FUZZY DATABASE FRAMEWORK—
RELATIONAL VERSUS OBJECT-ORIENTED
MODEL

T.C. Ling
Mashkuri Hj. Yaacob
K.K. Phang
Faculty of Computer Science and Information Technology
University of Malaya
50603 Kuala Lumpur, Malaysia
email: {tchaw, mashkuri, phang}@fsktm.um.edu.my

ABSTRACT
Describes i) fuzzy data and linguistic qualifiers (ii) fuzzy
data representation and retrieval in a fuzzy database (iı)
database aspect of a fuzzy database iv) Relational and
Object-Oriented model and v) the advantages of using
object-oriented database framework in fuzzy database. A
prototype of fuzzy object-oriented database system
(FOODS) has been implemented to demonstrate the
feasibility of fuzzy object-oriented database system.
KEYWORDS: Relational database management systems,
Object-oriented database management systems, Fuzzy set theory, Fuzzy object-oriented
database management system and database framework

1.0 INTRODUCTION

Today, the feasibility and usefulness of Fuzzy Mathematics [1, 2, 3] are attested in a wide area of
successful applications, such as engineering, computer science, artificial intelligence, information processing and
database systems, to name just a few [4, 5, 6]. Database
management system is one of the fields where Fuzzy
Mathematics can be effectively and fruitfully applied.
Some proposals and implementations of fuzzy database
management system [7, 8, 9] have been reported. As is
well known, traditional DBMS can only handle “hard”
data that are precise and deterministic in nature. Real
world applications invariably involve “soft” data, and as
such, it is to be expected that FDBMS will give better
performance in so far as having inputs which can tolerate
linguistic vagueness and imprecision.

This paper, based on the state-of-the-art development
in Database Management System (DBMS), intends to
examine the database aspect of Fuzzy Database System
(FDS). The question of which database model is best
suited to the Fuzzy Database System is discussed.

2.0 Fuzzy Set Theory

Fuzzy set theory provides a formal mathematical
framework for a systematic treatment of fuzzy data. A
fuzzy subset A of X is defined as:

\[ A = \{ m_A(x) \mid x \in X \} \text{ and } m_A(x) \rightarrow [0,1] \]

where \( m_A(x) \) is the membership function.

3.0 IN SEARCH OF A SUITABLE
FUZZY DATABASE FRAMEWORK

Database systems have evolved over a number of
years. In this section, a review on the development of the
DBMS is given. It is hoped that this review will help to
eucidate the limitations and advantages of the DBMS at
each development phase. DBMS's have undergone four
major phases of change [10, 11, 12, 13, 14].

Phase 1, Traditional Indexed File System; is the pre-
database period where data used by an organization are
often stored in many independent indexed files. A major
limitation of Indexed File Database Systems lies in its
lack of support for automatic linkage of files.

Phase 2, Hierarchical and Network Database System;
in hierarchical model, record types are linked as an
ordered set of treelike structure and the network data
structure can be considered as a more general form of the
hierarchical one [10], where arbitrary linkages of record
types can be formed. These systems realise the sharing of
an integrated database among multiple users. Traversing
the tree structure carries out data retrievals. The key
problem of these Database Systems lies in data
dependence and tedious navigation in accessing the database.

Phase 3, Relational Database Systems [10]; the elegance of the Relational Model lies in its simplicity and its sound mathematical foundation. The essence of RDBS is its designation of a data structure as relations, giving rise to easy access of data. RDBS is characterised by declarative query with support for data independence. However, as new applications like CAD, CASE tools and multi-media system emerge, RDBS becomes inadequate, the reason being that these applications require non-atomic, abstract data types which are not easily handled by the RDBS. Therefore, a new database system, which supports data independence as well as a richer set of data structure, should be constructed. A number of approaches has been taken. One approach is to extend the Relational Model to include user-defined data types. Another approach uses the Object-Oriented Model.

The final phase, Object-Oriented Database Systems [15], deals with OODBS. OODBS is based on the concepts of object-oriented paradigm. It supports multiple users and is characterised by a richer set of data types and database facilities. A higher level of modelling construct such as entities, relationships and inheritances accompanies the move from relational to OODBS. Furthermore, OODBS support the notion of data encapsulation that can be viewed as an integration of the structural and behavioural aspect of data.

At this juncture, it is pertinent to mention the constituents of a general-purpose DBMS. General-purpose database system in the main is intended to provide the following functions [16,17]:

1. Support the independent existence of a database, apart from the application programs and systems that manipulate it;
2. Provide a conceptual/logical level of data abstraction;
3. Support the query and modification of databases;
4. Accommodate the ability to evolve for both the conceptual structure and internal organization of a database;
5. Control a database that involves semantic integrity, security, and recovery

A careful study of the functional requirements provided by various models of database systems mentioned above reveals that only the RDBS and OODBS fulfilled the above requirements. Therefore, the searches for a suitable candidate of database framework for the Fuzzy Database are narrowed down to the relational and object-oriented models.

