Flexible Querying of Relational Databases: Fuzzy Set Based Approach

Adel A. Sabour, Ahmed M. Gadallah, and Hesham A. Hefny

Institute of Statistical Studies and Research, Cairo University
1adelsabour@gmail.com, ahmgad10@yahoo.com, hehefny@ieee.com

Abstract. This paper presents a flexible fuzzy-based approach for querying relational databases. Although, many fuzzy query approaches have been proposed, there is a need for a more flexible, simple and human-like query approach. Most of previously proposed fuzzy query approaches have a disadvantage that they interpret any fuzzy query statement into a crisp query statement then evaluate the resulted tuples to compute their matching degrees to the fuzzy query. The main objective of the proposed fuzzy query approach of this paper is to overcome the above disadvantage by evaluating each tuple directly through the use of stored database objects namely packages, procedures and functions. Consequently, the response time of executing a fuzzy query statement will be reduced. This proposed approach makes it easy to use fuzzy linguistic values in all clauses of a select statement. The added value of this proposed approach is to accelerate the execution of fuzzy query statements.

Keywords: Fuzzy Query, Fuzzy Logic, InformationRetrieval, Fuzzy SQL.

1 Introduction

Structured Query Language (SQL) is a very essential language for querying relational databases. It manipulates and retrieves data which is crisp and precise by nature. In contrary, it is unable to respond to human-like queries which are uncertain, imprecise and vague in nature. Almost, human queries have a lot of vagueness and ambiguity due to using his/her subjective linguistic words. For example, excellent students have different definitions that depend on each person searching for them. However, while applying one's thoughts as a query in terms of linguistic words into the database, a lot of problems are experienced due to the inefficiency of DBMS to handle such queries. Consider the query "retrieve the names and addresses of the university students who have height around the ideal height for a handball player". This query cannot be expressed and manipulated directly by a traditional SQL statement. In contrary, it can be expressed and manipulated easily through a fuzzy query statement in a very flexible and human-like manner based on the ideas of fuzzy set theory. Although classical SQL has great querying capabilities, it lacks the flexibility to support human-like queries. Human-like queries depend essentially on manipulation of linguistic values rather than numeric ones. Linguistic values such as: short, tall, hot and calls human concepts that cannot be handled as authorized attributes values in standard SQL statements. Moreover, standard SQL adopts classical Boolean expressions with crisp logical connectors which are too rigid and very limited in manipulation of linguistic values or linguistic modifiers such as: about, nearly, fairly, etc. which can be represented easily based on the concept of fuzzy sets [1].

The rest of this paper is organized as follows: section 2 presents previous works of fuzzy querying in databases. The proposed fuzzy query approach is introduced in Section three. Section 4 addresses a developed tool based on the proposed approach with an illustrative case study. The conclusion is addressed in section 5.

2 Previous Works of Fuzzy Querying in Databases

This section presents several architectures and applications that have been proposed and developed for allowing fuzzy queries of databases. Unfortunately, most of previously developed fuzzy SQL architectures have set of drawbacks. Both of previously proposed approaches in [1] and [2] have the inability for dealing with multi subjective views when interacting with multiusers. Instead, preferences are used to help in reducing the amount of information returned in response to user queries [4]. Also, the proposed approachesin [11], [2] and [1] lack of a standardized format in writing statements that is each tool has its own syntax.On the other hand, most of the proposed approaches depend mainly on using a time consuming parser/translator to check and convert fuzzy queries to crisp SQL queries like in [1],[2], [10] and [11]. Also, such approaches define a Meta base named Fuzzy Meta-knowledge Base (FMB) that includes a set of tables to store all necessary information to describe and manipulate fuzzy attributes and terms [2].Such FMB must be accessed each time a fuzzy query statement is generated. This operation is essential in order to obtain the definitions of each used fuzzy term in order to complete the processing of the generated fuzzy query statement which is a time consuming[5], [10] and [11].This slows down the process of querying because each generated fuzzy query must be analyzed and translated into a standard SQL statement respecting the definitions of all used fuzzy terms or operators stored in FMB. [7]. Also, the approaches proposed in[2],[5] and [10] process nested and correlated fuzzy queries inefficiently [3].

