International Journal of Computer Technology and Electronics Communication *(ISSN: 2320 – 0081)*
Volume 2– Issue-3, 2013

Fuzzy Query Executer using Ontology

Mr. Kumar Kaushik
ABES Engineering College,
Ghaziabad, India.
kumarkaushik26@gmail.com

Mr. Abhishek Kr. Gaur
Computer Science Dept.
ABES Engineering College,
Ghaziabad, India
abhishekgaur@abes.ac.in

***Abstract*— Relational Database Management System (RDBMS) can only handle crisp data and we already know that Structured Query Language (SQL) is a very powerful tool but can handle data which is crisp and precise in nature. It is unable to satisfy the needs for data which is uncertain, imprecise, inapplicable and vague in nature. The goal of this work is to use Fuzzy technique in RDBMS. But, Fuzzy Relational Database Management System (FRDB) requires complex data structures, in most cases, are dependent on the platform in which they are implemented. A solution that involves representing an FRDB using an Ontology as an interface has been defined to overcome this problem. A new Fuzzy Query Ontology is proposed in this dissertation with implementation. The implementation layer, which is responsible for parsing and translating user requests into the corresponding DB implementations in transparent, is required to establish communication between the Ontology and the relational databases management system (RDBMS).**

Keywords—fuzzy; Ontology; database; query;

I. INTRODUCTION

Relational Database can store only crisp, precise data, but can able to store fuzzy values. Complexity normally arises from uncertainty in the form of ambiguity. The computerized system is not capable of addressing uncertain, imprecise data and ambiguous issues whereas, the human have the capacity to reason "approximately". As a result, human, when interacting with the database, want to make complex queries that have a lot of vagueness present in it. So, database which is in use cannot store uncertain data. But in real situations, these are not crisp and deterministic and therefore, cannot be described precisely. The techniques based on the fuzzy set theory are very much useful while modeling the uncertainties especially, when the uncertainties are non-random in nature. The conventional database management system does not handle imprecise, incomplete or vague information such as very high, approximately some values. To triumph over this problem, the fuzzy database system has been introduced. And SQL Language works with conventional database, so in order to interact with fuzzy database we need to extend SQL into language which handle fuzzy data. The easiest way to do this is, to use classical relational databases and develop a front end that will allow fuzzy querying to the database. Here the underlying database will always be crisp. This paper is organized as follows: Fuzzy logic concepts is presented in section 2. Section 3 explains fuzziness in database. Section 4 analyses SQL limitations. Section 5 discusses the proposed framework for FSQL. Section 6 gives the implementation detail of criteria that follow.

II. FUZZY LOGIC CONCEPTS

A. *Fuzzy Logic*

Fuzzy set theory is the base of fuzzy logic. In this logic, the truth-value of a sentence (or satisfaction degree) is in the real interval [0, 1]. The value 0 represents completely false, and 1 is completely true. The truth-value of a sentence "s" will be denoted as μ(s).Fuzzy logic was developed as a mean to do reasoning under uncertainty. Many new approaches and theories treating imprecision and uncertainty have been proposed since fuzzy set was introduced by Zadeh [4].
Its characteristics are
1. Fuzzy truth values are expressed in linguistic terms.
2. Imprecise truth tables.
Fuzzy Logic is a problem-solving control system methodology that lends itself to implementation in systems ranging from simple, small, embedded microcontrollers to large, networked, multi-channel PC or workstation-based data acquisition and control systems. It can be implemented in hardware, software, or a combination of both. FL provides a simple way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information. FL's approach to control problems mimics how a person would make decisions, only much faster. Fuzziness can be defined as the vagueness concerning the semantic meaning of events, phenomenon or statements themselves. It is particularly frequent in all areas in which human judgment, evaluation and decisions are important [10].
A fuzzy set is almost any condition for which we have words: short men, tall women, hot day, cold climate, new building, ripe bananas, high intelligence, low speed, overweight, etc., where the condition can be given a value between 0 and 1. Fuzzy set 'A' over a universe of discourse X (a finite or infinite interval) within which the fuzzy set can take a value) is a set of pairs:

$$A = \{\mu A(x) / x : x \in X, \mu A(x) \in [0, 1] \in R\}$$

Where, μA(x) is called the membership degree of the element x to the fuzzy set A. This degree ranges between the extremes 0 and 1 of the dominion of the real numbers: μA(x) = 0 indicates that x in no way belongs to the fuzzy set A, and μA(x) = 1 indicates that x completely belongs to the fuzzy set A. Note that μA(x) = 0.5 is the greatest uncertainty point [5].

III. FUZZINESS IN DATABASE

We present in this section the basic concepts of FRDB.

A Fuzzy Relational Database (FRDB) is an extension of the relational database. This extension introduces fuzzy predicates under shapes of linguistic expressions that, at the time of a flexible querying, permits to have a range of answers (each with a member ship degree) in order to offer to the user all intermediate variations between the completely satisfactory answers and those completely dissatisfactory [1]. We can also say that a FRDB is defined as an enhanced relational database that allows fuzzy attribute values and fuzzy truth values; both of these are expressed as fuzzy sets [15]. The basic model of fuzzy relational databases is considered the simplest one, and it consists of adding a *grade*, normally in the [0,1] interval, to each instance (or tuple). This makes keeping database data homogeneity possible. Nevertheless, the semantic assigned to this grade will determine its usefulness, and this meaning will be utilized in the query processes [9].This last model constitutes an eclectic synthesis of the various models published so far with the aim of dealing with the problem of representation and treatment of fuzzy information by using relational DB.

The GEFRED model

The GEFRED model (GEneralised model Fuzzy heart Relational Database) has been proposed in 1994 by Medina et al. [11]. It is based on the generalized fuzzy domain and generalized fuzzy relation, which include classic domains and classic relations, respectively. To model the flexible queries and the concept of the fuzzy attributes, this paper uses an extension of SQL language called SQLf. We present in the following section this language.

a) *Linguistic Labels:* If an attribute is able of fuzzy treatment then linguistic labels can be defined on it. These labels will be preceded with the symbol $ to distinguish them easily. There are two types of labels and they will be used in different fuzzy attribute types: (1) Labels for attributes with an ordered underlined fuzzy domain (Fuzzy Attributes Type 1 and 2) and (2) Labels for attributes with a non ordered fuzzy domain (Fuzzy Attributes Type 3 and 4).

b) *Fuzzy Comparators*: Besides the typical comparators (=,>, etc.), FSQL includes fuzzy comparators. There are many comparators, the most used are: FEQ (Fuzzy Equal than), (NFEQ: Necessarily FEQ), FGT (Fuzzy Greater than), (NFGT: Necessarily FGT), FGEQ (Fuzzy Greater or Equal than), (NFGEQ: Necessarily FGEQ), FLT (Fuzzy Less Than), (NFLT: Necessarily FLT), FLEQ (Fuzzy Less or Equal than), (NFLEQ: Necessarily FLEQ), MGT (Much Greater Than), (NMGT: Necessity MGT), MLT (Much Less Than), (NMLT: Necessarily MLT), FINCL (Fuzzy INCLuded in), INCL, FDIF (Fuzzy DIFrent), (NFDIF: Necessary FDIF) [4, 10]. Like in SQL, fuzzy comparators compare one column with one constant or two columns of the same type.

c) **Fuzzy constants**: Besides the typical constants (NULL), FSQL included many constants such as $[a,b,c,d], #n, $label, [n,m], UNKNOWN, UNDEFINED, etc.

d) *Fuzzy qualifiers:* they are of two natures, absolute and relative [10].

