databases: Fuzzy-based Approach

By:

Saleh Khalid Al-Saeed 439016667
Abdulelah Mohammed Al-Anazi 439015929
Rayan Khalid Al-Mengash 439016742

Supervised by:

Dr. Ahmed Gadallah

Project Submitted in Fulfillment for the [CS493-IS498-DS471] Course requirements

2nd Semester
2022

Abstract

This paper presents a human-like flexible fuzzy-based approach to querying relational databases. Although many fuzzy query approaches have been proposed, a more human-like query approach is required. The majority of previously proposed fuzzy query approaches have the drawback of converting any fuzzy query statement into a crisp query statement, then evaluating the resulting tuples to compute their fuzzy query matching degrees. Our main goal is to allow each user to define or set his or her own fuzzy profile, which is almost subjective and reflects his or her own point of view. The user's linguistic terms, such as linguistic variables, values, hedges, and fuzzy numbers, will be incorporated into the profile, which will then generate human-like fuzzy queries to obtain a ranked result based on the degree of matching. As a result, the time it takes to execute a fuzzy query statement will be reduced. This method makes using fuzzy linguistic values in all clauses of a select statement simple. The proposed approach has the added benefit of speeding up the execution of fuzzy query statements.

Keywords

Fuzzy Query,

Fuzzy Logic,

Fuzzy numbers,

Linguistic variable,

Linguistic values,

Linguistic hedges,

Membership functions.

List of abbreviation

SQL: Structured Query Language.

DBMS: Database Management System.

GEFRED: A generalized model of Fuzzy Relational Databases.

FMB: Fuzzy Management Base.

PL: Procedural language

Table of Contents

List of Figures

List of Tables

Chapter One: Introduction

1.1 Introduction

When it comes to querying relational databases, the Structured Query Language is essential. You can change and get data that is very clean and precise using this tool. on the other hand, it is unable to respond to questions that are not clear, precise, or vague in any way. When someone asks a question, they frequently use words that are personal to them. This makes the question ambiguous and unclear. For example, when it comes to finding a good student in class, different people have different ideas about what that entails. However, because the database management system isn't very good at dealing with queries that use words in the language that people speak, there are numerous issues. Consider the following query: "Obtain the names and addresses of university students who are approximately the right height to play football." A traditional SQL statement cannot be used to write or change this query. A fuzzy query statement, on the other hand, is very flexible and human-like because it is based on fuzzy set theory, which makes it simple to express and modify. Despite its excellent querying capabilities, SQL lacks the ability to support human-like queries. The majority of human-like queries involve manipulating linguistic values rather than numerical ones. People use words like "short," "tall," and "hot" to describe things that can't be expressed in standard SQL statements. It's also not very adaptable when it comes to manipulating linguistic values or modifiers like "something like," "nearly," "reasonably," and so on, which are easily represented by the concept of fuzzy sets.

1.2 Aims and objectives

Our main goal is to enable each user to define or set their own fuzzy profile, the user will include in his or her profile their linguistic terms, such as linguistic variables and values, hedges, and fuzzy numbers. which is almost subjective and reflects their own point of view. The main goals of this project are to generate human-like fuzzy queries and be able to obtain a ranked result based on the user's degree of matching, as well as to reduce the response time of executing a fuzzy query statement while making relational database querying as humane as possible, and to create a database that can store the fuzzy queries as objects.

1.3 Methodology

In order to achieve the best results during the development of this project, the approach that has been adopted relies heavily on the concept of fuzzy set. A fuzzy set is typically a set with smooth boundaries. Fuzzy set theory therefore generalizes classical set theory to permit partial membership of its elements to the set, and it was developed to cover areas that cannot be covered by classical set theory. This approach aims mainly to allow the generation and manipulation of flexible, human-like queries, by letting the user to create their own profile by entering their linguistic variables, as well as their own linguistic values and fuzzy numbers for each linguistic value, this approach handles multiple subjective views with multiple users and relies entirely on PL/SQL statements.[20] Consequently, all generated fuzzy queries will be compatible with Standard SQL without the need for an interpreter or parser. The proposed method permits the generation of fuzzy query statements using standard SQL syntax. In addition, the system architecture will be divided into two parts: the user interface and the database management system, because the user interface will be programmed in java or python, the system's development environment will be in NetBeans or Visual Studio Code, and MySQL workbench will be used to create and manage the database system.

1.4 Project timeline

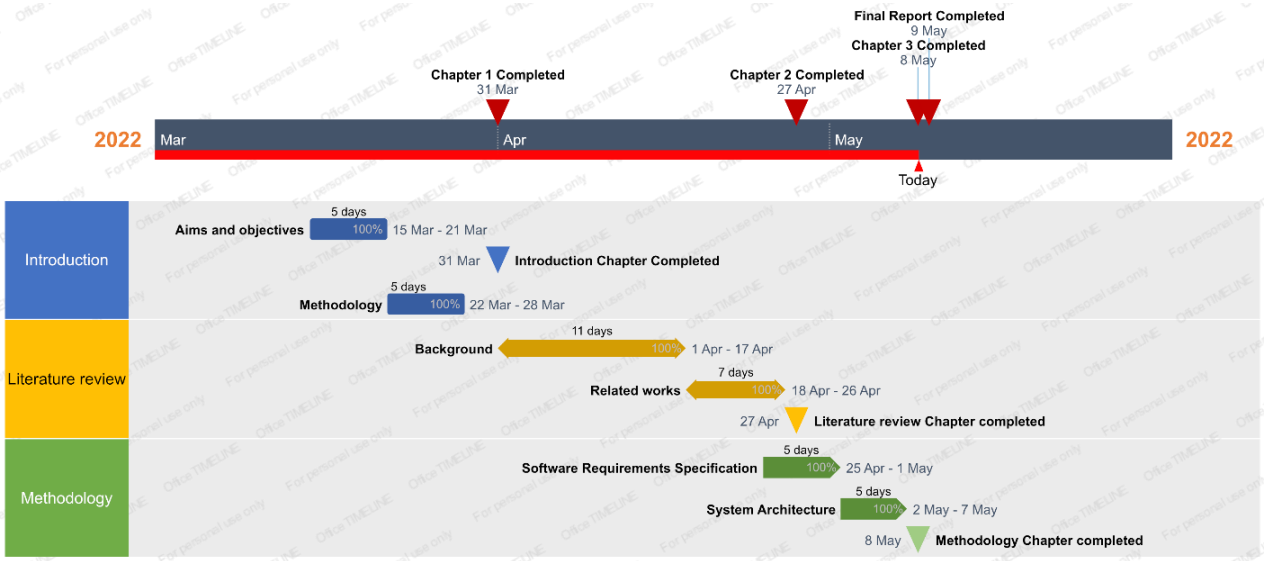


Fig. 1. Project Timeline.

1.5 Team qualifications

Student Name	Qualifications
Saleh Khalid Al-Saeed	• Has experience on web development • Knows how to program with SQL. • Fluent in databases. • Experienced with Java and Spring framework.
Abdulelah Mohammed Al-Anazi	• Computer and technology skills • Java developer. • Has experience in Artificial Intelligence. • Experienced in databases.
Rayan Khalid Al-Mengash	• Computer and technology skills • Java developer • Has experience in Artificial Intelligence. • Experienced in databases.

Table. 1. Team Qualifications.

1.6 Conclusion

This chapter provides an overview of fuzzy logic, as well as the project's goals and objectives, the methodology used, the project timeline, and the names and qualifications of the students. It also demonstrates the flexibility of fuzzy logic when dealing with vague and imprecise data, as well as the power of representing such imprecision in a human-like manner.

Chapter Two: Literature review

2.1 Introduction

In this chapter we will provide a brief introduction to fuzzy logic and fuzzy sets, as well as the crisp sets and the differences between them, followed by the definition of fuzzy numbers and linguistic variables, values and hedges, and membership function, followed by some problems in other proposed works and several related works.