3 The Proposed Fuzzy Query Approach

This section presents the proposed fuzzy query approach that has the architecture shown in Fig. 1. It aims mainly to allow the generation and manipulation of more flexible human-like queries. This approach aims mainly to overcome the drawbacks mentioned in previous works section. It deals with multi subjective views with multiusers and it fully depends on PL/SQL statements. Accordingly, all generated fuzzy queries will be Standard SQL compatible without any need for an interpreter/parser. The proposed approach allows generating fuzzy query statements using the standard SQL language. Accordingly, a fuzzy term can be used easily by write its package name followed by a dot and its function or procedure name with its defined parameters if required. On the other hand, using a procedural language, as PL/SQL, has better performance because data and queries are directly managed and manipulated by RDBMS without the need for intermediate software programs [10]. Also, the proposed approach will benefit from the advantages of using PL/SQL [12] which include: Tight Integration with SQL, High Performance by reducing traffic between the application and the database, High Productivity, Portability on any operating system, Scalability, Manageability and Support for Object-Oriented. Also, using database programming language approach does not suffer from the impedance mismatch that includes [13]:

- Existence of differences between programming language and database models.
- The need to have a binding for each programming language that determines for each attributes type the compatible programing language type.
- The need for a binding that maps the results multi-set of tuples into a corresponding data structure like a record set or a cursor.

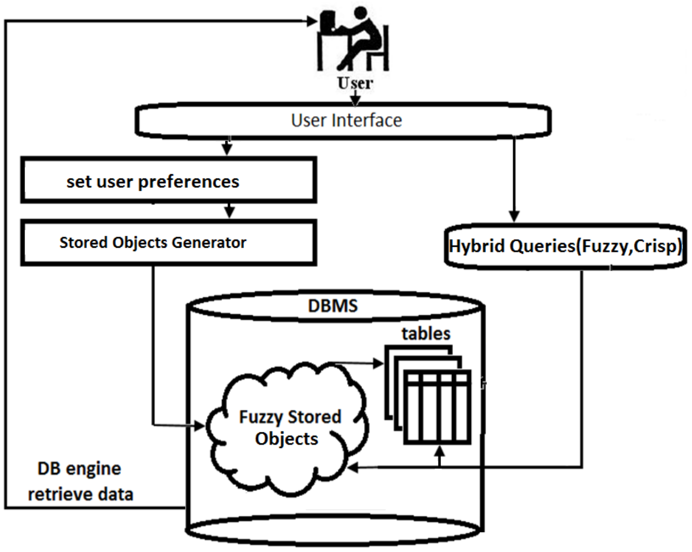


Fig. 1. Thearchitecture of the proposed Fuzzy query approach

The proposed approach depends mainly on the idea of fuzzy set. Commonly, a fuzzy set is a set with smooth boundaries. Accordingly, fuzzy set theory generalizes classical set theory to allow partial membership of its elements to the set, and it was developed to cover areas that cannot be covered by classical set theory. A fuzzy set A in the universe of discourse U is characterized by the membership function μ_A given by μ_A : U → [0, 1] and A is defined as the set of ordered pairs A = {(x, μ_A(x)): x $\in$ U}, where μ_A (x) is a membership function that determines the membership degree of element x in the set A. Commonly, a fuzzy set of continuous universe of discourse is defined as a membership function. A membership function maps an element in the universe of discourse U to some value in the interval [0, 1] that is $\mu_A : X \rightarrow [0,1]$.

A full membership in a fuzzy set has a value of 1 while a membership of 0 indicates excluding the element from the fuzzy set at all. A membership between 0 and 1 indicates a partial membership to the set. As traditional crisp set theory, fuzzy set theory includes a set of operations like complement, intersection and union. Yet, fuzzy set operations are not uniquely defined i.e. as membership functions but they are also context and user dependent.