e) *Fuzzy attributes:* the classification adopted for the types of attributes is based on the approaches of representation and treatment of the "imprecise" data [9, 10]. These fuzzy attributes may be classified in four data types. This classification is performed taking into account the type of referential or underlying domain. In all of them the values Unknown, Undefined, and Null are included:

1) *Fuzzy Attributes Type 1*: These are attributes with “precise data”, classic or crisp (traditional, with no imprecision). However, they can have linguistic labels defined over them and we can use them in fuzzy queries. This type of attribute is represented in the same way as precise data, but can be transformed or manipulated using fuzzy conditions. This type is useful for extending traditional databases allowing fuzzy queries to be made about classic data.

2) *Fuzzy Attributes Type 2*: These are attributes that gather “imprecise data over an ordered referential”. These attributes admit both crisp and fuzzy data, in the form of possibility distributions over an underlying ordered dominion (fuzzy sets). It is an extension of the Type 1 that does, now, allow the storage of imprecise information.

3) *Fuzzy Attributes Type 3*: They are attributes over “*data of discreet nonordered dominion with analogy*”. In these attributes some labels are defined ("blond", "red", "brown", etc.) that are scalars with a similarity (or proximity) relationship defined over them, so that this relationship indicates to what extent each pair of labels resemble each other. They also allow possibility distributions (or fuzzy sets) over this dominion, like for example, the value (1/dark, 0.4/brown), which expresses that a certain person is more likely to be dark than brown-haired.

4) *Fuzzy Attributes Type 4*: These attributes are original and they are defined in the same way as Type 3 attributes, without it being necessary for a similarity relationship to exist between the labels.

Ontology introduction

The ontology that describes a Fuzzy Database Schema,[2] as defined previously consists of a fuzzy Database schema and the fuzzy data stored in the Database (the tuples). This ontology, however, represent schemas and data simultaneously as ontology class cannot be instantiable twice, therefore two ontologies is defined, one of which describes fuzzy schemas as instances of a Database catalog ontology and the other which describes the same schema as a domain ontology which will allow the data instatiating it to be defined.

Ontologies vs. Databases

Ontologies allow representing knowledge of any domain in a formal and consensuated way . On the other hand, relational DBs also represent knowledge of any domain but following certain rules specified by ANSI Standard SQL . Nowadays, both technologies coexist together and they can exchange information to take advantage of the information that they represent. Several proposals have been developed for establishing the communication between ontologies and databases. Some of them consist of creating ontologies from a

database structures, others populate ontologies using database data, and there are another which represent databases as ontologies.

Fuzzy Catalog Ontology
Fuzzy Catalog Ontology has been defined to represent the fuzzy RDBMS Catalog [2]. This Ontology contains all the relational concepts defined in the ANSI Standard SQL, for example, the concepts of schema, Table, Column, Constraints, Data Types, etc, and the fuzzy structures extension to manage flexible information: Fuzzy Column, fuzzy labels, fuzzy discrete values and fuzzy data types. [2]
Moreover, the following fuzzy structures have been added to this ontology to complete the fuzzy RDBMS description:-

- *Fuzzy Constraints:* These restrictions, which are described in table can only be applied to fuzzy domains and are used either alone or in combination to generate domains such as, for example, not known, undefined, or null values are allowed, or only labels are allowed.[2]
- *Fuzzy Domains:* These represent a set of values that can be used by one or more attributes. They are defined by a fuzzy data type, one or more fuzzy constraints, and those labels or discrete values that describe this domain.[2]

