A Fuzzy Object Relational Approach to Flexible Real Estate Trade

CARLOS D. BARRANCO, JESÚS R. CAMPAÑA, JUAN C. CUBERO, JUAN M. MEDINA
Dept. Computer Science and Artificial Intelligence
University of Granada
Periodista Daniel Saucedo Aranda, s/n, 18071, Granada
SPAIN

Abstract: - The object-relational database management systems (ORDBMS) appear due to the common acceptance of the object oriented paradigm and its integration with relational databases, combining the powerful modelling capabilities of the object oriented model and the robustness of relational model. ORDBMSs user defined types allow to create a framework for fuzzy information handling. This paper proposes to use that fuzzy framework to improve computer assisted search-offering processes, focusing on the real estate trading area.

Key-Words: - Object-Relational, Fuzzy Sets, Fuzzy Databases, Fuzzy Objects, Offer Search, Real Estate Search

1 Introduction

The object oriented (OO) paradigm is becoming the prevalent one in the application development field. This success is due to the high expressive capabilities of object models, and the rapid application development capabilities of OO languages, because of the high code reusability level achieved by the OO paradigm key concepts: inheritance, encapsulation and polymorphism.

As a result of this success, database models are changing to include OO concepts in order to take advantage of the OO paradigm benefits mentioned before.

Nowadays, commercial database management systems (DBMS) are moving to the object-relational paradigm because of the advantages offered to users. Object-Relational Databases (ORDB) combine the powerful modelling capabilities of an OO data model and the proved robustness of the relational model. ORDBMSs integrate much better with OO software, offering to OO applications direct object persistence functionality.

A key feature of ORDBMSs is extensibility. Using OO concepts, ORDBMS functionality may be extended by means of User DataTypes (UDT), which allow transparent integration of user defined data structures and data processing, all of this encapsulated as a unit.

In the field of imperfect information management, fuzzy object-relational database (FORDB) models are appearing [1, 2].

FORDBs benefit from the advantages of ORDBs while they enrich this paradigm providing uncertain information management capabilities.

During the last years, several works led to a model [3, 4, 5] and an implementation [6, 7] of a Fuzzy Relational Database Management System, a later proposal [8] aims to represent fuzzy information in an object-oriented data model, and recent work [2] point to a model and implementation of a FORDBMS using the object features of current ORDBMSs to extend them by means of UDTs, which encapsulate fuzzy information representation and processing.

The transparent integration of OO applications with ORDBMSs and their new fuzzy data management extensions, combine to create enhanced data management capabilities for commercial applications, allowing them to store imprecise information and query data using flexible conditions easily.

Our proposal is to use these new data management capabilities to improve the user-application interaction in offer-searching systems, allowing sellers to express their offers as imprecisely as they need, and buyers to express their queries as flexibly as they want. This way of expressing queries and offers, makes the interaction with the systems more natural, emulating the flexible process applied by sales agent to match offers and demands.

Section 2 introduces briefly a proposed FORDB obtained from an ORDB extension. Section 3 exposes an example of a trading area which requires fuzzy information and flexible query in its typical way of work, real estate trading area. Section 4 focuses on an implementation example of a query comprising flexible conditions. Finally, Section 5 highlights the concluding remarks.

2 Fuzzy Object-Relational Database

An ORDBMS can be extended using UDTs to manage virtually any kind of complex data, like multimedia or spatial data. Extending an ORDBMS with fuzzy data management UDTs, produces a

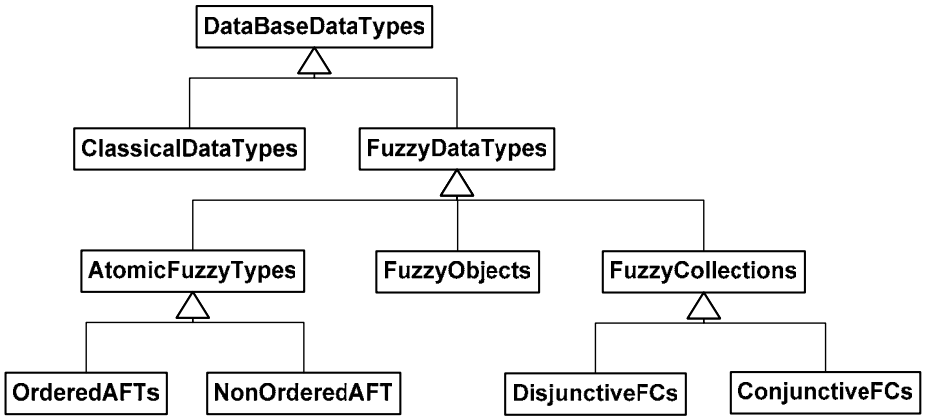

Figure 1: Extended Database Types

FORDBMS which combines the power of fuzzy sets, and the object oriented and relational paradigms.