3.1 The Processing Scenario of the Proposed Approach

- The proposed fuzzy query approach aims mainly to support human-like queries which almost contain linguistic terms and fuzzy connectors. Such linguistic terms include linguistic variables, linguistic values, fuzzy hedges and fuzzy numbers. The proposed approach enhances the definition and manipulation of such linguistic terms. A user can easily define his own linguistic terms which may be stationary or non-stationary using suitable fuzzy membership functions like triangular, trapezoid, S-shape, .etc. Stationary linguistic terms have fixed definition. On the other hand, non-stationary linguistic terms have dynamic definitions which may change from time to time like "excellent students" in a very simple course and in a very complex course. Each defined linguistic term will be stored as a database object namely stored function within a related stored package. In consequence, a user can use any of his defined linguistic terms in any clause in a select statement as using a user defined stored function. Accordingly, the defined linguistic terms can be used easily in even nested, complex and correlated query statements without the need for complex procedures to execute the query statement. A user can interact with a developed tool based on the proposed approach as follows:
- A user interacts with the tool via a graphical user interface to define the user own linguistic terms describing the user preferences like linguistic variables, linguistic values, fuzzy connectors, fuzzy numbers and hedges. All of user linguistic terms will be stored as database objects resembling the user profile. Hence, there is no need for additional database.
- The tool generates PL-SQL statements that create database stored objects (such as packages, procedures, functions, views, etc.) for the defined linguistic terms. These objects contain the description of its linguistic term instead of using Fuzzy Meta Knowledge Base (FMB). Accordingly, the execution of a fuzzy query statement will not need more access for additional database to obtain the definition of the used linguistic terms. Hence, the proposed approach saves the processing time and reduces the response time.
- Finally, the user can write a fuzzy query statement or generate it using the tool query builder. It is obvious that, the fuzzy query syntax is identical to the standard SQL language due to the use of stored database objects for representing the fuzzy terms. In other words, any generated fuzzy query statement agrees with the standard SQL language. So, there is no need for an analyzer or interpreter to map a fuzzy query statement to a standard one then evaluate the resulted tuples with additional algorithms for manipulating the used linguistic terms. The resulted tuples are associated with matching degrees to the generated fuzzy query.

4 A Fuzzy Query Tool Based on the Proposed Approach

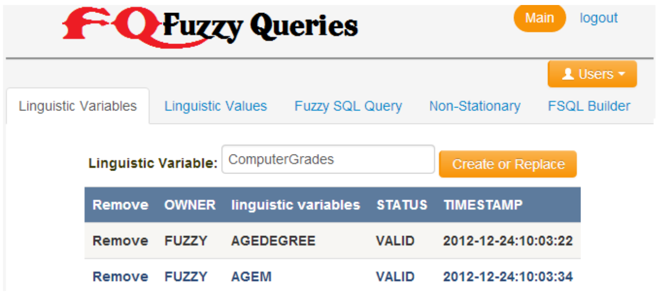
This section describes a developed fuzzy query tool (FQ) based on the proposed approach.

4.1 Login to FQ Service

At the first time the user aims to interact with the tool, he/she must create an account with a user name and a password. After that, the user can define his/her own linguistic terms via the tool GUI. Consequently, each created stored database object within the user session becomes part of the user fuzzy profile. Also, the user can generate a fuzzy query statement using his/her predefined linguistic terms and execute it.

4.2 Linguistic Variables Service

This service allows creating, modifying or deleting a linguistic variable that has a set of linguistic values that will be defined over it. The user enters just the name of the linguistic variable and a description for it. For example, a linguistic variable may be a computer grades, temperature degree, height or salary as shown in Fig.2.

