

Towards a XML Fuzzy Structured Query Language

Carlos D. Barranco
carlosd@decsai.ugr.es

Jesús R. Campaña
jesuscg@decsai.ugr.es

Juan M. Medina
medina@decsai.ugr.es

Dept. of Computer Science and Artificial Intelligence - University of Granada
Daniel Saucedo Aranda, s/n 18071 Granada (Spain)

Abstract

The paper introduces XFSQL (XML Fuzzy Structured Query Language), a preliminary prototype of a portable query language for Fuzzy Database Management Systems (FDBMS) using the XML formatting rules. The need for a portable query language for FDBMSs is discussed, and a basic language syntax and its associated datamodel are proposed. Additionally, the paper proposes a framework for implementing XFSQL query translator interfaces, which would make a particular FDBMS able to process XFSQL queries by translating them to the FDBMS proprietary fuzzy query language. Finally, an example of XFSQL query, together with its translation to the query language of a FDBMS implemented on a commercial DBMS, is shown.

Keywords: Fuzzy Databases, XML, Query Language, Object-Relational.

1 Introduction

Together with the inclusion of the fuzzy paradigm in many aspects of computer science, an easy and efficient way of managing, manipulating and storing fuzzy data is becoming more necessary. This fact makes fuzzy database management systems (FDBMS) an important part of complex soft-computing systems, and stimulates research in this area. This leads to different proposals of fuzzy database models and implementations [1, 2, 3, 4, 5, 6, 7].

The growing computational and storage needs of soft-computing systems demand high quality and

performance for components of these systems, especially for FDBMSs. Therefore, in order to create a FDBMS satisfying the performance requirements with low implementation effort, a strategy to implement a FDBMS extending a commercial DBMS is becoming more significant [1, 2, 7].

Extending commercial DBMS to implement a FDBMS takes advantage of the high performance, value added extensions and services – datamining, spatial data, clustering, etc. – of host DBMS. On the other hand, commercial DBMS implement proprietary extension mechanisms – stored procedures, user datatypes, etc. – which result in different and proprietary ways of accessing these extensions – procedure calls, object methods, etc. –. Due to these differences implementing and accessing extensions, it is almost impossible to specify a fuzzy query language accepted directly by the host DBMS and working in every compatible fuzzy extension of a commercial DBMS, like SQL:2003 tries to do in crisp DBMS area.

A common and portable fuzzy query language could represent an improvement in FDBMS area, analogous to SQL standards. It would allow query portability between different FDBMS and platforms, increasing interoperability of the systems. This query language should be specified in a format which ensures high portability between different platforms, as the XML document format does. Additionally, in order to use a portable common fuzzy query language to query FDBMS, a middle layer between FDBMS and clients, seems to be necessary to translate fuzzy queries from clients to the query language accepted by the host DBMS.

This paper introduces a preliminary prototype

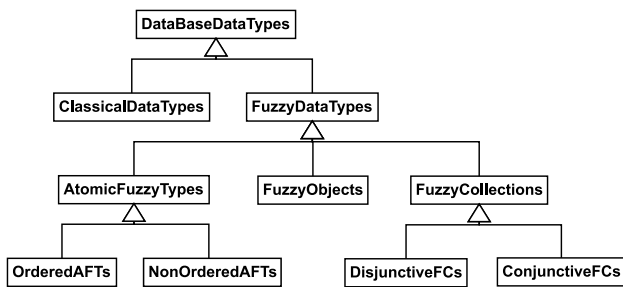


Figure 1: Datatype hierarchy

of a portable fuzzy query language in XML format, XML fuzzy structured query language (XFSQL), for the object-relational data model described later, which could represent the beginning of a common FDBMS query language. Additionally, it describes the framework for implementing middleware components to enable a FDBMS to be queried using XFSQL language.

The paper is organized as follows. Section 2 introduces the fuzzy data model on which XFSQL is based. The definition of XFSQL is presented in Section 3. In Section 4 a translator for XFSQL queries is introduced. Section 5 shows an example of a query executed using a XFSQL translator. Finally, Section 6 highlights concluding remarks and future works.