Fuzzy Schema Ontology
A fuzzy schema ontology [2] is a domain ontology that represents a specific FDB schema. This ontology is generated using the previously defined fuzzy schema as instances of the Fuzzy Catalog Ontology. The generation process for this ontology has been described previously and consists of translating the Table instances into classes, and the attributes instances into properties. The constraints restrictions establish the connections between the attributes properties and the fuzzy data structures. For example, the object property of the Tall attributes allows Trapezoidal, Approximate, Crisp, Interval, Labels values to be defined but not Null, Unknown or Undefined ones. Moreover, the previously defined structures, namely Fuzzy Labels, and Discrete Tags, are included in the Schema Ontology.
The result is a mixed ontology containing the Fuzzy Schema Ontology and the Fuzzy Catalog Ontology instances or a new ontology where the fuzzy values are imported as a separate ontology.
Ontology Database Description
An Fuzzy Catalog ontology work [3] which describes the ANSI SQL 2003 specification is defined by Martinez-Cruz et al. This ontology models fuzzy data specification presented in the GEFRED model, that is a fuzzy relational database catalog. Henceforth, this metaontology, called Fuzzy Catalog Ontology [3], models all fuzzy relational database structures (the system catalog) such as fuzzy data types, fuzzy columns, fuzzy constraints, etc. and their relations with classical SQL elements.
Thus, a database schema, that manages fuzzy or classic data, is viewed as an ontology regardless of any RDBMS implementation constraint. This representation presents a fuzzy database schema from two different perspectives: The first one, a fuzzy schema is presented as a set of instances of the Fuzzy Catalog Ontology [3]. The second one, a fuzzy schema is presented as a domain ontology where a set of classes, properties and constraints defines the reality and data (or tuples) are defined as instances of this ontology. This ontology is generated automatically from the instances of the Fuzzy Catalog Ontology.

Problem Statement
Fuzzy RDBMS requires complex data structures, in most cases, are dependent on the platform in which they are implemented. FRDB systems are poorly portable and scalable, even when implemented in standard RDBs. The solution is to use Ontology which makes system independent of platform used. But all the previous work which uses Ontology are used for web application, none of the work is carried out for window application. Here, a new Ontology is proposed for the above problem.

IV. PROPOSED WORK

A. Objectives

Fuzzy RDBMS requires complex data structures, in most cases, are dependent on the platform in which they are implemented. FRDB systems are poorly portable and scalable, even when implemented in standard RDBs. Our goal is to have a solution that involves representing an FRDB using an Ontology as an interface has been defined to overcome this problem. A new Ontology is proposed, whose definition is extended in this paper, provides a frame where fuzzy data are defined in a platform-independent manner. An implementation layer, which is responsible for parsing and translating user requests into the corresponding DB implementations in transparent, is required to establish communication between the Ontology and the relational databases management system (RDBMS).

Fuzzy Query Ontology
An ontology, called from now Fuzzy Query Ontology (FQO), represents all the basic fuzzy relational database query constituents to get a flexible Select clause definition. After the instantiation of this ontology, the query process can start. This Ontology is specially designed for localhost application. Database queries can be viewed as ontologies from two different perspectives: The first one, a query is a set of descriptions, conditions and rules defined as a set of instances of a Fuzzy Query ontology. In this sense, this query can be viewed as a SQLf [3] statement where any element of the sentence is modeled in the ontology. The second one, a query is described as a a reduced domain ontology where a set of classes, properties and axioms represent a query specification and the ontology instances represent the resulting tuples. A diagram of this ontology is shown in figure and it is described below:

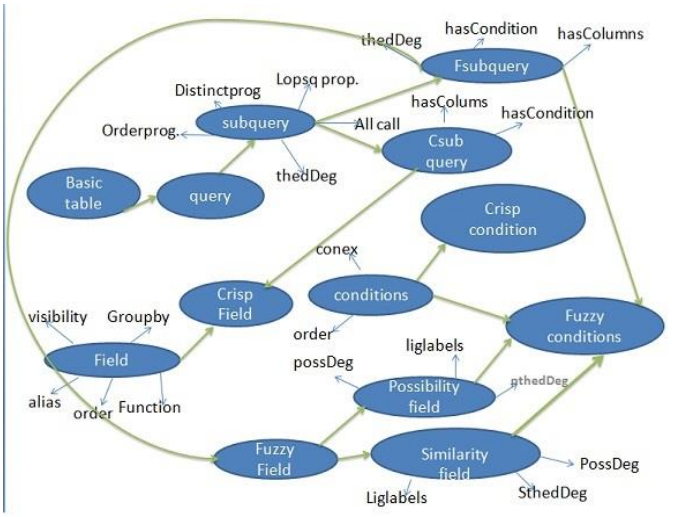
• *Query class:* This class represents a query composed of one or more subqueries joined by logical connectors.