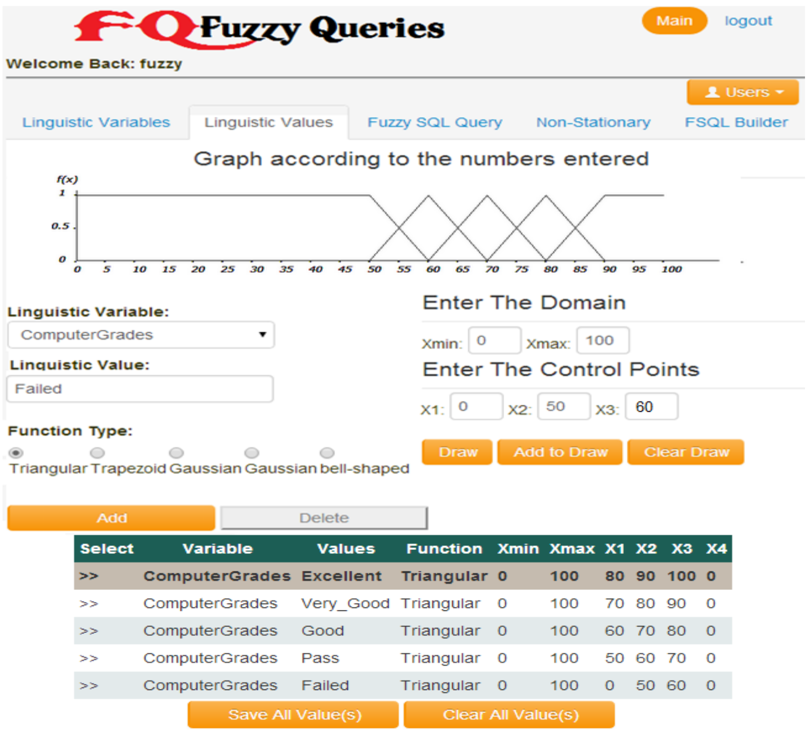


Remove	OWNER	linguistic variables	STATUS	TIMESTAMP
Remove	FUZZY	AGEDEGREE	VALID	2012-12-24:10:03:22
Remove	FUZZY	AGEM	VALID	2012-12-24:10:03:34

Fig. 2. Linguistic Variables Screen

4.3 Linguistic Values Services

This service is responsible for defining a set of linguistic values over a specific linguistic variable. A set of linguistic values can be defined easily for each predefined linguistic variable to be used when constructing a fuzzy SQL statement. For example, the user can define the linguistic values “Tall”, “Short” and “Medium” for the “Height” linguistic variable. Each linguistic value can be defined in a very flexible representation, that the user can select the more suitable membership function and modulate its control point’s values to be more closed to the meaning of the linguistic value. Through a graphical user-friendly representation, the user can check the linguistic values definitions and its overlapping for a specific linguistic variable. The graphical representation helps the user to make the effective modifications in the selected membership function to satisfy the meaning of the linguistic value, as shown in Fig.3.



Linguistic Variable: ComputerGrades

Linguistic Value: Failed

Function Type: [✓ Triangular] Trapezoid Gaussian Gaussian bell-shaped

Enter The Domain

Xmin: 0 Xmax: 100

Enter The Control Points

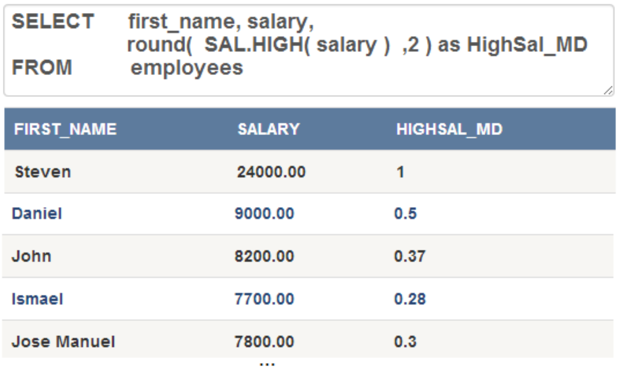
X1: 0 X2: 50 X3: 60

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

Fig. 3. Defining a set of linguistic values over a Linguistic Variable

4.4 Fuzzy SQL Query Service

This service enables the user to generate fuzzy query statements using the syntax of traditional SQL language. After the execution of the generated fuzzy query statement, the result it displayed as shown in Fig. 4. This query statement is generated to obtain each employee first name, salary and a matching degree specifying how much such salary is high.



