A Fuzzy Querying System based on SQLf2 and SQLf3

Juan Eduardo
Universidad Simón Bolívar, Departamento de Computación,
Apartado 89000, Caracas 1080-A, Venezuela
jceduardo@hotmail.com

and

Marlene Goncalves
Universidad Simón Bolívar, Departamento de Computación,
Apartado 89000, Caracas 1080-A, Venezuela
mgoncalves@ldc.usb.ve

and

Leonid Tineo
Universidad Simón Bolívar, Departamento de Computación,
Apartado 89000, Caracas 1080-A, Venezuela
leonid@ldc.usb.ve

Abstract

For improving the expressive power, there have been proposed and developed several extensions of SQL. One of them is SQLf, a fuzzy set based extension that allows the expression of flexible queries involving user preferences. On the other hand, SQL has evolved into: SQL2 that incorporates relational algebra operations constrains and sub-queries; and SQL3 that incorporates features of deductive, active and object oriented databases. In a previous work we have defined SQLf2 and SQLf3 as extensions of SQLf with the fuzzy set based treatment of new features from SQL2 and SQL3. In this paper we present a real fuzzy querying system based on SQLf2-SQLf3 that we have built on top of a RDBMS. This system provides a web based interface and an API.

Keywords: Queries, fuzzy queries, database, web interface, SQL.

1. INTRODUCTION

The use of boolean logic restricts the treatment of imprecision or uncertainty in database systems. The queries don't get prospective results, because these queries are restrictive and they don' t contemplate important elements that could be considered to analyze data. A solution to this problem is using fuzzy sets in databases systems in order to express non crisp data and gradual (or flexible) requirements. We have concentrated our efforts in the study and implementation of fuzzy querying languages on relational databases due to the fact that many organizations store their information in relational databases.

There are some different proposals of fuzzy set based flexible querying languages, SQLf [[3]] is one of them that has been proposed by the research team of IRISA-ENSSAT, whose under the leadership of Patrick Bosc. SQLf is a fuzzy extension of the standard SQL that allows using fuzzy conditions in any place where SQL allows a regular one. We have adopted this proposal since 1998 [[26]] due to its desired characteristics. We have proposed some extensions to SQLf [[25]] and we have work in a real implementation of a flexible querying system based in SQLf and its extensions. We call this query system SQLfi (SQLf on Internet).

The standard SQL has evolved due to the new technology present in database systems, ISO and ANSI organizations have carried out the work of updating the standards definition of SQL. The two later versions of SQL are known as SQL2 [[1]] and SQL3 [[20]]. The norm SQL2 incorporates some relational algebra operations, some mechanisms for constrains specifications and the possibility of subqueries use as the base relation for a query. The norm SQL3 incorporates some features of deductive databases, active databases and object oriented databases.

In previous works we have studied the new features of SQL2 and SQL3 determining which of theme are susceptible of a fuzzy treatment. These works leaded us to the definition of two extensions of SQLf, named: SQLf2 [[10]] and SQLf3 [[11]], corresponding to the definition of fuzzy querying components based in the norms SQL2 and SQL3, respectively. We think that it is not sufficient to contribute to the theme of fuzzy querying proposing the theoretical definition of such languages, but also making a real implementation of them in order to allow further studies of behavior and development of applications.

We have structured the present paper as follows: Section 2 briefly presents some other' s related works, in section 3 we present the fuzzy query languages SQLf, SQLf2 and SQLf3. On the other hand, section 4 tells how our system prototype has evolved throw the intervention of some collaborators. Section 5 is devoted to outline the new SQLf2 and SQLf3 features that are implemented in the current system prototype, we omit their features proper of SQLf that were implemented in previous versions of the prototype. The architecture of our fuzzy querying system is shown in section 6. Some remarks about the used implementation tools are in section 7. Finally, we summarize and point out to the future in the section 8.