2 Fuzzy Data Model

Every query language is associated to a data model, which characterizes the type and organization of the data on which language queries will be defined. This section introduces the data model, derived from previous work [1, 2], on which XFSQL query language, and its associated result format, is based. This data model is specified as a datatype hierarchy in order to take advantage of object oriented modeling benefits. Figure 1 shows this datatype hierarchy.

FuzzyDataTypes (FDT) is an abstract type which is the root ancestor of all supported fuzzy datatypes. This type declares abstract methods which must be implemented in its instantiable subtypes. These methods are common to the subtypes, for instance the method FEQ (fuzzy equal to), which extends the concept of the classical equality to the fuzzy framework. FEQ method

gets two fuzzy values as arguments and returns a value in the interval $[0, 1]$, standing for the resemblance degree between these fuzzy values.

AtomicFuzzyTypes (AFT) is an abstract type designed for collecting common behavior of subtypes aimed to represent atomic data.

OrderedAFTs (OAFT) datatype gives support for fuzzy numbers, which are atomic fuzzy data represented by a possibility distribution defined on an ordered domain. The extended relational operators for OAFT data are for instance FEQ (fuzzy equal to), FLEQ (fuzzy less than or equal to), etc.

NonOrderedAFTs (NOAFT) datatype is designed to store fuzzy data defined on a scalar domain without any order between its elements. Instead of an order relation, a fuzzy nearness relation is defined between domain members by the user. The FEQ operator is the only comparator for data of this datatype, which uses the nearness relation associated to the domain to compute the resemblance degree between two data values.

FuzzyCollections (FC) is an abstract datatype which extends the concept of classical collections to a fuzzy one. Membership of collection elements is fuzzified, so the membership value of collection elements is in the interval $[0, 1]$, representing different degrees of membership of the elements.

DisjunctiveFCs (DFC) datatype, a subtype of FC, supports fuzzy collections with disjunctive semantics. The collection semantics is relevant, because it affects the way in which the fuzzy equality (FEQ) between DFCs is computed.

ConjunctiveFuzzyCollections (CFC) datatype is similar to DFC but supports collections with conjunctive semantics.

FuzzyObject (FO) is an abstract datatype which sets a general framework for dealing with user defined complex fuzzy objects. Every fuzzy object type must be a subtype of FO datatype to inherit fuzzy object management functionality.

3 XFSQL Language Definition

Nowadays a wide variety of database systems are available, each one using different approaches to perform data handling, depending on the used

data model. There are relational databases, object-relational databases, object-oriented databases, etc. with different implementations depending on developers, which implies different query language syntaxes making it unlikely that a fuzzy query created for a particular implementation could work on other implementations. To solve this problem, the creation of a portable, flexible, extensible and implementation independent fuzzy query language is proposed.

Using XML is possible to define a fuzzy query language with the required characteristics. XML (eXtensible Markup Language) is a data description language designed to describe document containing structured data using tags. The use of XML provides benefits such as data portability, allowing data interchange between different platforms and heterogeneous environments, data independence from representation, interoperability, scalability, flexibility and extensibility.

Defining a Fuzzy Query Language in XML (XFSQL) takes advantage of XML benefits. In this case the extensibility is particularly useful, in order to define different extensions in the syntax language. These extensions may allow interaction with different databases, defining a more complex syntax for databases with more functionalities, and using a core syntax to interact with those that only implement the basic functionality.

XFSQL provides a structured format to express queries showing the data to retrieve, avoiding syntax particularities of the different query language implementations. The transformation of XFSQL syntax into a particular database syntax can be done by means of XSL/T or by a program specifically designed for that purpose. Different XSL/T stylesheets or programs allow communication with different databases, solving the problem of implementation dependent syntaxes.

Studying different FDBMS proposals and models, a set of basic common fuzzy characteristics shared by different models is determined to be implemented by XFSQL. Modeling of fuzzy conditions in XML is done taking into account previous works on fuzzy data representation in XML [8, 9]. XFSQL is defined by a XML Schema which allows more control, over language structure definition

and the data types to be used, to be obtained. It is possible to define extensions to the language, allowing systems with advanced fuzzy features to display their capabilities, which divides XFSQL in core functionality and extensions.