• *Subquery class*: It is a simple query composed of a list of ordered columns. This class has two subclasses:
classic and fuzzy subqueries. Subquery properties are:
– distinct prop: establishes if the result set allows duplicated registers.
– order prop: establishes the order of a subquery in the general query structure.
– LOpsq prop: establishes the logical operator to integrate this subquery in the main query structure.
The most common operators are: Union and Intersection (but they can be extended).
– thedDeg: establishes the threshold to accomplish the logic operation.
– allColl: establishes if all the columns are displayed in the answer.
• *Crispquery class:* It is a subquery that only uses classical data. A classic subquery involves only classic columns.
This representation can be made as a SQL sentence as well.
– hasColumns: establishes which columns are involved in a subquery.
– hasCondition: establishes the filtering conditions to get results.
– having: establishes the filtering conditions to get results in an aggregation operation.
• *FuzzyQuery class:* Subquery that uses fuzzy columns. This kind of queries can also use classical data. This class is represented by the properties:
– thedDeg prop: represents a threshold that the resulting records of the fuzzy query must accomplish.
– hasColumns prop: establishes the columns involved in a subquery. The range of this property is Field class.
– hasCondition prop: establishes the filtering condition to get results. This property goes to conditions class because conditions involved in this subquery can be fuzzy or not.
– having prop( ): establishes the filtering condition to get result in an aggregation operation.
• *Field class:* represents any column involved in an specific subquery, no matter if they are visible or not.
These columns are related with columns defined in the Schema Ontology Description modeled. It has two subclasses: fuzzyField and crispField. These classes have the following properties:
– visibility: determines if this column will be in the resulting set.
– alias: establishes the final name of the column.
– order: establishes the order of the column in the result set.
– function( ): establishes the operation to do in this column. This can be a simple operation or an aggregation function (AVG, MAX, MIN, COUNT, Other).
– groupBy( ): establishes if this column is used to make a group.
– name: establishes the relation between this column and the one defined in the schema definition.
• *FuzzyField class:* represents any column involved in an fuzzy subquery, no matter if they are visible or not.
It has two subclasses: PossibiliticField and SimilarityField class
• *PossibiliticField class:* this class deals with type 2 of fuzzy attributes[1] which is discussed above
  - possDeg: gives a degree of possibilitic distribution approach
  - liglabels: deals with linguistic labels with its degree
  - pthedDeg: represents a threshold that the resulting records of the fuzzy query must accomplish.
• *SimiliarityField class:* this class deals with fuzzy attributes type 3 [1] which is already discussed.
  - possDeg: gives a degree of similarity distribution approach
  - liglabels: deals with linguistic labels with its degree
  - sthedDeg: represents a threshold that the resulting records of the fuzzy query must accomplish.
• *Condition class:* This class represents any subquery condition to filter data. All conditions are related with a column or a set of them, one operator, and a condition object. This class has two subclasses depending if there are fuzzy: fuzzyConditions and Classic conditions.
Condition class properties are:
– order: establishes the order in the list of conditionsin a subquery.
– conex: establishes the logical operator that join this condition in a subquery: AND, OR.
– not: establishes if the condition is negated or not.
• *ClassicCondition class:* This class represents any classic condition, and the property origin defines the list of columns to be compared with.
• *FuzzyConditions class:* this class represents any fuzzy conditions, and the property origins that defines the list of columns to be compared with.