2.2 Background

2.2.1 Fuzzy logic and Classical logic

Classical logic is appropriate for formalizing reasoning involving bivalent propositions like "5 is a prime number" or "if x is a positive integer and y = x+ 1 then y is a positive integer," that is, propositions that can only be true or false in principle. Classical sets are also appropriate for representing collections (of objects) with sharp, clear-cut boundaries, such as "all prime numbers less than 100." An arbitrary given object is either a member of or not a member of any such collection. Most propositions that people use to communicate information about the world are not bivalent, so fuzzy logic is a computing approach based on "degrees of truth" rather than the usual "true or false" (1 or 0) Boolean logic on which the modern computer is based. Rather than being true or false, the truth of such propositions is a matter of degree. "The humidity outside is moderate," for example, is a statement whose truth is contingent on the actual outside humidity. Because humidity levels can range from 0% to 100%, the term "moderate" refers to values that are close to 50% in some way. The farther the actual outside humidity is from 50 percent (on both sides), the less true the proposition "the outside humidity is moderate," according to our intuition. In a bivalent equivalent, the linguistic term "moderate" would have to be replaced with either a single real number, h, referring to h percent humidity, or a pair of real numbers, h1 and h2, referring to a range of humidity values between h1 and h2 percent.[14][15].

2.2.2 Fuzzy sets and Classical sets

The concept of a classical set is central to almost every branch of mathematics, both pure and applied. A set is any definite and distinct collection of objects that is conceived as a whole. The members of a set are the objects that make up the set. Classical sets meet two fundamental criteria. First, members of each set can be distinguished from one another, and second, whether an object is a member of the set can be determined for any given object. Zadeh (1965) introduced fuzzy sets, which differ from classical sets by rejecting the second requirement. Fuzzy sets, unlike classical sets, do not require sharp boundaries to distinguish their members from other objects. The membership of any object in each fuzzy set is a matter of degree rather than affirmation or denial, as is the case with classical sets. On any given universal set, fuzzy sets are defined as functions that are analogous to the characteristic functions of classical sets. Each of these functions assigns a truth degree to each object in the universal set. The defined fuzzy set is called a standard fuzzy set if the truth degrees are real numbers in the unit interval [0,1]. A standard fuzzy set A in universe U is a function A: U $\rightarrow$ [0,1] that assigns a number A(x) from [0,1] to each element x of U. This number is known as the degree (or grade) of membership of x in A, and it represents the degree (or grade) to which x is considered a member of the fuzzy set A collection. That is, A(x) denotes the degree of truth of a proposition. " x is an A member.".[14][15].

2.2.3 Fuzzy numbers

Fuzzy sets defined on the set R of real numbers are of particular interest among the various types of fuzzy sets. These sets' membership functions are A: R $\rightarrow$ [0, 1]. These clearly have a quantitative meaning and can be viewed as fuzzy numbers or fuzzy intervals under certain circumstances. They should capture our intuitive conceptions of approximate numbers or intervals, such as "numbers that are close to a given real number" or "numbers that are around a given interval of real numbers," to be viewed in this light. These concepts are critical for describing the states of fuzzy variables and, as a result, play a key role in a variety of applications, including fuzzy control, decision making, approximate reasoning, optimization, and statistics with uncertain probabilities.[14][15].

2.2.4 Linguistic variables and values

A linguistic variable is one with ambiguous (linguistic) values. The linguistic variable "height," for example, could have the linguistic values "Tall," "Medium," and "short." The membership function represents each of these fuzzy values. Figure 2 depicts the fuzzy values "Tall," "Medium," and "Short."

As shown in figure 2, there are four different levels in the definition of a linguistic variable. At the top level, we have the variable's name (e.g., height). We have the labels of fuzzy values at the level below it (starting with an initial set of values called primary values or term set). Membership values are further down, and the universe of discourse is at the bottom. It's worth noting that linguistic variables have a dual nature: at higher levels, they take on a symbolic linguistic form, while at lower levels, they take on a well-defined quantitative analytical form, the membership function.

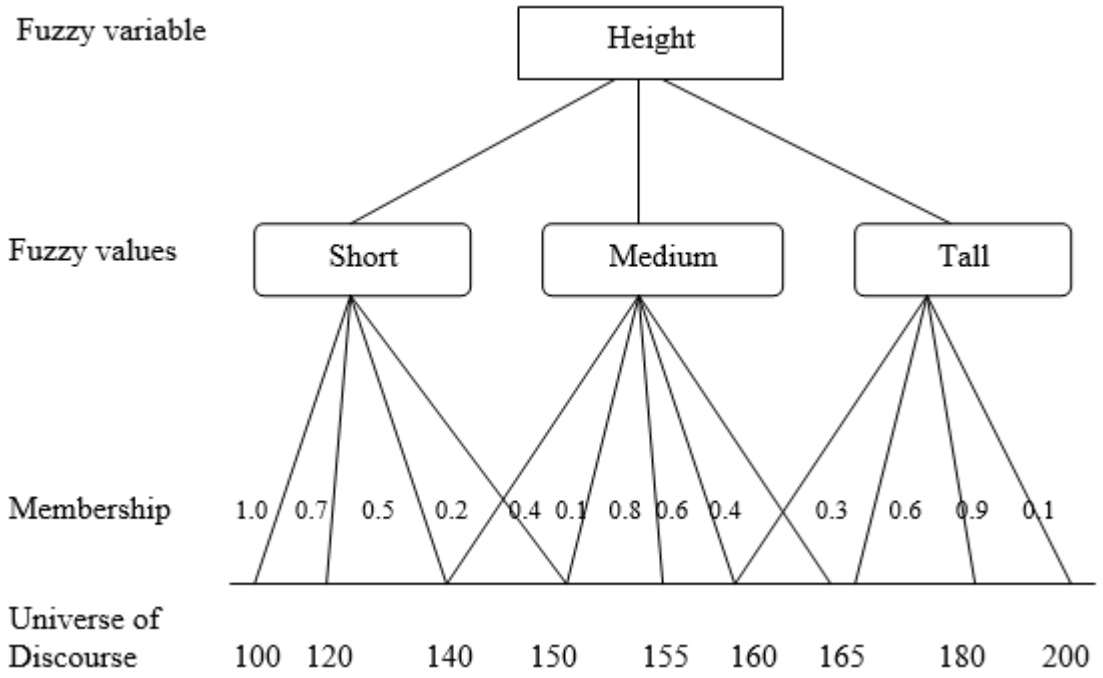


Fig. 2. The linguistic variable “Height” and its linguistic values “Tall”, “Medium” and “Short”.

2.2.5 Linguistic hedges

A hedge is a fuzzy set's modifier. It creates a compound fuzzy set by changing the meaning of the original set. Every linguistic variable can have a number of different linguistic

values, each of which is referred to as a "atomic term." The values allowed for the linguistic variable are represented by such atomic terms. When atomic terms are combined with linguistic hedges such as "very, more or less, slightly, fairly, etc.," the linguistic variable takes on new values. "Very" and "more or less" are two commonly used hedges, which are defined as follows:

Very: $\mu_{\text{very A}}(x) = [\mu_A(x)]^2$

More or less: $\mu_{\text{more or less A}}(x) = \sqrt{\mu_A(x)}$

Figure 3 shows how these hedges are used to illustrate the concept of tall students. As shown in the diagram, the word "very" narrows the membership function. "More or less," on the other hand, broadens the membership function. This makes sense because the criteria for "very tall" should be stricter than those for "tall," while the criteria for "more or less" should be more lenient. A hedge can theoretically be applied to any fuzzy set. However, in practice, it is only used when the compound term is meaningful.