This extension provides advantages over existing FRDBMSs, such as tight level of integration with the underlying DBMS, hiding implementation aspects of fuzzy types, which allow the user to be aware only of semantics and functionality, an extensible schema allowing future extensions, and efficient implementation, avoiding the use of software wrappers to allow fuzzy data management.

2.1 DataType Hierarchy for Fuzzy Data Management

In order to provide complex fuzzy data management capabilities for the underlying ORDBMS, new UDTs have been defined using host DBMS object-oriented features, organized in a hierarchy and extending the basic DBMS datatypes. These new datatypes allow the DBMS user to deal with several kinds of imprecise data. Figure 1 shows the database datatype hierarchy integrating classical and fuzzy types.

The types in the mentioned hierarchy are the following:

- FuzzyDataTypes (FDT) are an abstraction of all supported fuzzy data. This type declares common general methods to be implemented in the subtypes, for instance the FEQ (fuzzy equal to) method, which extends the concept of classical equality to the fuzzy framework, returning a value in the interval [0,1] representing the fuzzy resemblance degree between two fuzzy values.

- AtomicFuzzyTypes (AFT) gather all common behavior for the fuzzy extensions of scalar and numerical data.

- OrderedAFTs (OAFT) give structure and behavior to atomic fuzzy data represented by a possibility distribution defined on an ordered domain (numerical fuzzy data). As this type has an associated ordered domain which defines an order relation between the domain elements, the type can define an extension of the classical relational operators, for instance fuzzy equal to (FEQ), fuzzy greater than (FGT), fuzzy greater than or equal to (FGEQ), etcetera.

- NonOrderedAFTs (NOAFT) provide structure and behavior to data defined on a scalar domain without an order relation. The user defines a fuzzy nearness relation between domain's members, which is used to compute the resemblance degree between two members using the FEQ operator.

- FuzzyCollections (FC) extend the classical collection concept to a fuzzy one, in which the collection elements have a membership degree between [0,1]. Fuzziness affects only elements' membership, it does not affect collection elements, therefore collection elements' type can be fuzzy or crisp. FC type provides the required structure and behavior to manage the collection, like methods for adding, removing or getting the membership of collection elements.

- DisjunctiveFuzzyCollections (DFC) model fuzzy collections with disjunctive semantics, which determines the fuzzy equality method (FEQ) implementation.

- ConjunctiveFuzzyCollections (CFC) are the equivalent to DFC but with conjunctive semantics.

- FuzzyObjects (FO) provide a general framework for dealing with complex fuzzy objects defined by users. Every user defined fuzzy object type is a subtype of FO, inheriting common methods defined in FO for fuzzy object management. These methods are, one to weigh the importance degree of each object attribute, which is used by the fuzzy objects comparison algorithm, and a method encapsulating a general implementation of the FEQ comparator for fuzzy objects.

2.2 Fuzzy Data Comparison

In order to compare fuzzy data of the same fuzzy datatype, every fuzzy type, in the described hierarchy, has its own particular implementation of the FEQ method described earlier, adapted to calculate the resemblance degree between two fuzzy type elements.

For AFT subtypes the comparison is performed using user-defined resemblance relations, when dealing with NOAFT type, and by known resemblance computation method for possibility distributions on order domains, in case of OAFT type.

When dealing with FCs we have to take into account the possible recursive comparison process (because the collection elements can be any kind of fuzzy data) and the collection semantics, which determines the comparison method employed. Disjunctive semantics collections can employ a generalized resemblance method like ($\beth_{FEQ\Omega,\otimes}(o_1,o_2)$ [9]).