2. RELATED WORKS

Several efforts have been made by different authors in order to provide flexible querying database systems; some of them are the following:

OMRON [[21]] is a processor that contains a variant of SQL with fuzzy logic and is a fuzzy information retrieval library based. It' s a fuzzy query interface on traditional databases[[19]]. Don' t allow fuzzy quantifiers use.

FQUERY is an effort that adds fuzzy query functionality on a small database management system [[14]] (Microsoft ACCESS for Windows). Allow fuzzy quantifiers use in order to qualify criteria quantity that satisfies a query, but this is not allowed in partitioned or nesting queries.

BANDINS is a project that allows data flexible representation and manipulation. Its query language is SQLf [[3]]. There are some query processing mechanism [[4]][[5]][[6]][[7]] and the most relevant is derivation principle [[6]][[7]] because it has a less cost.

Fukami-Umano in [[27]] proposed a fuzzy relational model. Which query language is the Fuzzy Relational Algebra. They propose the building of a new database engine with fuzzy querying capabilities.

A fuzzy query language was presented by Wong-Leung in [[28]] to retrieve information by means of translation from a fuzzy query to a query for the database management system VAX Rdb/VMS and multicriteria decision making.

ISKREOT (Intelligent System for Knowledge Representation using Expert system and Object Technology) [[18]] is presented in and it is front-end intelligent information interface operating through a relational database ORACLE with fuzzy queries.

None of these previous works have incorporated in the proposed and developed querying systems features of SQL2 and SQL3 with a fuzzy set based treatment

3. QUERY LANGUAGES

One of the more remarkable efforts in the creation of Flexible Querying System for Relational Databases is SQLf [[3]]. It is the most complete fuzzy extension [[26]] to SQL due to the diversity of fuzzy queries that allows the extension of all SQL constructions with Fuzzy Logic.

Fuzzy queries involve fuzzy terms (atomic predicates, modifiers, connectors, comparators and quantifiers) whose meaning is user and context depending. Therefore a SQLf based querying system must provide a language for defining such terms.

In SQLf, the querying basic block is:

SELECT <attributes> FROM <relations>

WHERE <fuzzy condition> WITH CALIBRATION [n|λ|n,λ]

Semantic of this construction is cartesian product of relations involved in clause FROM, selecting those tuples that satisfied fuzzy condition and taking the fuzzy projection of those attributes indicated in clause SELECT. In clause WHERE, some logical expression can be used formed with terms defined by the user and predefined operators of fuzzy logic.

The result of a query in SQLf is a fuzzy set. In clause WITH CALIBRATION, a tolerance is specified to select tuples, which is called calibration. In the original definition of Bosc and Pivert [[3]], the calibration is specified in the clause SELECT, in a later work, Goncalves and Tineo [[10]][[11]] extended the definition with this clause in proof the orthogonality of language. Two calibration types exist: Quantitative and Qualitative. In case of quantitative calibration, a maximum number "n" of answers is obtained and in case of qualitative calibration those tuplas whose membership value is bigger or similar to the minimum level of tolerance "λ" is obtained.

SQLf also allows subqueries with different nesting operators. Subqueries are query structures like the basic block without the WITH CALIBRATION clause. Into the subqueries, attributes from the relations in the outer query may be used both in the SELECT and the WHERE clause. The general structure of such queries is:

SELECT <attributes> FROM <relations>

WHERE <nesting operator> <fuzzy subquery> WITH CALIBRATION [n|λ|n,λ]

SQLf also has a partitioned block querying structure. In this kind of query it is possible to impose a selection fuzzy criterion over the groups of the partition. Such queries are:

SELECT <attributes> FROM <relations> GROUP BY <attributes>

HAVING <fuzzy condition> WITH CALIBRATION [n|λ|n,λ]

Inspired in the norm SQL2, we have defined SQLf2 [[10]], an extension of SQLf that contemplates incorporation of fuzzy set based extensions of: relational algebra operators, integrity constraints, views definitions, support of date and time types, subqueries in from clause and data manipulation operations and new conditions kinds.