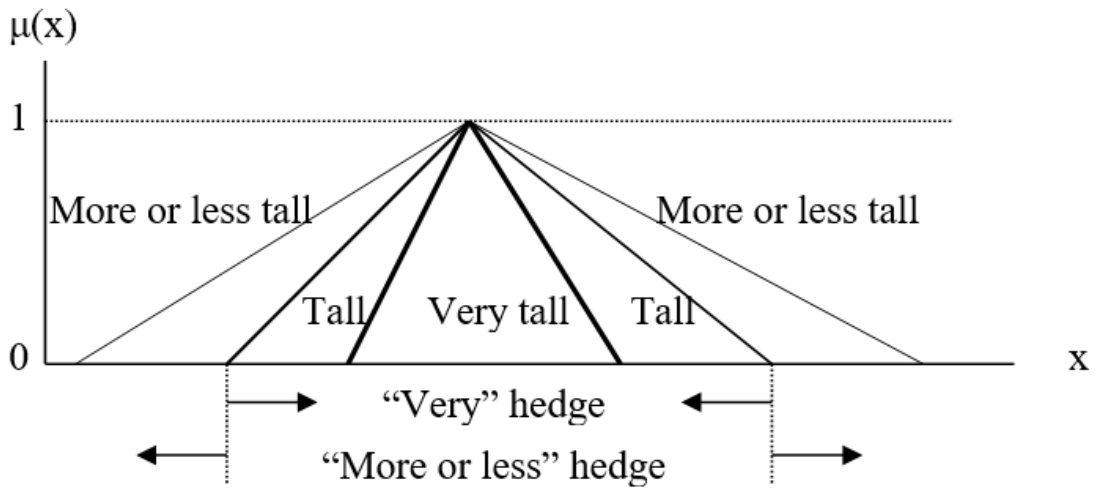


Fig. 3. Using the linguistic hedges "Very" and "More or Less" with the atomic term tall.

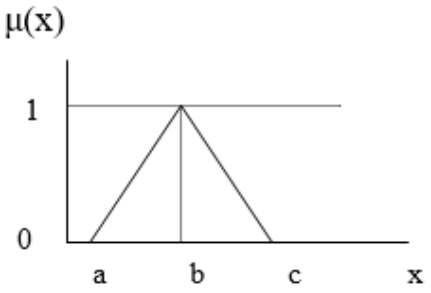
2.2.6 Commonly Used Membership Functions

To represent fuzzy sets, a variety of membership functions can be used. Designer experts and decisions are involved in choosing a specific membership function to represent a fuzzy set. When representing a fuzzy set, it should be evaluated to see if it corresponds to the linguistic concept. The efficiency here is in determining which membership function is best for the fuzzy set. The following are some of the most commonly used membership functions.

- **Triangular Membership Function**

The Triangular membership function, as shown in EQ 1 and figure 4, has three parameters: the first specifies that each predecessor element has zero membership in the fuzzy set, the second specifies that the elements have full membership, and the third specifies that successor elements have membership zero.

$$\text{Triangle}(x: a, b, c) = \begin{cases} 0 & x < a \\ (x-a)/(b-a) & a \leq x > b \\ (c-x)/(c-b) & b \leq x > c \\ 0 & x > c \end{cases}$$



EQ. 1.

Fig. 4. Triangular Membership Function.

- **Trapezoidal Membership Function**

The first and last parameters of the trapezoidal membership function define the boundaries of zero membership elements, while the second and third parameters define the gradual membership values that range from 0 to 1 and 1 to 0, respectively, as shown in EQ 2 and figure 5.

$$\text{Trapezoid}(x: a, b, c, d) = \begin{cases} 0 & x < a \\ (x-a)/(b-a) & a \le x < b \\ 1 & b \le x \le c \\ (d-x)/(d-c) & c \le x < d \\ 0 & x > c \end{cases}$$

EQ. 2.

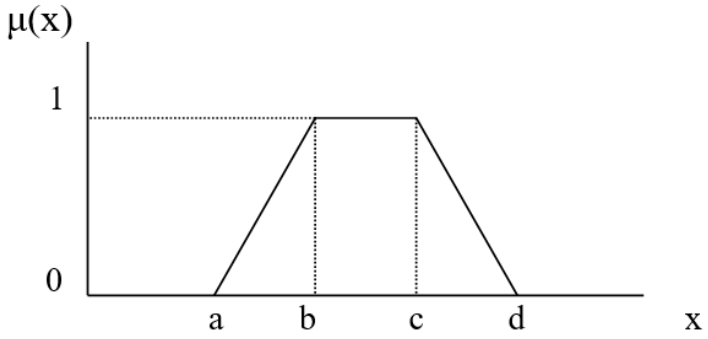


Fig. 5. Trapezoidal Membership Function.

2.3 Related work

2.3.1 Problems with related works

This section presents several architectures and applications that have been proposed and developed to allow fuzzy database queries. Unfortunately, the majority of previously developed fuzzy SQL architectures have a number of flaws. When interacting with a large group of people, both of the previously proposed approaches [1] and [2] are incapable of dealing with multiple subjective points of view. [4] Preferences are instead used to help reduce the amount of data returned in response to user queries. Furthermore, because each tool has its own syntax, the approaches proposed in [11], [2], and [1] do not use a standardized format when writing statements, making comparison difficult. However, the majority of the proposed solutions, such as those in [1], [2], [10], and [11], rely on a time-

consuming parser/translator to check and convert fuzzy queries to crisp SQL queries, which takes a long time. Furthermore, such approaches define the Fuzzy Meta-knowledge Base (FMB), which includes a set of tables for storing all necessary information to describe and manipulate fuzzy attributes and terms [2], as well as a set of tables for storing all necessary information to describe and manipulate fuzzy attributes and terms [3]. Every time a fuzzy query statement is generated, the FMB associated with it must be accessed. This operation is necessary in order to complete the processing of the generated fuzzy query statement, which is a time-consuming operation [5], [10], and [11]. This slows down the querying process because each generated fuzzy query must be analyzed and translated into a standard SQL statement that complies with the definitions of all fuzzy terms or operators stored in the Fuzzy Management Base (FMB). [7]. Furthermore, when processing nested and correlated fuzzy queries [3], the approaches proposed in [2], [5], and [10] are inefficient.

2.3.2 Fuzzy querying with SQL: Fuzzy view-based approach

The concept of making database management systems more flexible by computing a Boolean query from a fuzzy one to be evaluated is both costly and old. Studies in this area show that we need a simple and quick approach that more effectively collaborates with commercial DBMS to achieve better results. They proposed an alternative processing strategy in this paper that allows users to query databases directly with SQL without the need for a translation mechanism, based on the use of fuzzy views. The following is a breakdown of the paper's structure. It presents a brief overview of related works in the second section. After that, in section 3, it presents a description of the proposed fuzzy query approach. The architecture of a fuzzy querying user interface based on this proposal is described in Section 4. Finally, the paper presents their contribution's performance through a quantitative comparison and draws some conclusions. The primary goal of this proposal is to address the issues raised in the preceding section. It makes use of a view to keep track of the satisfaction levels associated with user-defined fuzzy predicates. As a result, fuzzy query execution will be sped up, and all fuzzy queries generated will be compatible with standard SQL, eliminating the need for an interpreter or analyzer.

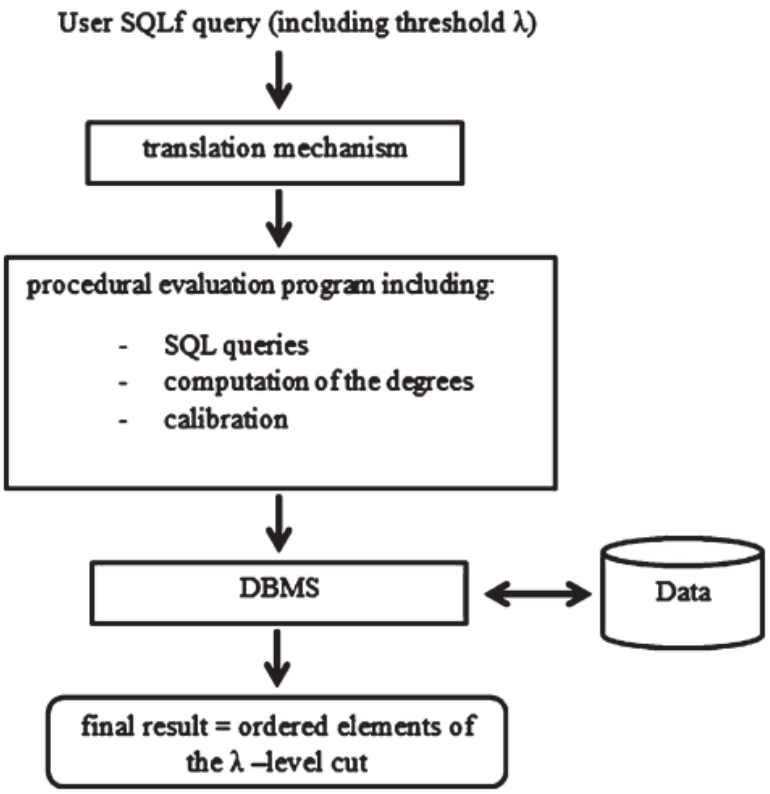


Fig. 6. Architecture for the evaluation strategy.