4.0 RDBS VERSUS OODBS

The major differences between a RDBS and an OODBS [18, 19, 20] are in the passive and active behaviours of the underlying system and the way these are implemented [21]. Data retrieval and manipulation operations are carried out using the query system or application programs via the RDBS. Since methods are not encapsulated in the data themselves, these data do not take active part during query evaluation. At the end of a session, data are written back to the passive database.

On the other hand, OODBS contains objects that encapsulate both passive (attributes) and active data (methods). Objects in the OODBS take active part during query processing. Relevant encapsulated methods in the objects are invoked. If necessary, these objects will automatically communicate with other objects and facilitate in generating answers to the query.

With the aids of active data, an application programmer can program, by a series of high level operations encapsulated in the object, instead of low level programming of operations on object in the program itself. Note that, as traditional RDBS frees users from details of the file system, the OODBS frees users from details of modelling and implementing real-world objects from the traditional RDBS. Consequently, OODBS is considered to possess higher modelling power than the RDBS.

5.0 FUZZY DATA MANIPULATION IN RDBS

Having discussed the basic concepts of the relational model and object-oriented modal, let us look at the current models of FRDBS.

Buckles et al. [7, 22] propose one of the earliest version of Fuzzy Relational Database System (FRDBS) by merging the theory of fuzzy set and Relational Database System (RDBS). They formulated a robust theoretical framework of similarity-based FRDBS.

A different approach in the representation and manipulation of fuzzy data is advanced by Umano [8] who develops Freedom-0, a FRDBS. Unlike the model by Buckle et al. which limits the fuzzy data to specific fuzzy number, Freedom-0 allows for both possibility distribution and Fuzzy number. However, even though Freedom-0 is more powerful in terms of its fuzzy data structure, it lacks the formal database framework that is found in the models of Buckle et al. Freedom-0 uses an embedded programming language in Fortran for fuzzy data manipulation.

Zemankova, [9] also develops a FRDBS which can handle both fuzzy set and possibility distribution data. RIM, a conventional non-fuzzy RDBS, is chosen as a
host in implementing the FRDBS. Vector attribute type supported by RIM is used to represent fuzzy data. Extension is made to the RIM RDBS data structure so that tuple component is not restricted to atomic values.

In the next step, we would like to inspect to what extent current models of the FRDBS satisfy the properties of the Relational model [10]:

P1 There are no duplicate tuples.
P2 Tuples are unordered
P3 Attributes are unordered
P4 All attributes value is atomic

It is observed that no FRDBS model satisfies property P4. P4 is relaxed to allow non-atomic values. Buckle's Similarity Based model only satisfies P1, P2 and P3. Although P4 is relaxed, this model still preserves the sound mathematical foundation of the Relational Model (Because property P1 is preserved. Also note that a more restricted set of fuzzy data is used in Buckle's model when compared with Umano and Zemankova's).

Umano and Zemankova's model, however, only satisfy P2 and P3. Since P1 is violated the sound mathematics foundation of the Relational Model is not preserved. These observations are summarised in Table 4.

<table>
<thead>
<tr>
<th>Models</th>
<th>Properties Violated</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buckle</td>
<td>P4</td>
<td>• Database is not normalised</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Sound mathematical foundation of the relational model is preserved.</td>
</tr>
<tr>
<td>Zemankova and Umano</td>
<td>P1,P4</td>
<td>• Database is not normalised</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Sound mathematical foundation of the relational model is not preserved.</td>
</tr>
</tbody>
</table>

It can therefore be concluded that all existing FRDBS models are based on the Extended Relational Model.

Furthermore, even in the Extended RDBS, it is laborious to represent and manipulate fuzzy data. Consider the relation Person in Table 5.

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom</td>
<td>23</td>
</tr>
<tr>
<td>Susan</td>
<td>young</td>
</tr>
</tbody>
</table>

In Zemankova's model, the fuzzy interval young is laboriously represented using another relation.

<table>
<thead>
<tr>
<th>Age</th>
<th>Membership Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>30</td>
<td>0.6</td>
</tr>
</tbody>
</table>

In Umano's model, on the other hand, the fuzzy interval young needs to be explicitly declared outside a relation, i.e., in the programming environment. For example young is declared as:

DEFP young = (1.0/1, .., 0.6/30, .., 0/50)
PEND

Where DEFP (define fuzzy predicate) is used to define a fuzzy interval and PEND ends the definition. DEFP is part of the data manipulation language in Umano's model. Furthermore, since the relational model is not expressive enough to capture the behavioural part of the fuzzy data, complex operations of fuzzy sets and linguistic modifiers must be coded explicitly by application programmers.

In addition, extended RDBS [13] is not as natural as the OODBS, as noted by Kim [15]:

"One important point which we must recognise is that an object-oriented data model is a more natural basis than an extended relational model for addressing some of the deficiencies of the conventional database technology... ; for example, support for general data types, nested objects, and support for compute-intensive applications."