V. DESIGN & IMPLEMENTATION OF FUZZY QUERY ONTOLOGY

In this implementation, an ontology is used which is proposed in above chapter, called Fuzzy Query Ontology. An implementation layer, which is responsible for parsing and translating user requests into the corresponding DB implementations in transparent, is required to establish communication between the Ontology and the relational databases management system (RDBMS). This implementation is designed with respect to corporate.

Parsing Technique Used:
SableCC [7] is a parser generator which generates fully featured object-oriented frameworks for building compilers, interpreters and other text parsers. In particular, generated frameworks include intuitive strictly-typed abstract syntax trees and tree walkers. SableCC [7] also keeps a clean separation between machine-generated code and user-written code which leads to a shorter development cycle.
SableCC,[7] an object-oriented framework that generates compilers (and interpreters) in the Java programming language. This framework is based on two fundamental design decisions. Firstly, the framework uses object-oriented techniques to automatically build a strictly-typed abstract syntax tree that matches the grammar of the compiled



language and simplifies debugging. Secondly, the framework generates tree-walker classes using an extended version of the visitor design pattern which enables the implementation of actions on the nodes of the abstract syntax tree using inheritance. These two design decisions lead to a tool that supports a shorter development cycle for constructing compilers.

Fuzzy Query Translator

The queries are written in SQLf which is an extention to SQL. Fuzzy Query Executer works as a translator, that translates fuzzy queries to standard, SQL queries, and executes them with RDBMS. Fuzzy Query Executer Using Ontology is completely written in *Java* (version 1.6), a programming language.

SQL Client directly interact with Relational Database Management System through SQL Executer and do not need any translation in between. SQLf is language used in the project to retrieve data which is fuzzy in nature. SQLf Client need translation because Database can understand only SQL Language. So, Fuzzy Query Translator is used to convert SQLf query into SQL query. Ontology which is used in this project is independent of the platform used. Figure shows ontology is outside of RDBMS. Fuzzy Query Ontology is implemented in this project through various classes and its properties and behavior. To interact with fuzzy database as discussed in chapter 2, a FMB is needed. In relational database, FMB is created as shown in Block Diagram figure 4.1. FMB contains few tables in database which stores the fuzzy attributes with their parameters to calculate fuzzy degree which handles the fuzzy database. This Block Diagram shows how data is carried in database. An ontology for fuzzy information representation is proposed in this paper. The ontology includes knowledge about how to manage and represent uncertain and imprecise data. Figure 4.1 shows how the system catalog is related to the ontology modeling it. The Ontology Client module carries out the same operations through the Ontology than the DBMS Clients. The connection between the ontology and the database needs an interface, the Ontology Interface, which establishes the communication and refreshes the data. The fuzzy model representation comprises two well-differentiated parts. Firstly, the ontology must define the necessary classes and slots to represent the metadata. Secondly, the ontology will be able to represent classical or fuzzy information as instances of the relations. In this ontology, metadata establish how the fuzzy information will be stored.

4.4 Verification of Implemented Fuzzy Query Ontology

This project is implemented in context of some workers who are working for a various organization like Academics, government, Industry. The job expertise for organizations are AI, Statistics, Robotics, Expert System. There are various fuzzy attributes like age, salary, expertise and there are various linguistic labels for fuzzy attributes like *age* there are BABY, YOUNG, MIDDLE, MATURE, OLD and for *salary* labels are SMALL, MIDDLE, HIGH.