This approach will also benefit from the advantages of using PL / SQL, such as high performance by reducing traffic between the application and the database, tight integration with SQL, high productivity, scalability, manageability, and object-oriented support, and portability to any operating system, because it uses the SQL language to query the database without the need for intermediate software. In the subsections that follow, the concept of view in the context of a database is first discussed, followed by the query processing strategy. A view is a logical representation of a user request made on one or more tables. It is a virtual table, meaning its data is not stored in a database table. As a result, a view can be thought of as a dynamic window on the data, or even a stored query, but only the definition is saved, not the result, which is calculated dynamically. The view's benefit here is that it allows the user to store complex queries and use them as if they were tables. The approach here uses views to optimize the processing of fuzzy queries in this context, which is that of fuzzy queries, in order to speed up the fuzzy query. The basic idea is to create a view from a table that allows storing the satisfaction levels associated with tuples rather than calculating them using user functions each time a fuzzy query is run, which incurs a significant additional cost because the DBMS analyzes the entire table

without using indexes due to restrictions imposed by the use of user functions. In the generated fuzzy view, user-defined linguistic values attributed to the column associated have been represented as columns.[17].

Consider this employee relation:

EMP(#emp, e-name, salary, age, #dep)

to better understand the proposal. Table 2 shows an example of content

EMP relation

emp	e-name	salary	age	dep
12	Martin	5000	29	30
10	Smith	7000	46	2
20	Marc	8000	38	40
30	Aline	2000	18	55
5	Jean	20000	56	101

Table. 2. EMP relation.

2.3.3 New Architecture of Fuzzy Database Management Systems

The work of Medina et al., who introduced the GEFRED model and its associated language, FSQL, in 1994, is the subject of this paper. Fuzzy comparators, fuzzy attributes, fuzzy constants, and other new concepts are introduced in this language. Fuzzy Query is the software that is used to use this language (FQ). Despite the fact that it solved several problems related to flexible query modeling, FQ has some limitations: (1) It only allows for flexible FRDB querying, (2) the FRDB is assumed to be already implemented under Oracle, and (3) the DB implementation is done manually by the user. (4) FQ is not suitable for FRDB in practice because there are more than ten tables in this set. This paper proposed a new Fuzzy Relational DBMS (FRDBMS) architecture based on the GEFRED model. The RDBMS Oracle is used in this architecture, which is based on the weak coupling principle. This FRDBMS includes all of the features of a traditional DBMS, including FRDB description, manipulation, and querying. This paper is divided into five sections, in addition to the introduction. The basic concepts of FRDB are presented in Section 2. The architectures that have already been used for flexible querying modeling are presented in Section 3. The architecture type of FRDBMS is discussed in Section 4.

The new architecture of the FRDBMS, as well as its implementation, are presented in Section 5. Section 6 provides an assessment of this work as well as some future perspectives. Medina et al. proposed the Generalized model Fuzzy heart Relational Database (GEFRED) in 1994. One of the most significant advantages of this model is that it is based on a general abstraction that allows for the use of a variety of approaches, no matter how dissimilar they may appear. It is, in fact, based on the generalized fuzzy domain and generalized fuzzy relation, which are classic domains and classic relations, respectively. The table 3 below lists the data types that this model can handle. [18].

1. A single scalar (e.g., Behavior=good, represented by the possibility of distribution 1/good).
2. A single number (e.g., Age=28, represented by the possibility of distribution 1/28).
3. A set of mutually exclusive possible scalar assignations (e.g., Behavior={Bad, Good}, represented by {1=Bad, 1=Good}).
4. A set of mutually exclusive possible numeric assignations (e.g., Age={20, 21}, represented by {1/20, 1/21}).
5. A possibility distribution in a scalar domain (e.g., Behavior={0.6/Bad, 1.0/Regular}).
6. A possibility distribution in a numeric domain (e.g. Age={0.4/23, 1.0/24, 0.8/25}, fuzzy numbers or linguistic labels).
7. A real number belonging to [0, 1], referring to the degree of matching (e.g., Quality=0.9).
8. An Unknown value with possibility distribution Unknown={1/u: u ϵ U} on domain U, considered.
9. An Undefined value with possibility distribution Undefined={0/u: u ϵ U} on domain U, considered.
10. A NULL value given by NULL={1/Unknown, 1/Undefined}.

Table. 3. Data types in the GEFRED model.

With Oracle DBMS, this paper proposes a weak coupling approach. The proposed FRDBMS remains committed to the GEFERD model, so the data description and manipulation language is FSQL. A FRDBMS can model an RDB (described in SQL language) or a FRDB (described in FSQL language) because the FSQL language is an extension of the SQL language (described in FSQL language). The principle behind this coupling is the creation of a software layer that allows users to convert commands written in the FSQL language into SQL equivalents. This principle is depicted in Figure 7 and 8. [18].

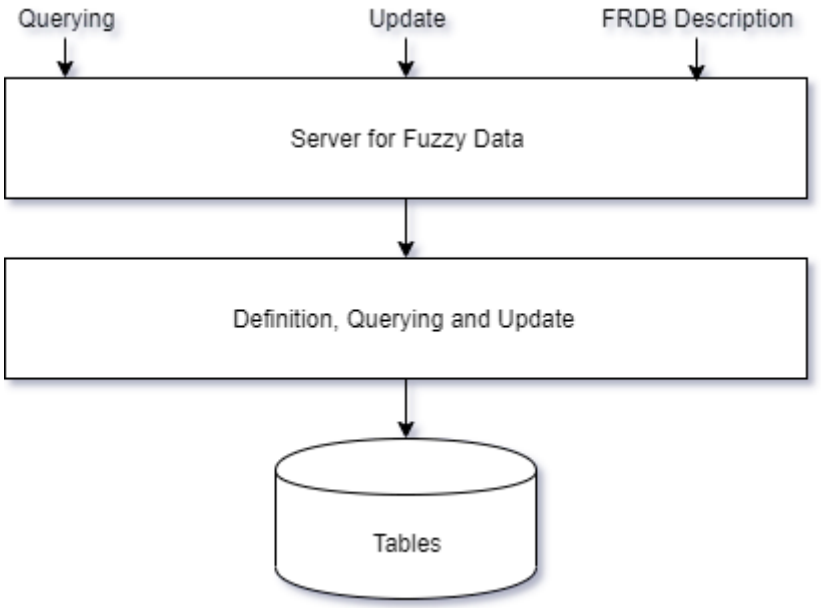


Fig. 7. Illustration of the FRDBMS architecture.

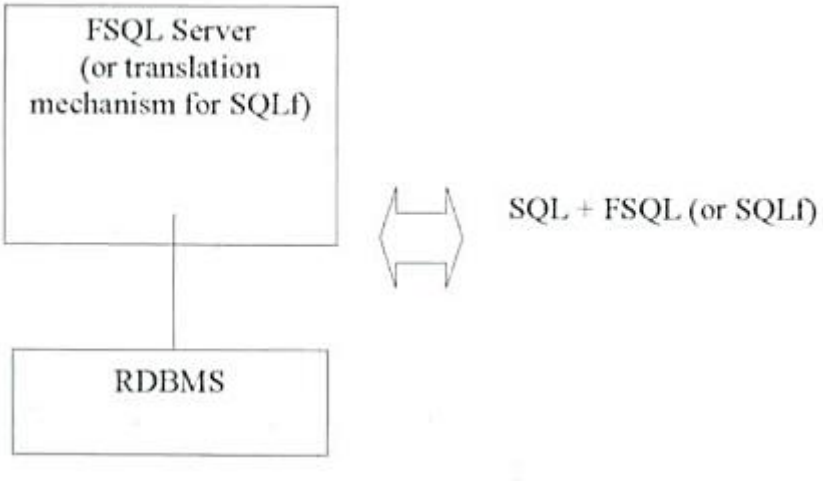


Fig. 8. Weak Coupling.