Complex FOs need a similar treatment. They can be composed of complex fuzzy data, therefore the comparison method increases its complexity performing the following tasks:

1. Compute the resemblance in basic domains (i.e. AFT attributes and collection elements).
2. Compute the resemblance between fuzzy collections of imprecise objects, when the attributes of a FO, or elements of a FC, are complex fuzzy types.
3. Aggregate the resemblance information collected in the previous steps, referring to the resemblance degree between objects' attributes, in order to obtain a resemblance degree for the compared objects.

Cycles in the resemblance degree calculation process may happen, taking into account that FOs can reference themselves creating a cycle. Therefore, the method needs a guard mechanism to avoid cycles.

A suitable resemblance degree method is $FEQ(C,o_1,o_2,\Omega_{visited},\Omega_{aprox})$, described in detail in [2]. The method compares, peer to peer, attributes' values, obtaining a resemblance degree for each object attribute. For each attribute, the comparison process can be recursive and cycles may appear, but the method manages well these situations. When a resemblance value has been calculated for each attribute, an object resemblance value is calculated aggregating these values using the aggregation operator V_Q [9]. Figure 2 summarizes the described process.

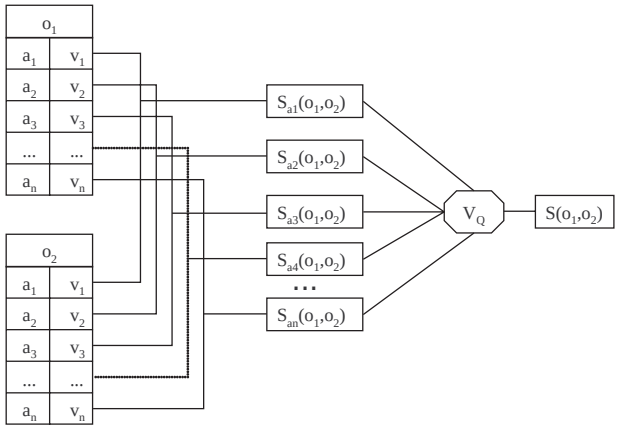

Figure 2: Complex object comparison

3 The Real Estate Searching Problem

The real estate management process is chosen as the object of our research, because of the suitability of real estate attributes to fuzzy treatment, due to the high level of imprecision in their values.

In the real estate searching process a set of characteristics is specified for the real estate to have, but usually these characteristics are not fully defined. A customer has a set of preferences, a general idea of what is being looked for, that idea not necessarily should fit to a crisp value, it might be most accurately represented by a value range, an approximate value or even an upper or lower bound. The imprecise representation of these characteristics may allow to obtain results that verify our preferences on different degrees.

Generally the imprecision is managed by sales agents who can easily process and handle fuzzy information. The real estate management process occurs between two humans, the customer and the

sales agent, both of them can handle fuzzy information naturally.

The problem arises when one of these entities, the sales agent in our case, capable of handling fuzzy information, is replaced by an automatic system. It is necessary to provide the system with methods to handle fuzzy information in the same way the sales agent was doing before. So, a way to represent fuzzy information about real estates is proposed, in order to be able to design a system that can mimic the sales agent behavior, to interact fluidly with a customer.

Which attributes are suitable for fuzzy handling will be examined, and also, the way to represent them in the framework defined in previous sections.

3.1 Real Estate Fuzzy Attributes

There are some real estate attributes which can be modelled using the fuzzy types described in the previous section. From a wide variety of attributes, most representative and those which can illustrate better the example have been selected. The attributes selected are the ones shown in Table 1.

Table 1 : Real estate attributes and fuzzy types associated

Type	***Attributes***
Ordered AFT	Price, Area, Rooms, Floors, Age
Non Ordered AFT	Kind, Orientation, Illumination, Views, Conservation
Conjunctive Fuzzy Collections	Additional features

OAFTs are used to store imprecise data, represented as trapezoidal possibility distributions. For instance, attribute *Price* stores the real estate price range: "between €100,000 and €500,000" or "up to €150,000". The same goes for *Area* and *Age*. However, for attributes *Room* and *Floors* a scalar representation is used, also allowed by this data type, because these attributes are easily measurable and the imprecision introduced is minimal, although if it is necessary an imprecise representation can be used.