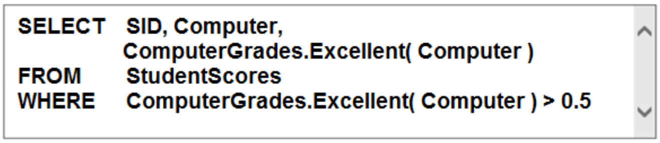
```
SELECT    first_name, salary,
          round(  SAL.HIGH( salary )  ,2 ) as HighSal_MD
FROM      employees
```

FIRST_NAME	SALARY	HIGHSAL_MD
Steven	24000.00	1
Daniel	9000.00	0.5
John	8200.00	0.37
Ismael	7700.00	0.28
Jose Manuel	7800.00	0.3

...

Fig. 4. A fuzzy query statement and the result of its execution

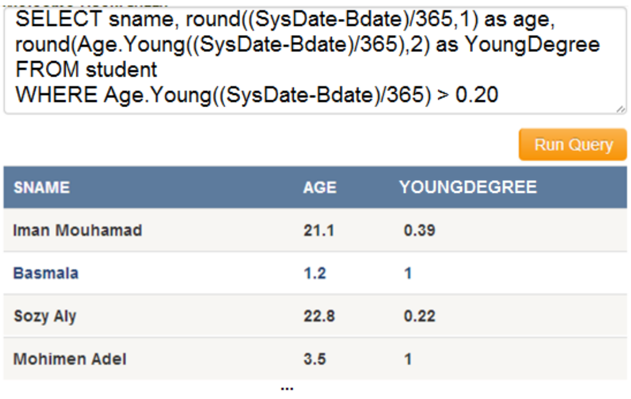
The proposed approach supports to write fuzzy expressions as a condition in the *where* clause in a select statement as shown in Fig.5.Such query statement intended to show each student Id, computer grade and a matching degree specifying how much such student is excellent in computer subject.



```
SELECT   SID, Computer,
         ComputerGrades.Excellent( Computer )
FROM     StudentScores
WHERE    ComputerGrades.Excellent( Computer ) > 0.5
```

Fig. 5. Using a fuzzy expression in where clause

Also, it allows passing fuzzy expressions as arguments to a function or a stored procedure in a select statement, as shown in Fig. 6. The generated query statements in Fig. 6 aims to retrieve each student name, age and a matching degree specifying how much such age represents a Young one. The resulted tuples must have matching degrees greater than the specified threshold value 0.20.



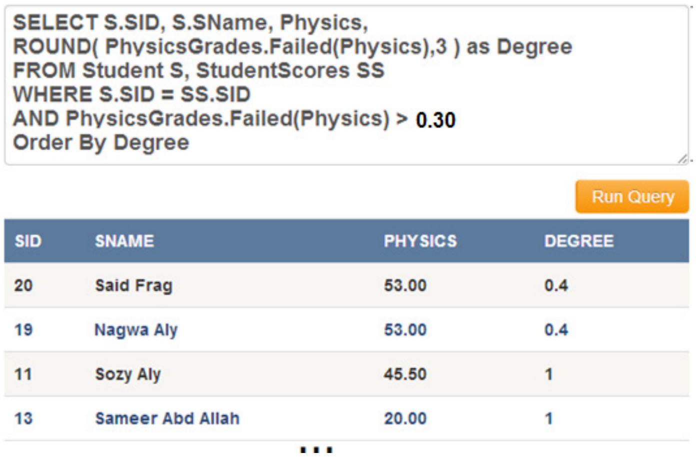
```
SELECT sname, round((SysDate-Bdate)/365,1) as age,
round(Age.Young((SysDate-Bdate)/365),2) as YoungDegree
FROM student
WHERE Age.Young((SysDate-Bdate)/365) > 0.20
```

SNAME	AGE	YOUNGDEGREE
Iman Mouhamad	21.1	0.39
Basmala	1.2	1
Sozy Aly	22.8	0.22
Mohimen Adel	3.5	1

...