The installation of this architecture is described in figure 9 below.

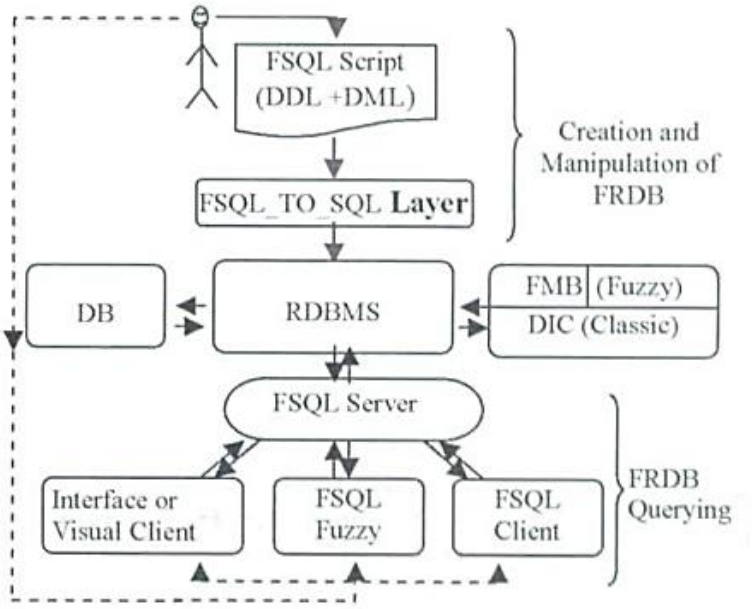


Fig. 9. Architecture of extended FIRST.

While looking through the information stored in the FMB, the FSQL Server translates fuzzy queries written in FSQL language. After this phase is completed, the DBMS transparently manages the crisp data translated by the FSQL Server. It employs a tool known as FSQL_TO_SQL, which enables the implementation of a FRDB described in FSQL under the Oracle 8i database management system. This tool converts FSQL scripts into SQL scripts automatically, while also indicating the changes made to the FMB level.[18].

2.3.4 Towards the Methodology for Development of Fuzzy Relational Database Applications

Using fuzzy logic and fuzzy sets, this paper investigates the possibilities of extending the relational data model with mechanisms that can handle imprecise, uncertain, and inconsistent attribute values. It presents a fuzzy relational data model that can be used to represent fuzzy knowledge in relational databases and guarantees the model's 3rd normal form. It also describes a CASE tool for developing fuzzy database models, which appears to be the first implementation of such a tool. In this sense, this paper represents a significant step forward in the specification of a methodology for the development of fuzzy relational database applications. This paper begins with a detailed description of a number of references that describe research on the use of fuzzy logic in relational databases. This overview covers everything from the origins of the concept to the most recent approaches. The approach to fuzzy-relational data modeling is described in the third section. It examines the theoretical value of the fuzzy meta model and provides a detailed description of it. The fourth section includes a description of the CASE tool for fuzzy-relational data modeling, which is the first of its kind. In the fifth section, it compares its approach to some previous approaches. Then the conclusion at the final section. The relational model extensions that make up the fuzzy relational data model variant. The model stores crisp values in the same way that a relational model does, but it defines a fuzzy meta data model for fuzzy values. Furthermore, it elucidates the process of transforming a classic relational model with fuzzy attributes into a corresponding fuzzy relational data model. It must find a way to store data about its characteristic function if it wishes to store a fuzzy value. It could theoretically store any fuzzy value this way. However, only a few characteristic functions are used in practice. From this perspective, we'll call them fuzzy data types. As a result, it only considers a small number of fuzzy data types in order to produce a data model that is both efficient and simple. Tables Worker and Car, as well as an intersection table, are included in an example relational model shown in Fig. 10. It employs to represent this many-to-many relationship. Worker and Car each have two fuzzy attributes. [19].

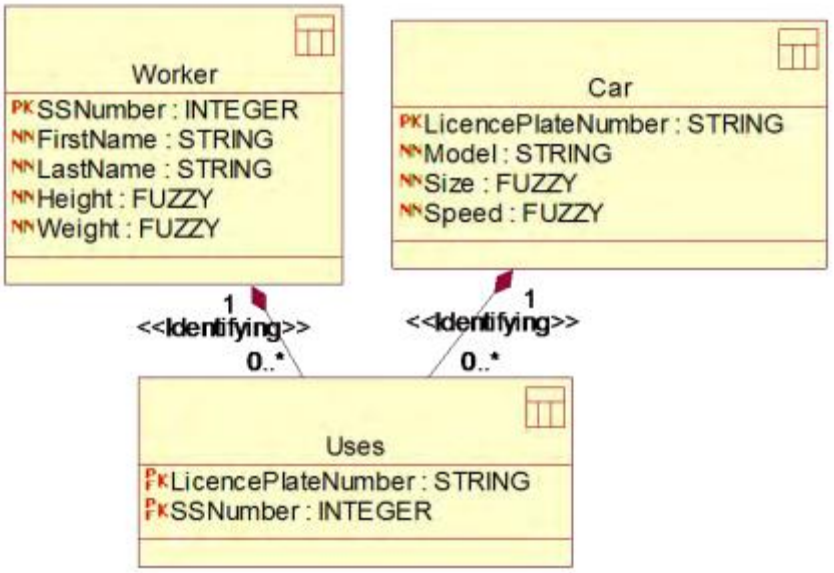


Fig. 10. Example fuzzy data model.

the corresponding Figure 10. shows fuzzy-relational data model. At the top of the figure are the tables Worker, Car, and Uses. Except for the data type of fuzzy columns, they are the same as before. They are of the type INTEGER in this model. Furthermore, they became foreign keys that originated from the table FuzzyValue's attribute ValueID. It extends this model with some additional tables that make up the fuzzy meta data model in order to represent these fuzzy values in the database. The table IsFuzzy simply keeps track of whether or not an attribute is fuzzy. All database attribute names are stored here, along with the table and attribute names (attributes TableName and AttributeName), as well as whether or not the attribute is fuzzy (value of the attribute IsFuzzy is 1) or not (value of the attribute IsFuzzy is 0). A link between the fuzzy data model and the fuzzy data meta model is represented by the table FuzzyValue. Every fuzzy value in every table is a foreign key that refers to the table FuzzyValue's primary key, ValueID. As a result, every record in the database with a fuzzy value has one record in the table FuzzyValue. From the table FuzzyType, the attribute Code is a foreign key. This table contains the names of all the different types of fuzzy values that can be used in the model. [19].

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.

There is a separate table in the meta model for each value in this list that stores data for all fuzzy values of a specific fuzzy type. ValueID, a foreign key from the table FuzzyValue, is present in each of these tables. As a result, depending on the type of fuzzy attribute, the value is stored in one of these tables. In the table FuzzyValue, the attribute ForValueID is a foreign key that represents a recursive relationship and refers to the FuzzyValue table's primary key. Linguistic labels are represented by this attribute. If the type of the attribute it represents is linguisticLabel, it has a value other than null. As previously stated, linguistic labels are nothing more than names for previously defined fuzzy values. If an attribute is a linguistic label, its name is stored in the table LinguisticLabel in this way. The value of ValueID of a fuzzy value that this linguistic label represents is assigned to the attribute ForValueID in this case. It is concluded that two records in the table FuzzyValue are required to represent a linguistic label. [19].