NOAFTs are defined on a scalar domain, with an associated proximity relation defined between elements of that domain. For instance, attribute *Kind* has scalar domain "Apartment", "Flat", "House", "Duplex" and "Attic". The associated proximity relation for these values of the scalar domain on attribute *Kind* is shown in Table 2. Each attribute *Orientation*, *Illumination*, *Views* and *Conservation* has his own scalar domain and a proximity relation defined on it.

Table 2 : Proximity relation defined on the scalar domain of attribute Kind

Flat	*House*	*Duplex*	*Attic*	***Kind***
0.75	0.3	0.2	0.75	*Apartment*
	0.3	0.3	0.75	*Flat*
		0.75	0.1	*House*
			0.1	*Duplex*

CFCs are used to store any real estate's additional features, like "garden", "tennis court", "fireplace", "swimming-pool", "basement", "backyard", etcetera. Each additional feature is added to the CFC with a membership degree equal to 1, and during the search every pair of additional features sets is compared in order to obtain a resemblance degree.

More details and description of an application using this approach can be seen in [10, 11].

4 Real Estate Search

The following example shows a set of real estates and defines a query with fuzzy terms over that set.

Query and real estate attribute definition are imprecise due to the acceptance of linguistic labels and numeric values.

In the example, a subset of the attributes presented is used, because the procedure is the same for attributes with equal data type and reducing the amount of them simplifies the example and eases its comprehension. Query definition and a set of real estate FOs can be seen in Fig. 3.

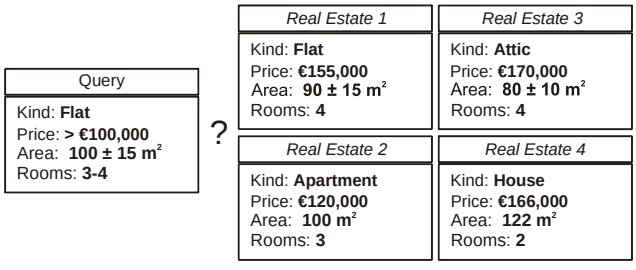

Figure 3 : Real Estate Search Example

To do a search over a set of FOs that represent real estates, a query is built showing the conditions to verify each attribute.

Representing queries as attributes of a loosely defined real estate FO allows to characterize a query as the real estate FO to look for. If a query is represented as a real estate FO, the searching process is reduced to a comparison between fuzzy objects, obtaining the resemblance degree between the query FO and the FOs modelling real estates present in the database.

Comparison between FOs is performed computing the resemblance degree between attributes of FOs, then, resemblance degrees are aggregated using the aggregation operator V_Q.

4.1 Data Definition and Querying

An implementation example in SQL is accomplished according to the schemata defined in [2] and the extended type hierarchy depicted in Fig. 1, completed with the methods and constructors needed to fulfill the example.

Static method `extends(typeName)` is used by AFT type for new subtype creation. When invoked on a type, creates a subtype with the name `<typeName>`.

Method `nearnessDef(memberList,degreeList)` is needed by NOAFT type. This method defines and stores the nearness relation for the domain members of the type.

A real estate is defined as follows:

```
--Creation of OAFT subtypes
OAFT.extends('Price');
OAFT.extends('Area');
OAFT.extends('Rooms');

--Creation of NOAFT subtype Kind
NOAFT.extends('Kind');

--Definition of the nearness relation for Kind
--Domain
--The first parameter is comprised of linguistic
--labels defined in the Domain.
--The second parameter is a nearness degree list.
--First we put on the list the nearness degrees
--between 'apartment' an the rest of labels, then
--that of 'flat' with the rest of label except
--for 'apartment' (already defined), and so on.

Kind.nearnessDef(('apartment','flat','house','dup
lex','attic'),(0.75,0.3,0.2,0.75,0.3,0.3,0.75,0.7
5,0.1,0.1))

--FO subtype RealEstate
create type RealEstate under FO( RKind Kind,
RPrice Price, RArea Area, RRooms Rooms );
```

All attributes have the same importance in this example but it can be changed by means of method `setFieldImportance(attribImportanceList)` defined for FO types.