Fig. 6. Passinga fuzzy expression as an argument to a function

On the other hand, complex queries including joining of two or more tables are supported. The resulted tuples can be sorted using their matching degree to the specified fuzzy criteria in the generated fuzzy query statement, as shown in Fig. 7.This query statement aims to retrieve each student Id, student name, physics grade and a matching degree specifies how much such student if failed in physics.



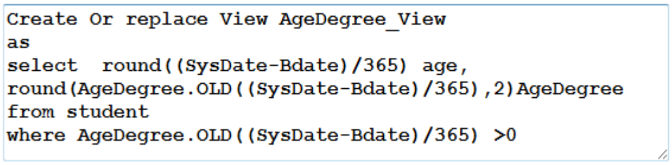
```
SELECT S.SID, S.SName, Physics,
ROUND( PhysicsGrades.Failed(Physics),3 ) as Degree
FROM Student S, StudentScores SS
WHERE S.SID = SS.SID
AND PhysicsGrades.Failed(Physics) > 0.30
Order By Degree
```

SID	SNAME	PHYSICS	DEGREE
20	Said Frag	53.00	0.4
19	Nagwa Aly	53.00	0.4
11	Sozy Aly	45.50	1
13	Sameer Abd Allah	20.00	1

...

Fig. 7. The result of a complex fuzzy query statement sorted ascendingly

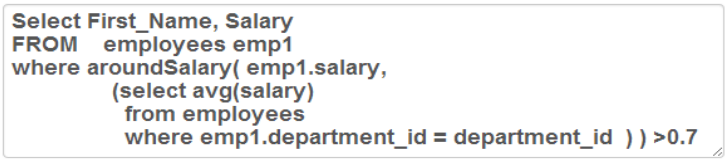
Also, it is available to creatingdatabase views based on fuzzy expressions, as shown in Fig. 8.



```
Create Or replace View AgeDegree_View
as
select  round((SysDate-Bdate)/365) age,
round(AgeDegree.OLD((SysDate-Bdate)/365),2)AgeDegree
from student
where AgeDegree.OLD((SysDate-Bdate)/365) >0
```

Fig. 8. Creating a database view based of fuzzy expressions

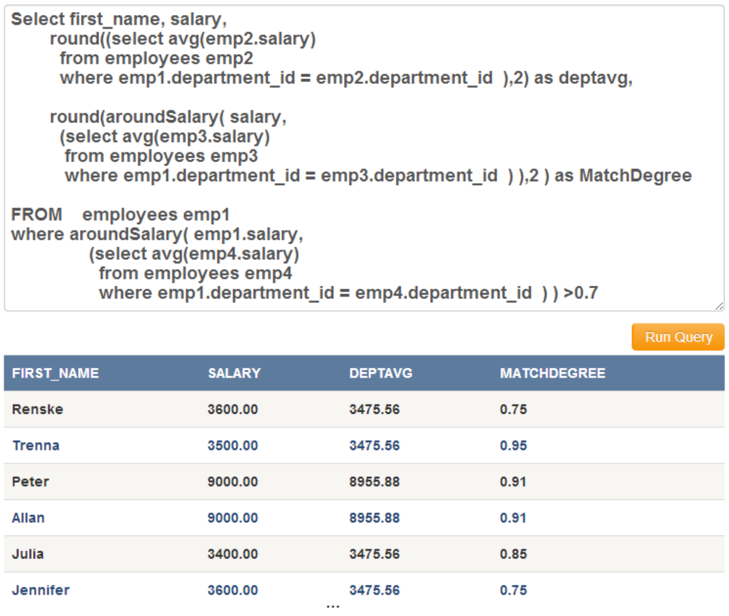
Correlated subqueries known as synchronized subqueries are also allowed by the tool. Fig. 9 shows a correlated query statement that retrieves each employee name and salary for each employee has salary around the average salary in his/her department respecting a threshold value of 0.7.



```
Select First_Name, Salary
FROM   employees emp1
where aroundSalary( emp1.salary,
          (select avg(salary)
           from employees
           where emp1.department_id = department_id  ) ) >0.7
```