XFSQL Core. This part of the specification includes the minimum fuzzy functionality implemented in the studied FDBMSs. XFSQL Core allows querying on AFT attributes – OAFT and NOAFT – and fuzzy querying on crisp attributes. From a general structure SELECT-FROM-WHERE it is possible to indicate attributes to retrieve, tables from which data is retrieved and to establish conditions.

```
<xfsql><query>
  <select>
    <columns>
      <attribute name="AName"/>
      ...
    </columns>
  </select>
  <from>
    <table name="TName"/>
    ...
  </from>
  <where>
    <condition>
      <fuzzy-equal>
        <left><attribute name="AName"/></left>
        <right>
          <trapezoid a="0" b="1" c="2" d="3">
        </right>
      </fuzzy-equal>
      ...
    </condition>
  </where>
</query></xfsql>
```

XFSQL FO Extension. Specifies an extension to deal with FO, allowing querying, access to attributes and calls to methods. The following code shows a excerpt of a XFSQL query in which a method is called.

```
<xfsql><query>
  ...
  <object name="ObjectName">
    ...
    <object name="SubObjectName">
      <method name="MethodName">
        <parameter value="ParameterValue"/>
        ...
      </method>
    </object>
    ...
  </object>
  ...
</query></xfsql>
```

There are other possible extensions for XFSQL. For instance, *XFSQL FC Extension*, could define the language syntax for fuzzy collection comparison, retrieval and condition establishment.

4 XFSQL Interface

Every XFSQL-able FDBMS needs a XFSQL interface which translates XFSQL queries to the native fuzzy extension query language, and query results to the XFSQL query result format.

Due to the significant differences between existing FDBMS, it is necessary to implement a particular XFSQL interface for each one. The main differences between different FDBMS – assuming that they share a basic group of fuzzy datatypes and fuzzy relational and logic operators – are the fuzzy query language, the query result format, and the mechanisms – communication ways – to send to the FDBMS queries and receive query results.

Therefore, the whole process, which takes part when a XFSQL sentence arrives to the XFSQL translator, is the following. Firstly, the sentence is sent to the *sentence translator* resulting in a native sentence. Secondly, the native sentence is sent to the *sentence dispatcher* which sends the native sentence to the host FDBMS and receives the sentence result from the same system, using host FDBMS interfacing mechanisms. Thirdly, when the sentence is a query, the query result in native format is sent to the *result translator* which convert it to a XML document representing query result in XFSQL format. Finally, the resulting XML document is sent back as response of the received XFSQL query.

5 Implementation Example

This section is devoted to develop an example of real estate searching based on the model of ImmoSoftWeb [10], a fuzzy real estate web search application that uses XML for independent interchange of information, but adapted to the XFSQL translator proposed in this article.

From a wide variety of attributes, the most representative and those which can be modeled using the fuzzy types described before are selected as shown in Table 1.

Table 1: Real estate fuzzy type attributes

Type	Attributes
Crisp	ID
Non Ordered AFT	Kind
Ordered AFT	Price, Area, Rooms

Table 2: Proximity relation for attribute *Kind*

Flat	House	Duplex	Attic	Kind
0.75	0.3	0.2	0.75	<i>Apartment</i>
	0.3	0.3	0.75	<i>Flat</i>
		0.75	0.1	<i>House</i>
			0.1	<i>Duplex</i>

Attribute *Kind* scalar domain elements are $\{Apartment, Flat, House, Duplex, Attic\}$. The associated proximity relation for these values on attribute *Kind* is shown in Table 2.

Once the structure of real estates is defined, a reduced set of them is described in order to be queried. A query can be defined in natural language as “*Search for Flats which price is lower than 175,000 euros with an area of approximately 100 m² with an error of 15 m² and 3 or 4 rooms*”. In Fig. 2 a representation of the query and database real estates is presented.