Once all types are defined, a table *RealEstate* is created to store the example data.

```
create table RealEstates_tab of RealEstate;
```

RealEstate instances are inserted into the table, using the defined constructors. The DML statements are the following:

```
-- The OAFT type constructors used are defined as
-- follows:
--
-- OAFT(value) creates a crisp value.
-- OAFT(a,b) creates an interval value [a,b].
-- OAFT(l,value,u) creates an approximate value
-- with 'u' and 'l' as upper and lower bounds.
-- OAFT(a,b,c,d) creates a trapezoidal
-- possibility distribution value.

insert into RealEstates_tab values (
RealEstate(
   Kind('flat'), Price(155000),
     Area(75,90,105), Rooms(4)
     )
);

insert into RealEstates_tab values (
RealEstate(
   Kind('apartment'), Price(120000),
   Area(100), Rooms(3)
     )
);

insert into RealEstates_tab values (
RealEstate(
   Kind('attic'), Price(170000),
     Area(70,80,90), Rooms(4)
     )
);

insert into RealEstates_tab values (
RealEstate(
   Kind('house'), Price(166000),
     Area(122), Rooms(2)
     )
);
```

Now real estate instances are stored and the database is ready to accept queries. A query is built to search the most similar real estate to that expressed, as follows:

```
-- 'binary_double_infinity' is the literal which
-- represents positive infinity.

SELECT (RealEstate(Kind('flat'),
        Price(100000, binary_double_infinity),
        Area(85,100,115), Rooms(3,4))
       ).feq(re1)
FROM RealEstate_tab re1;
```

Calculating the resemblance between query object and real estate instances involves the aggregation of the resemblance degree between each pair of attribute values, using aggregator V_Q detailed in [9].

For instance, resemblance degree between the query and *Real Estate 1*, using $\gamma_Q = 0.2$, $\mu_D(x) = 1$, minimum as *t-norm*, maximum as *t-conorm*, and possibility measures for comparing atomic fuzzy values.

$$V_Q(A/D) = 0.2\max\{\min(1,1), \min(1,1), \min(1,0.67), \min(1,1)\} + (1-0.2)\min\{\max(1,0), \max(1,0), \max(0.67,0), \max(1,0)\} = 0.2*1 + 0.8*0.67 = 0.736$$

Table 3 shows resemblance degrees computed for the query. It is possible to set up a threshold to avoid real estates with low resemblance degree to appear in the query result.

Table 3 : Resemblance degrees calculated

Real Estate	*Resemblance Degree (γ_Q=0.2)*
RE 1	V_Q[1, 1, 0.67, 1] = 0.736
RE 2	V_Q[0.75, 1, 1, 1] = 0.8
RE 3	V_Q[0.75, 1, 0.2, 1] = 0.68
RE 4	V_Q[0.3, 1, 0.12, 0] = 0.296

The resemblance degrees obtained indicate that *Real Estate 2* is the best suited to the query although it is not a perfect match.

In a crisp system this example would not show results, because none of the database real estates fits the query with resemblance degree 1. Therefore an improvement in the searching process is experienced, if there not exists a real estate with the characteristics specified in the query, another instance with the most similar characteristics can be obtained.

This makes the fuzzy real estate search a very useful tool for customers and sales agents alike.

5 Concluding Remarks and Future Works

In this paper we have presented an object oriented representation for real estates as fuzzy objects, based on previous works. Also, benefits of the object relational approach applied to fuzzy data management, a method to compare real estate FOs based on previous research, and a set of real estate attributes and the fuzzy data which describe them have been exposed.

Real estate search has been simplified and improved adding fuzzy object data management capabilities and depicting it as a comparison of fuzzy objects.

Future works will address the calculus of resemblance degrees based on user preferences, automatic definition of linguistic labels based on context, and fuzzy treatment of real estate location.

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] 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 (IPMU 2004)*, 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. Umano, 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. Umano, Freedom-O: A Fuzzy Database System, *Fuzzy Information and Decision Processes*. Gupta-Sanchez edit. North-Holand 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] R. D. Caluwe, Fuzzy and Uncertain Object-Oriented Databases: Concepts and Models, *Advances in Fuzzy Systems-Applications and Theory*. Vol 13. World Scientific, 1997.

[9] N. Marín, J.M. Medina, O. Pons, D. Sánchez, and M. A. Vila, Complex object comparison in a fuzzy context, *Information and Software Technology*, 45, 431-444, 2003.

[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.

[11] ImmoSoftWeb: http://idbis.ugr.es/immosoftweb