6.0 FUZZY DATA MANIPULATION IN OODBS

Based on the advantages of the Object-Oriented model, we proposed and implemented a Fuzzy Object-Oriented Database (FOODS) [23]. FOODS is a research
prototype of a working, general purpose fuzzy DBMS that is based on the Object-Oriented model and Fuzzy Set Theory. It is implemented using Smalltalk [24] and can be viewed as an enhancement towards Smalltalk to make it a database system. As a PC-based, sequential access database system, FOODS is intended for applications involving information that is inherently imprecise. Major components of FOODS are as follows:

1. The database kernel: it is designed to cater for persistent objects support. Due to the fact that Smalltalk objects are inappropriate for database usage, a new type of objects, which is persistent is required. Characteristics of persistent objects include object identity, referential integrity, data type, inheritance and storage. Every persistent object has an object identity that can be used to reference the object itself. The existence of object identity ensures that there is something about an object that remains invariant across all possible modification of the object's value. In other words, object identity is independent from the actual value of the object. In FOODS, the pair <className, anInteger> gives the identity of a persistent object, where className is the class an object belongs to, and anInteger is a system assigned number. Object identity is unique across the database and it is crucial for supporting referential integrity and composite object. The notion of class inheritance plays a very important role in OODBs. It allows the programmer to specialize and reuse a class. In FOODS we limit ourselves to single inheritance, i.e., a class inherits from only one class.

2. The Data Definition Language allows the creation of new classes for a given database schema;

3. Object SQL allows ad hoc fuzzy queries based on an extended version of SQL [Date90];

4. FuzzySet provides protocol for the creation, representation and manipulation of fuzzy data; All categories of Fuzzy data can represented in FOODS directly. For example:

   smart = 0.9;
   high income = {0/1300, 0.5/1500, 0.7/1700, 0.8/1800, 1/2000}

Operations encapsulated in FuzzySet include operations such as intersect, union, compliment, equality and level set. In addition, linguistic modifiers such as very, sortOf and approximate are also supported. To illustrate how fuzzy data are stored and retrieve in FOODS. Let us consider the salary list of project managers in Table 1. It can be converted into fuzzy sets as in Table 2.

<table>
<thead>
<tr>
<th>Table 1. SALARY (Raw Data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
</tr>
<tr>
<td>INCOME</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2. SALARY (Fuzzy Data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>Dick</td>
</tr>
<tr>
<td>Sam</td>
</tr>
<tr>
<td>Tim</td>
</tr>
</tbody>
</table>

Next, consider the query "Print the Name and Income of all project managers whose income is very high, with threshold(Income) = 0.4." Or, in SQL parlance [10]:

Query: Select * from SALARY where

Income = very high, threshold (Income) = 0.4;

Querying the fuzzy database involves intensive computations of fuzzifications and defuzzification of the non-atomic fuzzy data. As these operations are encapsulated in Class FuzzySet, they will be handled by FOODS automatically without interventions from the application programmer.

This is a fuzzy query with linguistic modifier "very" and it will be processed as follows:

Step 1. The target "very high" is computed: "very high" = high^2 = {0/1300, 0.25/1500, 0.49/1700, 0.64/1800, 0.79/1900, 1/2000}

Step 2. The query is defuzzified by computing the alpha-level-set of income: high^0.4 = {1700, 1800, 1900, 2000}.

Step 3. Each row in the table is compared with this crisp set to obtain the answer in Table 3.

<table>
<thead>
<tr>
<th>Table 3. Answer to query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>Dick</td>
</tr>
<tr>
<td>Sam</td>
</tr>
</tbody>
</table>

It is observed that the non-atomic nature of fuzzy data makes it extremely difficult to represent them directly in a RDBS. However, fuzzy data can be represented directly and easily in the OODBs, as an object encapsulating its attributes and methods. Here, attributes consist of storage for fuzzy data while methods specified relevant operations such as intersection, union and linguistic modifier. Data encapsulation makes application programs development easier and more natural. In other words, application programmers can now
program, by a series of high level operations on fuzzy data instead of dealing with the low-level operations on fuzzy data in the program itself. In additions, should changes to the fuzzy data methods be required, these changes would be confined within the fuzzy data object. One the other hand, under the RDBS approach, the duplications of fuzzy data methods will render changes more difficult and intractable.

7.0 CONCLUSION

Based on the foregoing discussions, the advantages of the OODBs over the RDBS or the Extended RDBS in handling fuzzy data can be summarised as follows:
1. OODBs supports a richer set of data structure which is important for the direct representation and manipulation of fuzzy data;
2. OODBs supports data encapsulation allowing complex operations of fuzzy data to be encapsulated in the data themselves;
3. OODBs provides a greater modelling power;
4. OODBs reduces the problem of mismatch of host language as everything is object.

The great advantage enjoyed by OODBs clearly suggest that it is a better choice as a database framework over RDBS in implementing fuzzy database systems.

REFERENCES