The query can be introduced via web form into the application. The web form is filled with the query conditions as in Fig. 3, then the application processes the form data and generates a XML document of XFSQL type, containing all the necessary information to perform the query.

The XFSQL document obtained is:

```
<xfsql><query>
<select>
<columns>
<attribute name="ID"/>
<function name="relevance">
<attribute name="Kind"/>
<attribute name="Price"/>
<attribute name="Area"/>
<attribute name="Rooms"/>
</function>
<attribute name="Kind"/>
<attribute name="Price"/>
<attribute name="Area"/>
```

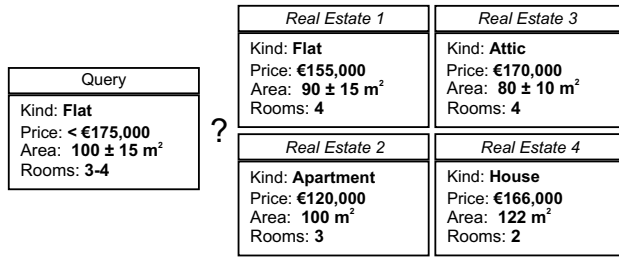


Figure 2: Real estate search example

Figure 3: Query expressed as web form

```

<attribute name="Rooms"/>
</columns>
</select>
<from>
<table name="RealEstate_tab"/>
</from>
<where>
<condition>
<and>
<fuzzy-equal>
<left><attribute name="Kind"/></left>
<right>
<linguistic label="flat" type="TKind"/>
</right>
</fuzzy-equal>
<fuzzy-less-or-equal>
<left><attribute name="Price"/></left>
<right><numeric value="150000"/></right>
</fuzzy-less-or-equal>
<fuzzy-equal>
<left><attribute name="Area"/></left>
<right>
<approximate value="100" margin="15"/>
</right>
</fuzzy-equal>
<fuzzy-equal>
<left><attribute name="Rooms"/></left>
<right>
<range lbound="3" ubound="4"/>
</right>
</fuzzy-equal>
</and>
</condition>
</where>
</query></xfsql>

```

This XFSQL query is independent from the FDBMS on which it is going to be executed. In

Table 3: Fulfillment degrees obtained

ID	Relevance	Kind	Price	Area	Rooms
RE1	0.917	1	1	0.67	1
RE2	0.937	0.75	1	1	1
RE3	0.737	0.75	1	0.2	1
RE4	0.355	0.3	1	0.12	0

order to adapt to the target FDBMS syntax, the XFSQL document must be transformed. Once the XFSQL query is translated into a particular syntax, it can be executed by the FDBMS. In this case, the target FDBMS is an object-relational database implemented on Oracle DBMS, such as the one described in [1, 2], expressing the query as:

```

SELECT ID,relevance(Kind,Price,Area,Rooms),
Kind,Price,Area,Rooms
FROM RealEstate_tab
WHERE FEQ(Kind,TKind('flat')) and
FLEQ(Price,TPrice(150000)) and
FEQ(Area,TArea(85,100,115)) and
FEQ(Rooms,TRooms(3,4));

```

The function `relevance()` computes the fulfillment degrees obtained for each condition and calculates a global fulfillment degree value for the real estate as a whole, by calculating the average of each condition fulfillment degree.

The query is executed in the host FDBMS and the fulfillment degrees are calculated obtaining the results of Table 3.

The results are expressed as a XML document too, in order to be sent back to the application, and then rendered. The XML response document generated in this case is:

```

<xfsql><results>
<columns>
<description name="ID" ID="ID1"/>
<description name="relevance()" ID="ID2"/>
<description name="Kind" ID="ID3"/>
<description name="Price" ID="ID4"/>
<description name="Area" ID="ID5"/>
<description name="Rooms" ID="ID6"/>
</columns>
<rows>
<row>
<element IDREF="ID1"><string value="RE1"/>
</element>
<element IDREF="ID2"><numeric value="0.917"/>
</element>
<element IDREF="ID3"><label value="flat"/>

```

```

</element>
<element IDREF="ID4"><numeric value="155000"/>
</element>
<element IDREF="ID5">
  <approximate value="90" margin="15"/>
</element>
<element IDREF="ID6"><numeric value="4"/>
</element>
</row>
...
</rows>
</results></xfsql>

```

When results come back to the application they are shown in form of web page as in Fig. 4.