On the other hand, we have created SQLf3 [[10]] as an extension of SQLf inspired in the norm SQL3. SQLf3 includes elements of deductive databases, active databases and object oriented databases. They all with a fuzzy set based treatment.

For simplicity and space restrictions, we don' t present here the features of SQLf2 and SQLf3 in detail. The reader is referred to the previous works [[10]][[11]]. We will just enumerate the SQLf2 and SQLf3 features that where implemented in the built interpreter, see section 5

4. PROTOTYPE EVOLUTION

Since 1998, Tineo [[25]][[26]] has been directing the creation of a flexible querying system to relational databases based in SQLf. This system has been developed on top of the ORACLE Relational Database Management System (RDBMS). The development has been made in the database laboratory at Simón Bolívar University and Venezuela' s Central University.

As all software, SQLf has experienced some iteration in its life cycle in order to increasing its functionality. Some times, the prototype was limited to existing technology and enhanced in a new version.

In 1999, Borrajo and Rengifo [[2]] carried out a first prototype of an interpreter with a minimum subset of SQLf. This prototype was limited to the type of queries expressed with a simple block, and it was developed with certain built in linguistic terms stored in a database.

In 2000, Gutiérrez [[12]] has performed the implementation of evaluation mechanisms for SQLf. He has dealt with the three kinds of queries of SQLf: Basic Block, Nesting and Partitioning. This implementation was restricted to specific queries programmed in the code, based in a real database.

Also in 2000 Ramírez [[24]] extended the prototype integrating the mechanisms, allowing interactive definition of fuzzy terms. So this new prototype satisfied the complete SQLf. With it, the user may use his proper database and define his proper linguistic terms.

In 2001 Goncalves [[9]], extended the existing prototype of SQLf with some new characteristics defined for SQLf2 and SQLf3. Nevertheless, Goncalves' work has been focused mainly in the theoretic definition of new extensions rather than the completion of the prototype.

In 2002 Hernández, Montaña [[13]] and León, Martínez [[15]] have worked in the extension of the existing prototype with the new versions of the querying language: SQLf2 and SQLf3. They have implemented some of characteristics that had not been carried out in the previous version.

Also in 2002 Rodríguez and León [[16]] developed an improved new prototype of flexible querying system with all SQLf characteristics. This new prototype is based on Internet technology, so it's called SQLfi.

In this paper we present the ultimate SQLfi prototype, which includes the characteristics defined for SQLf2 and SQLf3. This implementation provides an application program interface in order to be used in some languages as ASP, JSP, JAVA, among others [[8]].

5. NEW FEATURES

Current system prototype implements most the new features of SQLf2 and SQLf3 functionality. The used developing tools limited the selection of the implemented features. We have chosen those features that were compatible with the support offered by the used RDBMS, Oracle 8i.

5.1 SQLf2 Features

The implemented SQLf2 characteristics are:

1. Fuzzy comparison over date type: Support of fuzzy comparison operators for Date type.
   CREATE COMPARATOR <symbol>
   ON {DATE | TIMESTAMP} AS <expression>
2. Checks with λ calibrated fuzzy conditions: It's possible to having sub-queries with calibration in the clause CHECK of CREATE TABLE sentence.
   CHECK (<fuzzy- condition>)
   WITH CALIBRATION λ
3. Views based in fuzzy queries: Support of creating and modifications on views defined by selection with calibration of some attributes of a relationship.
   CREATE VIEW <name> AS <subquery>
   WITH CALIBRATION λ
4. Constraints that involves multiple tables: The constraints are defined by means of CHECK clause. It is sometimes required that a constraint involves a complete relationship or more than a relationship.
   CREATE ASSERTION <name> CHECK < fuzzy- condition >
   WITH CALIBRATION λ
5. Unique operator over fuzzy conditions in partitioning: Unique operator indicates if duplicates exist in a fuzzy sub-query.
   SELECT <attributes> FROM <relations> GROUP BY <attributes>
   HAVING UNIQUE <fuzzy condition>
   WITH CALIBRATION λ
6. Fuzzy selection control structure: Support of case constructor with fuzzy comparisons.
   SELECT CASE (<attributes>)
   WITH CALIBRATION λ
   {WHEN <fuzzy predicate> THEN action}
   FROM <relations> WHERE <fuzzy condition>
7. Join over the result of fuzzy queries: Support of cross and natural join with calibration and fuzzy condition.
   (<subquery> {CROSS | NATURAL} JOIN <subquery>)
   WITH CALIBRATION λ
8. Theta joins operators over the result of fuzzy queries: Support of outer join with calibration and fuzzy condition
   (<subquery> [[LEFT | RIGHT | FULL] OUTER] JOIN <subquery>
   ON <fuzzy condition>)
   WITH CALIBRATION λ
9. Queries over the table resulting from a fuzzy sub-query: Support of a fuzzy sub-query o join in FROM clause.
   SELECT <attributes> FROM <subquery>
   WHERE <fuzzy condition>
   WITH CALIBRATION λ
10. Set operators over the result of fuzzy queries: Support of set operators for fuzzy query.
    (<sub-query> {UNION | INTERSECT | EXCEPT} <subquery>)
    WITH CALIBRATION λ
11. UPDATE operation with fuzzy condition: Support of UPDATE operators with fuzzy sub-queries on the same table.
    UPDATE <table> SET <attrib>=<value> [{, <attrib>=< value>}]
    WHERE <fuzzy condition>
    WITH CALIBRATION λ
12. DELETE operation with fuzzy condition: Support of DELETE operators with fuzzy sub-queries on the same table
    DELETE FROM <table> WHERE <fuzzy condition>
    WITH CALIBRATION λ

5.2 SQLf3 Features

The implemented SQLf3 characteristics are:

1. Fuzzy predicates on complex structures: Definition of fuzzy predicates on complex structures and datatypes defined by the user.
   CREATE FUZZY PREDICATE <name> ON <tuple type> AS <fuzzy condition>
2. Triggers with fuzzy conditions: Support of fuzzy trigger with components of traditional trigger (an action and a condition) viewed as fuzzy extension.
   CREATE TRIGGER <name> … WHEN (<fuzzy condition>) WITH CALIBRATION λ {<action>}
3. Fuzzy conditions in control structure IF: Fuzzy extension of IF sentence of the language procedural.
   IF <fuzzy condition> THEN {<action>}
   [{ELSEIF <fuzzy condition> THEN {< action >}}]
   [ELSE {< action >}]
   WITH CALIBRATION λ
   END IF
4. Fuzzy conditions in control structure FOR: Fuzzy extension of FOR sentence of the language procedural.
   FOR <result> AS <fuzzy query> DO <action> END FOR
5. Fuzzy conditions in control structure WHILE: Fuzzy extension of WHILE sentence of the language procedural.
   WHILE <fuzzy condition> DO {<action>} WITH CALIBRATION λ END WHILE
6. Fuzzy conditions in control structure REPEAT: Fuzzy extension of REPEAT sentence of the language procedural.
   REPEAT {<action>} UNTIL <fuzzy condition> WITH CALIBRATION λ END REPEAT
7. Fuzzy quantifiers FOR SOME and FOR ALL: To extend FOR ALL and FOR SOME so that they operate on fuzzy sub-queries.
   SELECT <attributes> FROM <relations>
   WHERE [FOR SOME | FOR ALL] <attribute> <fuzzy subquery>
   WITH CALIBRATION λ

6. SYSTEM ARCHITECTURE

For building the fuzzy querying system, we have used a three layers' architecture (Fig. 1). The lowest, data layer, is the RDBMS. The middle, logical layer, is an interpreter of SQLf2-SQLf3. The upper layer is the client's interface.