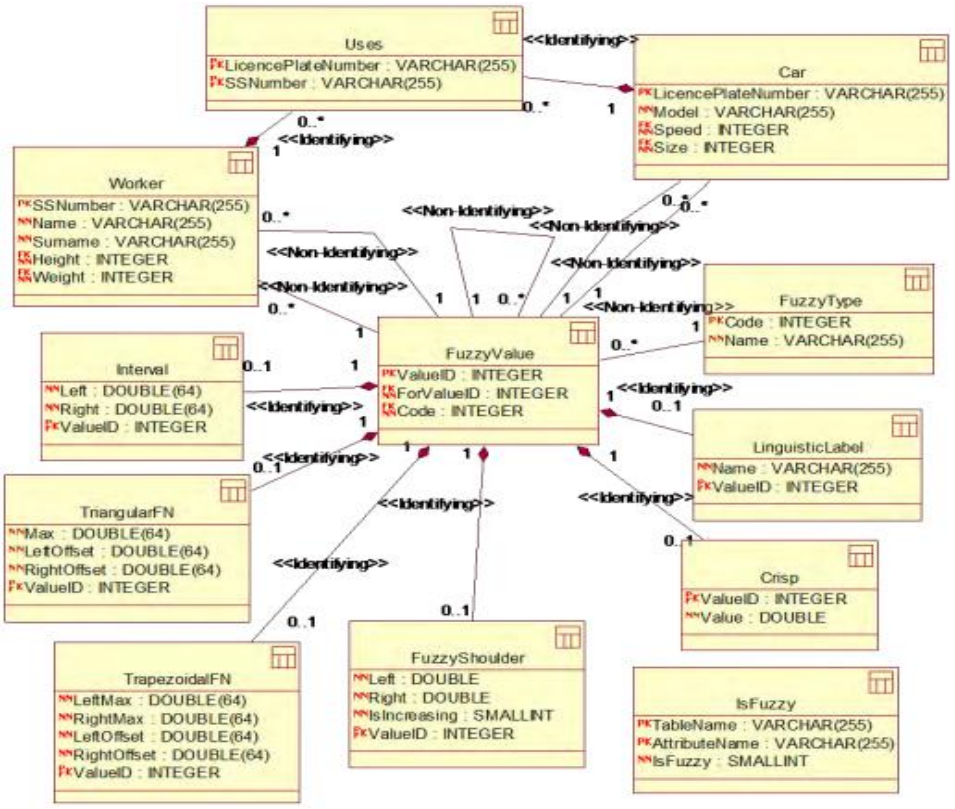


Fig. 11. Fuzzy relational data model.

Chapter Two: Literature review

The requirements set during the CASE tool's modeling process includes the functions for simplified building of a fuzzy relational data model, as well as functions for its transformation to SQL script. The goal was to create a CASE tool that could be used to visual model and manage all aspects of a fuzzy relational data model, including tables, attributes, fuzzy data types, and relationships. The GUI of the CASE tool is similar to that of all modern tools of this type that allow for the modeling of classical relational models. As a result, the details are not described here. All automation related to key migration through relationships is included in the model building process. This feature includes the ability to delete cascades, migrate keys during the relationship creation process, and detect and prevent circular references. Furthermore, the CASE tool is required to facilitate the creation of SQL scripts for the specific database management system. In this sense, it was necessary to include capabilities for specifying the data types used by the DBMS, as well as rules for mapping the types used in the model (including fuzzy data types) to these types. [19]. The Java programming language and the Swing platform for the GUI are used to create the CASE tool for this fuzzy relational model development (Fig. 12). It's available at http://www.is.pmf.uns.ac.rs/fuzzydb, along with the source code and an accompanying UML model.

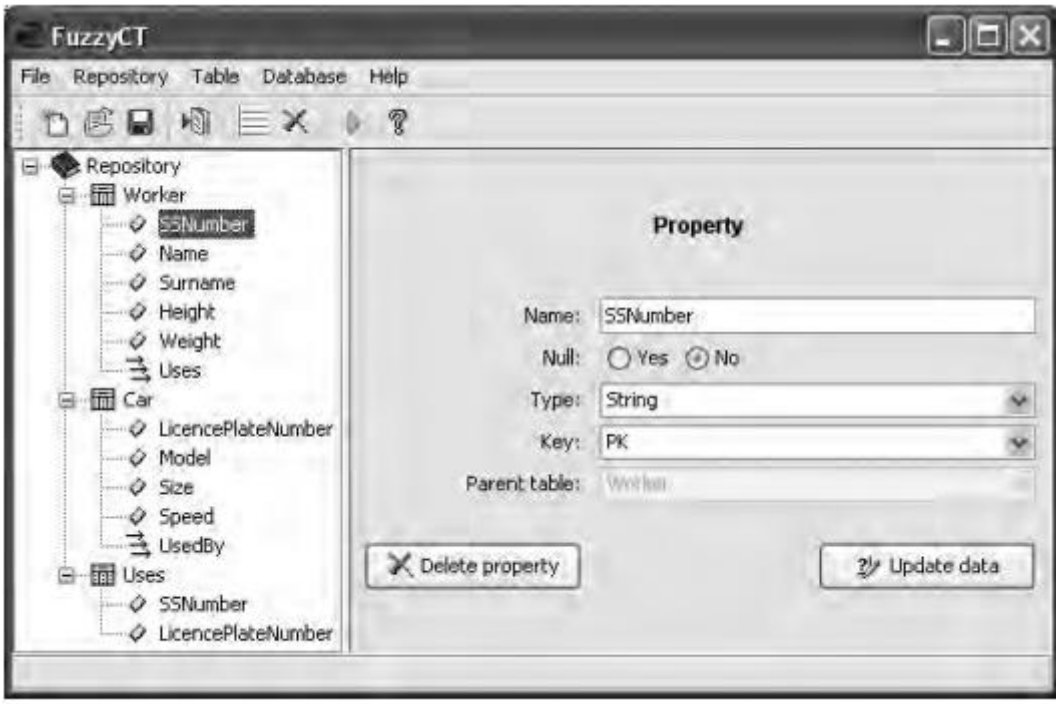


Fig. 12. The main window.

As shown in Fig. 12. the menu, toolbar, navigation tree, main panel, and status bar are the five parts of the main window, all of the commands in the CASE tool are accessible from the menu and toolbar. The user can manage the data model, specify data types in the model, and their mapping to SQL data types using those commands. Fuzzy is the only data type available by default. Crisp data types must be explicitly defined, along with their mapping to SQL data types specific to the database management system. The navigation tree is a visual representation of data in the repository about the model. A set of tables can be found in the repository, and the tables contain attributes and relationships (Fig. 12). To the left of the navigation tree is the main panel. Its content is determined by the navigation tree element selected. The panel that allows editing of that element is defined for each type of element in the navigation tree. At the bottom of the application window is a status bar with some useful information.[19].

2.4 Conclusion

In conclusion, this chapter has provided an overview of the literature review, beginning with a background that explains fuzzy logic and how it works, focusing on the main concepts of fuzzy logic, such as fuzzy sets and classical sets, fuzzy numbers, linguistic variables and values, and hedges, as well as the most commonly used membership function in this field. Furthermore, some problems with other proposed approaches have been identified, as well as a number of related works.

Chapter Three: Methodology

3.1 Introduction

In the previous chapter we presented a background to the concept of fuzzy logic, its main ideas, and differences between them, as well as some issues with previous works and a number of other works that are related to ours.

The software requirements specification will be presented in section 3.2, followed by the system architecture in section 3.3. In the software requirements specification, we will provide the user characteristics and user requirements, which will include both functional and non-functional requirements.

3.2 Software Requirements Specification

3.2.1 Scenarios

First Scenario: The user will log in to a user interface that is specifically designed for running human-like fuzzy queries. The user is then taken to the front page, where they can create their own profile, which includes entering their linguistic variables, as well as their own linguistic values and fuzzy numbers for each linguistic value. Finally, there are the Fuzzy query services, which allow users to create fuzzy query statements using standard SQL syntax. Following the execution of the generated fuzzy query statement, the result will be displayed to the user.

Second Scenario: Our fuzzy querying system will be integrated into another full-system, and the user interface will be integrated and designed to match the look of the user system, making it appear as if it is a part of it. This will allow the user to create, modify, or delete a linguistic variable with a set of linguistic values defined over it and its fuzzy numbers, and provide the user with the ability to generate human-like fuzzy queries with ease.

Example:

First we add the linguistic variable (StudentGrades), then we define a set of linguistic values over this linguistic variable (StudentGrades).