Results :: Buy					
ID	Relevance ▼▲	Kind ▼▲	Price € ▼▲	Area m ² ▼▲	Rooms ▼▲
RE2	93%	Apartment	120,000 €	100.0 m ²	3
RE1	91%	Flat	155,000 €	90.0 ± 15 m ²	4
RE3	73%	Attic	170,000 €	80.0 ± 10 m ²	4
RE4	35%	House	166,000 €	122.0 m ²	2

Figure 4: Results shown in web application

6 Concluding Remarks and Future Works

In this paper a fuzzy query language XFSQL has been presented, as a first step to ease access to different FDBMSs. In addition, a framework for implementing translators to perform independent queries has been proposed. Once created translators for different FDBMS and extended the proposed language prototype with new elements from others FDBMS not considered before, the basis of a fuzzy common query language could be set. The creation of a Fuzzy Data Definition Language (DDL) integrated in the proposed translator, and definition and implementation of a *XFSQL FC Extension* for fuzzy collection handling are main guidelines for future works. Additionally, future works will focus on the study of new elements used in other FDBMS such as fuzzy types, complex possibility distributions, intelligent labels, etc.

Acknowledgments

This work has been partially supported by the Spanish "Ministerio de Ciencia y Tecnología" (MCYT) under grant TIC2002-00480.

References

[1] J.M. Medina, J. Galindo, F. Berzal, J.M. Serrano, "Using Object Relational Features to

- Build a Fuzzy Database Server", VIII Intl. Conf. of information processing and management of uncertainty in knowledge-based systems (IPMU 2002), pp 307-314. July 1-5 2002. Annecy (France).
- [2] J.C. Cubero, N. Marín, J.M. Medina, O. Pons, M.A. Vila, "Fuzzy object Management in an Object-Relational Framework", X Intl. Conf. of information processing and management of uncertainty in knowledge-based systems, pp.1767-1774. July 4-9 2004. Perugia (Italy).
- [3] H. Prade, C. Testemale, "Generalizing Database Relational Algebra for the Treatment of Incomplete or Uncertain Information and Vague Queries", Information Sciences Vol. 34, 1984, pp. 115-143.
- [4] M. Zemankova-Leech, A. Kandel, "Fuzzy Relational Databases – A Key to Expert Systems", Köln, Germany, TÜV Rheinland, 1984.
- [5] S. Fukami, M. Umamo, M. Muzimoto, H. Tanaka, "Fuzzy Database Retrieval and Manipulation Language", IEICE Technical Reports, Vol. 78, N. 233, pp. 65–72, AL-78-85 (Automata and Language) 1979.
- [6] M. Umamo, "Freedom-O: A Fuzzy Database System", Fuzzy Information and Decision Processes. Gupta-Sanchez edit. North-Holland Pub. Comp. 1982.
- [7] J. Galindo, J.M. Medina, O.Pons, J.C. Cubero, "A Server for Fuzzy SQL Queries", Flexible Query Answering Systems, eds. T. Andreasen, H. Christiansen and H.L. Larsen, Lecture Notes in Artificial Intelligence (LNAI) 1495, pp. 164–174. Ed. Springer, 1998.
- [8] J. Lee and Y. Fanjiang, "Modeling imprecise requirements with XML", Information and Software Technology, Vol. 45, Issue 7, pp. 445-460, 2003.
- [9] K. Turowski, U. Weng, "Representing and processing fuzzy information – an XML-based approach", Knowledge-Based Systems, Vol. 15, Issues 1-2, pp. 67-75, 2002.
- [10] C.D. Barranco, J.R. Campaña, J.M. Medina O. Pons, "ImmoSoftWeb: a Web Based Fuzzy Application for Real Estate Management", Advances in Web Intelligences, LNAI 3034, pp.196-206, J. Favela et al. (Eds.) 2004.