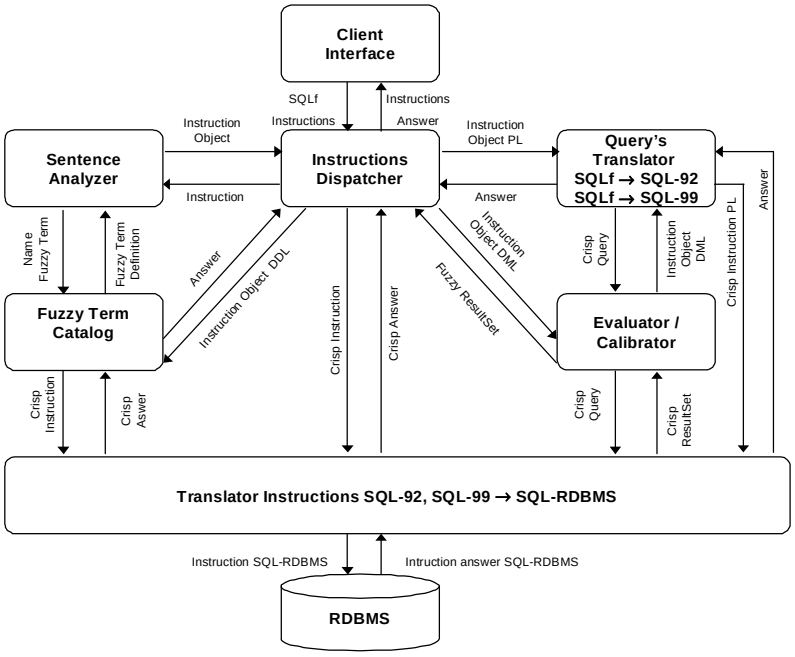

Fig. 1. Fuzzy Querying System Architecture

The main components inside the querying system are:

Client's Interface. Receives user' s fuzzy queries and term definitions. Shows the final results of user operations: the fuzzy query answer.

Instructions Dispatcher: It is the responsible for delivering at remainder of the modules the necessary structures for the execution of the instruction and to receive of these the respective answers.
Fuzzy Terms Catalog. Allows the specification of user defined fuzzy terms. Retrieves the definition of such terms in order to be used by the sentence analysis and evaluation mechanisms. These terms are stored into a database.
Sentence Analyzer. Analyzes the sentences introduced by the user, checking the syntactic and semantic correctness of the statements. Perform the translations needed for the execution of the statements. In case of fuzzy queries, builds a tree structure of the fuzzy query that is used in the evaluation process.
Evaluator/Calibrator. Performs the evaluation of fuzzy queries. Interacts with the RDBMS in order to retrieve database element relevant to the query processing. For so doing, uses a regular SQL query that is given by the Sentence Analyzer. Computes the satisfaction degrees and calibrates the answer of the fuzzy query. For so doing uses the result of the regular query and the tree structure of the fuzzy query.
Query's Translator SQLf → SQLf92 and SQL99. It is the responsible for translating queries that contain fuzzy terms in regular queries using the derivation principles, producing queries in SQL-92 and SQL-99 standards.
Translation Instructions SQL-92 and SQL-99 → SQL-RDBMS. We have adopted to express regular queries in the standards SQL-92 and SQL-99 in order to make our system as portable as possible. Therefore, we need to translate regular queries into specific RDBMS query language.

7. IMPLEMENTATION

This querying system has been made using:
The Java programming language [[22]]. Java is a language of programming object oriented for a distributed application on WEB platform. Besides it offers the facility of libraries and/or interfaces for connection with databases or server pages using JDBC convention.
The ORACLE 8.0.5.0.0 [[15]] [[17]]. RDBMS. Oracle is one of the most efficient in the market for storage capacity, answer capacity, stability, backup mechanisms, security, etc. The tool of Oracle used was OCI (Oracle Call Interface) that is an interface for programming of applications that allows to the applications written in C interact with an Oracle server.
The CLAIRE [[23]] is a 100% Java compatible LR (1) parser and lexical analyzer generator, designed and developed by programming language research group at Simón Bolívar University. It's also language independent that is it can generate code on any language it has a plug in for. The resulting parser is a Java program module.