Which will be:

1. Excellent
2. Very_Good
3. Good
4. Pass
5. Failed

Then we choose the function for this linguistic variable, which is triangular, after that we enter the fuzzy numbers for each fuzzy value as shown in Table 3.

Variable	Values	Function	Xmin	Xmax	X1	X2	X3	X4
StudentGrades	Excellent	Triangular	0	100	80	90	100	0
StudentGrades	Very_Good	Triangular	0	100	70	80	90	0
StudentGrades	Good	Triangular	0	100	60	70	80	0
StudentGrades	Pass	Triangular	0	100	50	60	70	0
StudentGrades	Failed	Triangular	0	100	0	50	60	0

Table. 3. A set of linguistic values over a Linguistic Variable.

Finally, as shown in Fig.13, we write the fuzzy expressions as a condition in the where clause of a select statement. This query statement was designed to display each student's Id, grade, and a matching degree indicating how good that student is.

```
SELECT StudentID, StudentGrades,
            StudentGrades.Excellent( Student )
FROM    StudentScores
WHERE StudentsGrades.Excellent( Student ) > 0.5
```

Fig. 13. Using a fuzzy expression in where clause.

This query will return all the students who have the matching degree of excellence by 50% or more.

3.2.2 User Characteristics

In order to use our system effectively, the user must have a strong background in computers as well as the manipulation of linguistic variables and their values. The user of this system is most likely an employee who will be in charge of dealing with this system and who has a bachelor's degree in computer field and computer systems.

3.2.3 User Requirements

In terms of requirements, the user will have two types of requirements: functional requirements and non-functional requirements.

- **Functional Requirements**

Number	Functional Requirement
1	User login to the system
2	User logout from the system
3	User create linguistic variables
4	User modify linguistic variables
5	User delete linguistic variables
6	User enters the linguistic values for the linguistic variables
7	User modify the linguistic values for the linguistic variables
8	User chooses the function
9	User enters the fuzzy numbers for each linguistic value
10	User modify the fuzzy numbers for each linguistic value
11	User execute and generate the fuzzy query

Table. 4. Functional requirements.

- **Non-Functional Requirements**

Non-Functional Requirement	Description
Performance	• Due to the fact that the objects are stored in the database, the queries should execute significantly faster. • The time required to generate the fuzzy queries will be less than 5 seconds, as they will generate only statements and will be stored directly in the database management system.
Usability	• The user interface will be straightforward and easy to understand, with no complications.
Reliability	• Every fuzzy query statement will be generated correctly and without errors by the system. • The generated fuzzy query statement will be executed without any errors.
Security	• The database management system will be protected in a secure manner. • Since the user profile contains all of the user's personal information, a strong password is required to access the user interface.

Table. 5. Non-Functional requirements.

3.3 System Architecture

In this section we will explain the system architecture which is divided into two main components which are the user interface and the database management system.

3.3.1 User Interface

The user interface will focus mainly on allowing the user to login and create their own profile, and it will be divided into three parts

- **Login page**

This part of the user interface is where the user enters his username and password and get authenticated in order to access the system.

- **Linguistic variables**

This is the part of the user interface is where the user enters the name and description of his linguistic variables.

- **Linguistic values**

This part is in control of defining a set of linguistic values for a particular linguistic variable. For each predefined linguistic variable to be used when constructing a fuzzy SQL statement, a set of linguistic values can be easily defined.

- **Fuzzy SQL query generator**

This section allows the user to generate fuzzy query statements using traditional SQL syntax. The result is displayed after the generated fuzzy query statement has been executed then it will be stored in the DBMS as an stored object.

3.3.2 Database Management System

Our Database Management System (DBMS) is a software system that stores, retrieves, and executes data queries. As shown in figure 14 below, this DBMS acts as an interface between an end-user and a database, allowing users to create, read, update, and delete data in the database, as well as storing objects in the DBMS, which in this case will be the generated fuzzy SQL queries. This approach will allow for faster execution of the fuzzy SQL queries because they will be stored in the DBMS and will run instantly.

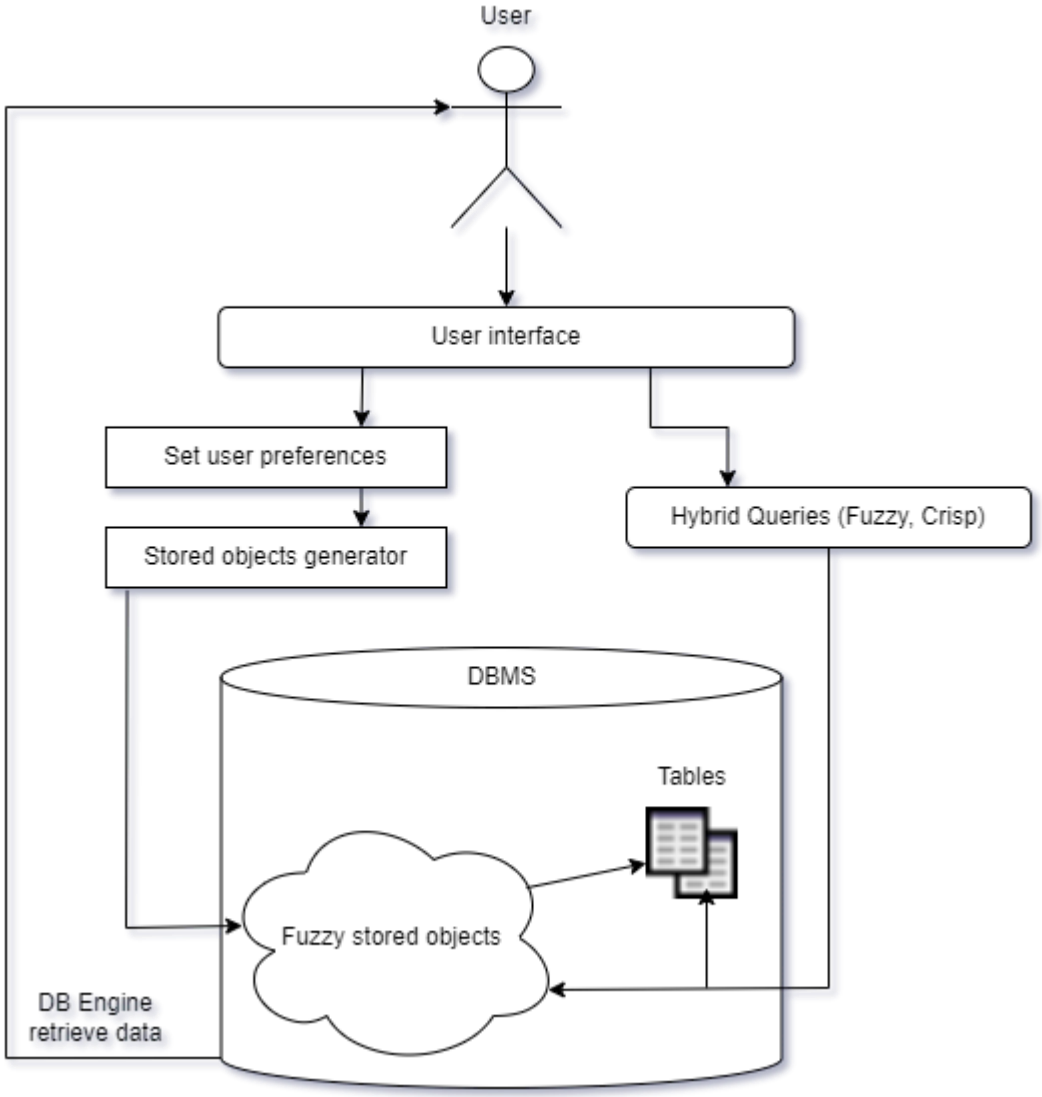


Fig. 14. The architecture of the Fuzzy query system.