Fig. 9. An example of a correlated Fuzzy query statement

In order to enhance the previous query statement to show each employee name, salary, the average salary of his/her department and a matching degree representing how much the employee salary is closed to such average, the fuzzy query statement shown in Fig. 10 is used. The result of such a query statement shows that the more closed salary to the average, the higher the matching degree.



```
Select first_name, salary,
       round((select avg(emp2.salary)
         from employees emp2
         where emp1.department_id = emp2.department_id  ),2) as deptavg,

       round(aroundSalary( salary,
         (select avg(emp3.salary)
          from employees emp3
          where emp1.department_id = emp3.department_id  ) ),2 ) as MatchDegree

FROM    employees emp1
where aroundSalary( emp1.salary,
               (select avg(emp4.salary)
                 from employees emp4
                 where emp1.department_id = emp4.department_id  ) ) >0.7
```

FIRST_NAME	SALARY	DEPTAVG	MATCHDEGREE
Renske	3600.00	3475.56	0.75
Trenna	3500.00	3475.56	0.95
Peter	9000.00	8955.88	0.91
Allan	9000.00	8955.88	0.91
Julia	3400.00	3475.56	0.85
Jennifer	3600.00	3475.56	0.75

...

Fig. 10. Another example of a more complex correlated fuzzy query statement

5 Conclusion

This paper presents a more simple fuzzy set based approach for flexible querying of relational databases. Commonly, many approaches have been proposed allowing fuzzy querying of relational databases. Yet, such approaches present complicated solutions which are two-fold. Firstly, they depend, mainly, on defining a fuzzy meta database (FMD) that is used for storing the definitions of all fuzzy terms that can be used. Accordingly, each generated fuzzy query statement needs to be analyzed to fetch each fuzzy term and brings its definition from FMD which is time consuming. If such approach depends on converting the generated fuzzy query to an equivalent traditional SQL query statement, it reads the definition and then passes it to its counterpart stored database object then process the query. Secondly, some other approaches needs to convert any generated fuzzy query statement to classical SQL statement by eliminating all used fuzzy terms. In consequence, the definition of such fuzzy terms, stored in FMD, is used to fuzzily evaluating the resulted tuples and assigning a matching degree for each resulted tuples. In contrary, the proposed approach depends mainly on storing all used fuzzy terms as database objects namely stored procedures and functions within packages in an organized fashion. Such stored database objects resembles the user fuzzy profile. Consequently, a fuzzy query statement can be generated directly as a traditional SQL statement using the fuzzy terms predefined as database objects. In consequent, there is no need for an analyzer or a translator to covert a generated fuzzy query into an executable query statement. Also, there is no need for a Fuzzy Meta-Knowledgebase that existed in most of previously proposed approaches for fuzzy queries. Hence, not only the complexity of allowing a fuzzy query is reduced but also the response time of executing such query. The proposed approach allows simple, complex, nested and correlated fuzzy query statements in a human-like fashion which almost contain linguistic terms and fuzzy connectors. Also, because linguistic terms are represented as a set of user defined stored functions; the proposed approach allows writing any simple, nested, correlated and complex fuzzy query statements as traditional select statements in a very flexible manner.

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)
3. 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)
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)
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 Management. In: Pons, O., Vila, M.A., Kacprzyk, J. (eds.) Proceedings of Knowledge Management 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)