8. CONCLUDING REMARKS

In this paper is shown the feasibility of implementation of a fuzzy querying system on web that embraces most of the characteristics of SQLf2 and SQLf3, enriching the functionality of the existent prototype of SQLf with a bigger quantity of requirements that the user can execute. Also, it was appealed to a series of tests whose results guaranteed that this prototype adapts to the requirements and necessities outlined for the support of the new characteristics of SQLf2 and SQLf3.
The built fuzzy querying system supports: fuzzy comparison on dates. fuzzy checks. Partitioned queries using the UNIQUE operator over fuzzy conditions, fuzzy views, fuzzy assertions, fuzzy joins, fuzzy subqueries in the from clause, fuzzy set operations, update and delete with fuzzy conditions, fuzzy predicates on complex structures, control instructions with fuzzy conditions, triggers with fuzzy conditions in the when clause, stored functions, procedures and ADT's with fuzzy elements.
We are working now in the implementation of the remaining SQLf2 and SQLf3 features. We also points to performance studies of system prototype and definition of different query evaluation mechanisms for fuzzy queries in SQLf2 and SQLf3.
We know a fuzzy query can return a bigger quantity of results that a classic query. The most remarkable fuzzy queries process mechanisms are based in *λ-cut*, by the application of a principle called Derivation Principle, for tkin advantages of the relationship between fuzzy queries and regular ones. This principle has been defined and studied by Bosc and Pivert [7] and Tineo [7]. We work in this principle to SQLf2 and SQLf3 and its implementation as query evaluation mechanism in our prototype.

References

[1] ANSI X3. Database Language SQL, 135-1992, American National Standards Institute, New York.

[2] Borrajo, F., Rengifo, G., Implementación del Lenguaje de Interrogaciones Flexibles a Base de Datos Relacionales SQLf, Informe final de Proyecto de Grado, Universidad Simón Bolívar., Julio 1999.

[3] Bosc P. and Pivert O. SQLf: A Relational Database Language for Fuzzy Querying, IEEE Transactions on Fuzzy Systems, Vol 3, No. 1, Feb 1995.

[4] Bosc P. and Brisson A. On the evaluation of some SQLf nested queries, Proceeding International Workshop on Fuzzy Databases and Information Retrieval, 1995.

[5] Bosc P., Pivert O. and Farquhar K. Integrating Fuzzy Queries into an Existing Database Management System: An Example, International Journal of Intelligent Systems, V.9, pp 475-492,1994

[6] Bosc P. and Pivert O. On the efficiency of the alpha-cut distribution method to evaluate simple fuzzy relational queries, Advances in Fuzzy Systems-Applications and Theory, Vol 4, Fuzzy Logic and Soft Computing, B. Bouchon-Meunier, R.R.Yager, L.A. Zadeh eds, Wold Scientific, pp 251-260, 1995.

[7] Bosc, P. & Pivert, O., SQLf Query Functionality on Top of a Regular Relational Database Management System, Knowledge Management in Fuzzy Databases, Pons, O., Vila, M. and J. Kacprzyk (Eds.), Physica-Verlag, (2000), Pp. 171-190.

[8] Eduardo J., Sistema de Interrogación Flexible a Bases de Datos SQLf2 – SQLf3. Informe Final de Trabajo Especial de Grado presentado ante la Universidad Simón Bolívar, Diciembre 2003.

[9] Goncalves M., Extensión del Lenguaje de Interrogación Flexible a Bases de Datos SQLf mediante las normas SQL2 y SQL3. Informe Final de Trabajo Especial de Grado presentado ante la Universidad Simón Bolívar, Septiembre 2001.