3.3.3 Use Case Diagram

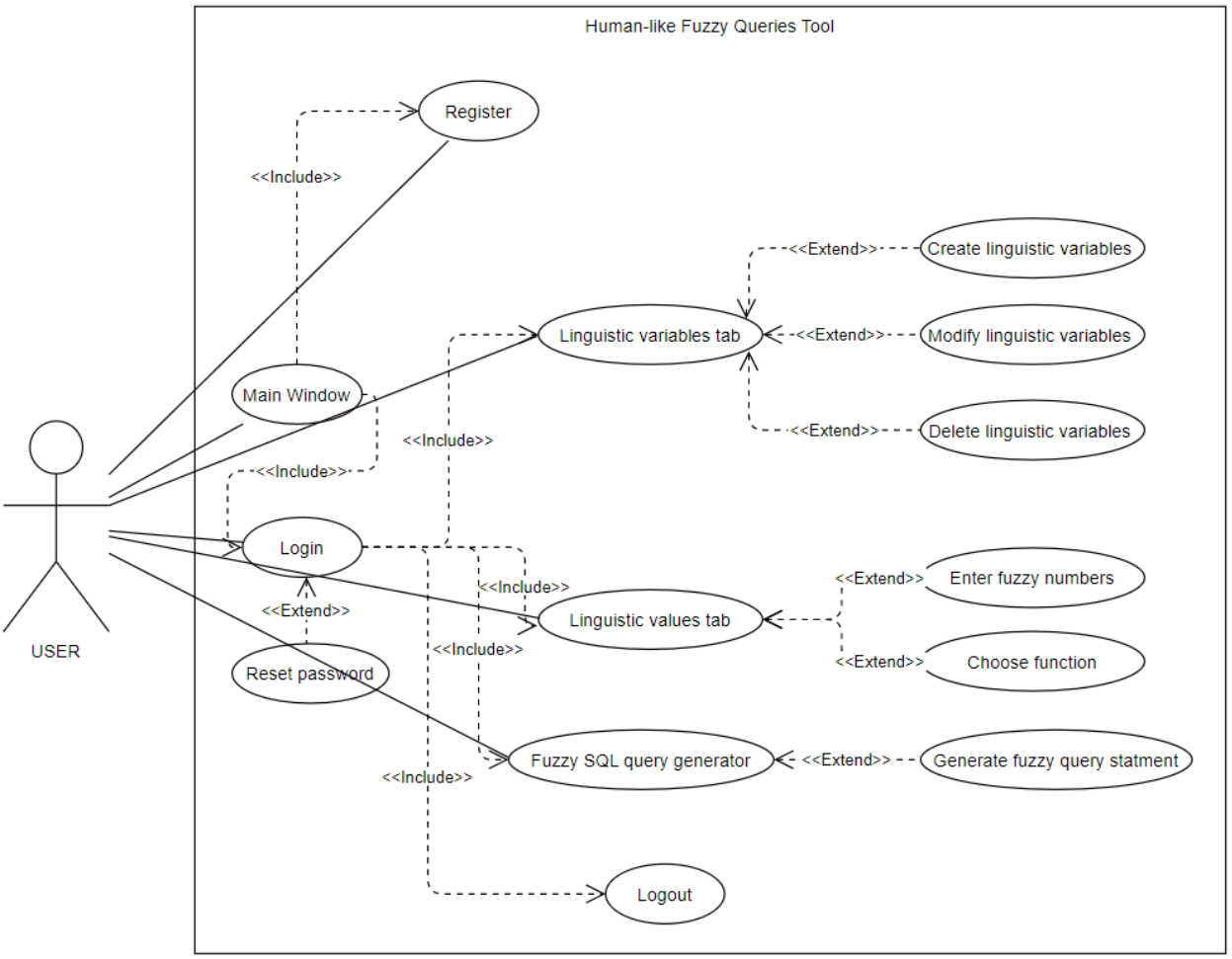


Fig. 15. Use case diagram.

3.4 Conclusion

To conclude this chapter, we have provided an introduction to the methodology, outlining what this chapter entails, followed by an explanation of the software requirements specification, which includes scenarios and an example of the system, as well as user characteristics and user requirements, which are divided into functional and non-functional, and finally, the system architecture, which includes the user interface and database management system.

References

[1] Mishra, J.: Fuzzy Query Processing. International Journal of Research and Reviews in Next Generation Networks 1(1) (March 2011).

[2] Grissa, A., Ben Hassine, M.: New Architecture of Fuzzy Database Management Systems. The International Arab Journal of Information Technology 6(3) (July 2009).

[2] Qi, Y., et al.: Efficient Processing of Nested Fuzzy SQL Queries in a Fuzzy Database. IEEE Transactions on Knowledge and Data Engineering 13(6) (November/December 2001).

[3] B. Tsikos, "Electricity Load Forecasting" Ph.D. dissertation, Univ. of Pennsylvania, BCE Dept., Philadelphia, 1987.

[4] Abbaci, K., Lemos, F., Hadjali, A., Grigori, D., Liétard, L., Rocacher, D., Bouzeghoub, M.: Selecting and Ranking Business Processes with Preferences: An Approach Based on Fuzzy Sets. In: Meersman, R., Dillon, T., Herrero, P., Kumar, A., Reichert, M., Qing, L., Ooi, B.-C., Damiani, E., Schmidt, D.C., White, J., Hauswirth, M., Hitzler, P., Mohania, M., et al. (eds.) OTM 2011, Part I. LNCS, vol. 7044, pp. 38–55. Springer, Heidelberg (2011).

[5] Singh, K., et al.: Study of Imperfect Information Representation and FSQL processing. International Journal of Scientific & Engineering Research 3(5) (May 2012).

[6] Garg, A., Rishi, R.: Querying Capability Enhancement in Database Using Fuzzy Logic. Global Journal of Computer Science and Technology 12(6) (Version 1.0 March 2012).

[7] Wahidin, I.: Fuzzy Control (2007)

[8] Gadallah, A., Hefny, H.: An Efficient Database Query Processing Tool Based on Fuzzy Logic. In: The 37th Annual Conference on Statistics and Computer Science (2002)

[9] Shawky, A., et al.: FRDBM: A Tool For Building Fuzzy Relational Databases. The Egyptian Computer Journal (2006)

References

[10] Galindo, J., et al.: Handbook of Research on Fuzzy Information Processing in Databases. Chapter XI FSQL and SQLf: Towards a Standard in Fuzzy Databases (2008)

[11] Bosc, P., Pivert, O.: SQLf Query Functionality on Top of a Regular Relational Database Man-agement. In: Pons, O., Vila, M.A., Kacprzyk, J. (eds.) Proceedings of Knowledge Manage-ment in Fuzzy Databases. STUDFUZZ, vol. 39, pp. 171–190. Springer, Heidelberg (2000)

[12] Moore, S.: Oracle Database PL/SQL Language Reference, 12c, p. 1 (2014)

[13] Elmasri, R., Navathe, S.B.: Fundamental of Database Systems (2011)

[14] MLA 9th Edition (Modern Language Assoc.)
Radim Belohlavek, and George J. Klir. Concepts and Fuzzy Logic. The MIT Press, 2011.

[15] APA 7th Edition (American Psychological Assoc.)
Radim Belohlavek, & George J. Klir. (2011). Concepts and Fuzzy Logic. The MIT Press.

[16] L. A. Zadeh, “Fuzzy logic,” IEEE Comput. Mag., pp. 83-93, Apr. 1988.

[17] Rachid Mama and Mustapha Machkour, " Fuzzy querying with SQL: Fuzzy view-based approach," Journal of Intelligent & Fuzzy Systems 40 (2021) 9937–9948 DOI:10.3233/JIFS-202551 IOS Press.

[18] Amel Grissa Touzi and Mohamed Ali Ben Hassine, "New Architecture of Fuzzy Database Management Systems," The International Arab Journal of Information Technology, I b/. 6, No. 3, July 2009.

[19] Srđan Škrbić1, Miloš Racković1, and Aleksandar Takači2 " Towards the Methodology for Development of Fuzzy Relational Database Applications," Faculty of Science, Trg Dositeja Obradovića 3, 21000 Novi Sad, Serbia {shkrba, rackovic}, Faculty of Technology, Bulevar Cara Lazara 1, 21000 Novi Sad, Serbia.

References

[20] Gadallah, A., Hefny, H., Adel A. Sabour: Flexible Querying of Relational Databases: Fuzzy Set Based Approach (2014).