Examples:

1. select name of worker, location of worker and age of worker from table name workers where fuzzy attribute age is ‘young’;

```
Output - Fuzzy_Query_Executor (run)
run:
PostgreSQL
Welcome to the Fuzzy Query Executor developed by Kumar Kaushik for

SQL_FQE>select fname,lname, age from workers where ~age is 'young';
+---------+------------+-----+--------------+
| fname   | lname      | age | fuzzy_degree |
+---------+------------+-----+--------------+
| neha    | ghaziabad  |  16 |  0.933333333 |
| mandeep | meerut     |  23 |  0.466666667 |
| rajesh  | muradnagar |  27 |  0.200000000 |
+---------+------------+-----+--------------+
3 rows in set
OK
SQL_FQE>
```

Figure 3 shows the result of example 1

2. select name of worker, location of worker and salary of worker from table name workers where fuzzy attribute is salary is ‘high’;

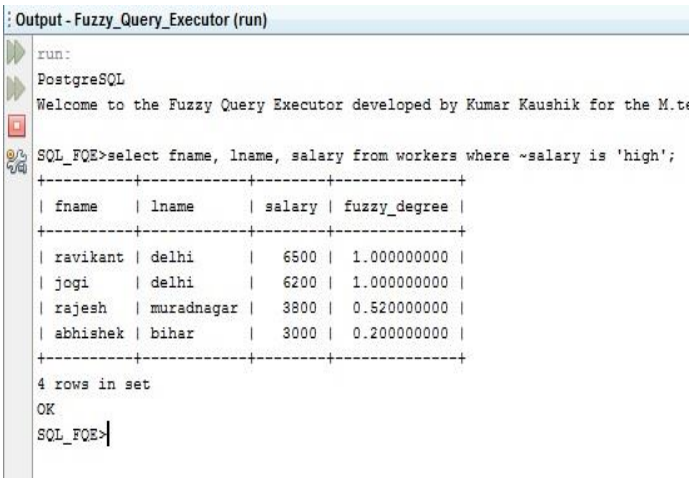


Figure 4 shows the result of example 2

VI. CONCLUSION & FUTURE WORK

6.1 Conclusion

The above defined Ontology is specific to data retrieval from database. It is specifically designed for a flexible Select clause which represents all the basic fuzzy relational database query constituents. In this proposal of an ontology which represents a query structure regardless of any RDBMS implementation is defined. This ontology allows generating and executing any query on fuzzy or classical data according to the system where it is executed. The use of ontologies has provided of an independent layer where data can be modeled regardless of any RDBMS implementation particularity. Thus, the client interaction is performed through this ontology and an interface between the ontology and a real RDBMS is in charge of establishing the communication. A new Ontology is defined in this dissertation called Fuzzy Query Ontology.

A Fuzzy Query Ontology design, developed, implemented and verified successfully on given system environment (hardware & software) for data manipulation(data retrieval (select clause)).

6.2 Future Work

An Fuzzy Query Ontology can be design for data manipulations (insert, delete and update clause) and further can be extended for database definition (create and delete clause).

REFERENCES

[1] J. Galindo, A. Urrutia, and M. Piattini, “Fuzzy Database Modeling, Design and Implementation”. Idea Group Publishing, 2006

[2] “Describing Fuzzy Database DB Schemas as Ontologies: A System Architecture View”. Springer 2010. 13 th International Conference, IPMU july 2010.

[3] Carmen Martinez-Cruz.. “An Ontology to represent Queries in Fuzzy Relational Databases”. IEEE 2011

[4] C. Martinez-Cruz, I. J. Blanco, and M. A. Vila, “The use of ontologies for representing database schemas of fuzzy information,” International Journal of Intelligent Systems, vol. 23, no. 4, pp. 419–445, February 2008.

[5] Natalya F. Noy and Deborah L. McGuinness. Ontology development 101: A guide to creating your ¯rst ontology. Technical report, http://protege.stanford.edu/publications/ontologydevelopment/ontology101.html.

[6] Extended_FSQL_Server: a Server for the description and themanipulation of FRDB P. Bosc and O. Pivert, "SQLf: A Relational Database Language for Fuzzy Querying", IEEE Transactions on Fuzzy Systems, Vol 3, No. 1, Feb 1995.

[7] www.sablecc.com