[10] Goncalves M. and Tineo, L. SQLf Flexible Querying Language Extension by means of the norm SQL2, The 10th IEEE International Conference on Fuzzy Systems, Vol 1, Dec 2001.

[11] Goncalves, M. and Tineo, L. SQLf3: An extension of SQLf with SQL3 features, The 10th IEEE International Conference on Fuzzy Systems, Vol 3, Dec 2001.

[12] Gutiérrez, L. Desempeño de Mecanismos de Evaluación de SQLf, Informe final de Proyecto de Grado, Universidad Simón Bolívar, Septiembre 2000.

[13] Hernández, G., Montaña A. HECDOCf: Una Herramienta para la Evaluación de Cursos y Docentes mediante SQLf2 a través del Web, Informe final de Proyecto de Grado presentado ante U.C.V., Junio 2002.

[14] Kacpryzyk J. and Zadrozny S., Fuzzy Queries in Microsoft AccessTM v.2, Proceedings of Fuzzy IEEE'95 Workshop on Fuzzy Database Systems and Information Retrieval, pp 61-66, 1995.

[15] León, G., Martínez, D. SISECDf3: Sistema de apoyo basado en la tecnología Internet para la Evaluación de Cursos y Docentes mediante SQLf3, Informe final de Proyecto de Grado presentado ante U.C.V., Junio 2002.

[16] León W., Rodríguez H., "Sistema de Interrogación Flexibles en Internet a Bases de Datos Relacionales SQLfi" Informe final de proyecto de grado presentado ante la U.S.B., Octubre 2002.

[17] Loney K. and Koch G. Oracle8i The Complete Reference. ISBN: 0072123648. McGraw-Hill. 2000.

[18] Loo G. and Lee K. An Interface to Databases for Flexible Query Answering: A Fuzzy-Set Approach. Lecture Notes in Computer Science 1873. DEXA 2000. Septiembre 2000. Londres, Reino Unido. pp 654-663.

[19] Mansfield W. & Fleischman R. A High-performance, Ad-hoc, Fuzzy Query Processing System Journal of Intelligent Systems, Vol. 2, pp. 397-420, 1993.

[20] Melton J. ISO/ANSI Working Draft: Database Language SQL (SQL3), X3H2-93-091/ISO DBL YOK-003.

[21] Nakajima H., Sogoh T., Arao M. Fuzzy Database Language and Library-Fuzzy Extension to SQL, Proceedings of Second IEEE International Conference on Fuzzy Systems, pp 477-482, 1983.

[22] Naughton P., Schildt H. Java Manual de Referencia, McGraw-Hill.

[23] Pacheco P., "Implementación del Analizador Lexicográfico, Sintáctico y Semántico Claire", Informe final de proyecto de grado presentado ante la U.S.B., Enero 2000.

[24] Ramírez, J. Interpretador del Lenguaje de Interrogaciones Flexibles a Bases de Datos Relacionales SQLf, Informe final de Proyecto de Grado, Universidad Simón Bolívar, Enero 2001.

[25] Tineo L. Algunos Aportes en Bases de Datos Difusas. Trabajo de Ascenso presentado ante la Universidad Simón Bolívar, Sartenejas Septiembre 2001.

[26] Tineo L. Interrogaciones Flexibles en Bases de Datos Relacionales. Trabajo de Ascenso presentado ante la Universidad Simón Bolívar, Sartenejas Enero 1998.

[27] Umano M., Fukami S. Fuzzy Relational Algebra for Possibility-Distribution-Fuzzy-Relational Model of Fuzzy Data, Journal of Intelligent Information System, Vol 3, pp 7-27, 1994.

[28] Wong M. and Leung K. A fuzzy Database-Query Language. Information Systems. Vol 15. No. 5, pp 583-590, 1990.