Abstract

This paper deals with the conceptual design of FSM. FSM is a database model that supports fuzziness, uncertainty and impreciseness of real-world at attribute, class and class/class relationship levels. The paper also addresses some implementation issues.

Keywords: Fuzzy database, Database design, Imperfect information, Database mapping.

1 Introduction

In database literature, there are several extensions of semantic and object-oriented database models to support fuzziness, uncertainty and impreciseness of real-world at attribute, class and class/class relationship levels. Most of them introduce fuzziness only at the attribute level and consider that entities are fully encapsulated into their classes, which means that they fully verify the properties of these classes. We enumerate also some proposals for extending object-oriented and semantic database models to support fuzziness, uncertainty and impreciseness of real-world at the class definition level [4, 6]. FSM (fuzzy semantic model) is a recently proposed database model [2] that supports fuzziness, uncertainty and impreciseness at attribute, class and class/class relationship levels. This paper deals with the conceptual design of FSM and addresses some implementation issues. More specifically, section 2 briefly introduces FSM. Section 3 details the conceptual design of FSM. Section 4 addresses some implementation issues. Section 5 discusses some related works. Section 6 concludes the paper.

2 Fuzzy Semantic Model

The space of entities $E$ is the set of all entities of the interest domain. A fuzzy entity $e$ in $E$ is a natural or artificial entity that one or several of its properties are fuzzy. A fuzzy class $K$ in $E$ is a collection of fuzzy entities: $K = \{(e, \mu_K(e)) : e \in E \land \mu_K(e) > 0\}$. $\mu_K$ is a membership function and $\mu_K(e) \in [0, 1]$ represents the degree of membership (d.o.m) of the fuzzy entity $e$ in fuzzy class $K$.

Fuzziness is induced whenever an entity verifies only (partially) some of its class properties. We denote by $X_K = \{p_1, p_2, ..., p_n\}$ the set of these properties for a given fuzzy class $K$. $X_K$ is called the extent of fuzzy class $K$. The extent properties may be derived from attributes and/or from common semantics. The degree to which each of the extent properties determines fuzzy class $K$ is not the same. To ensure this, we associate to each extent property $p_i$ a non-negative weight $w_i$ reflecting its importance in deciding whether or not an entity $e$ is a member of a given fuzzy class $K$. We also impose that $\sum_{i=1}^{n} w_i > 0$.

On the other hand, an entity may verify fully or partially the extent properties of a given fuzzy class. Let $D^i$ be the basic domain of extent property $p_i$ values and $P^i$ is a subset of $D^i$, which represents the set of possible values of property $p_i$. The partial membership function of an extent property value is $\rho_{FK}$ which maps elements of $D^i$ into $[0, 1]$. 


For any attribute value \( v_i \in D^i \), \( \rho_{P_K}(v_i) = 0 \) means that fuzzy entity \( e \) violates property \( p_i \) and \( \rho_{P_K}(v_i) = 1 \) means that this entity verifies fully the property. The number \( v_i \) is the value of the attribute of entity \( e \) on which the property \( p_i \) is defined. For extent properties based on common semantics, \( v_i \) is a semantic phrase and the partial d.o.m \( \rho_{P_K}(v_i) \) is supposed to be equal to 1 but the user may explicitly provide a value less than 1. More generally, the value of \( \rho_{P_K}(v_i) \) represents the extent to which entity \( e \) verifies property \( p_i \) of fuzzy class \( K \). Thus, the global d.o.m of the fuzzy entity \( e \) in the fuzzy class \( K \) is:

\[
\mu_K(e) = \frac{\sum_{i=1}^{n} \rho_{P_K}(v_i) \cdot w_i}{\sum_{i=1}^{n} w_i}
\]

Each fuzzy class is uniquely identified with a name and has a list of characteristics or properties, called attributes. Some of these attributes are used to construct the extent set \( X_K \). To be a member of a fuzzy class \( K \), a fuzzy entity \( e \) must verify (fully or partially) at least one of the extent properties, i.e., \( \mu_K(e) > 0 \). Classes in FSM are categorized as exact or fuzzy. An exact class \( K \) is a class that all its members have a d.o.m equal to 1; i.e., \( \mu_K(e) = 1 \forall e \in K \). A fuzzy class \( K \) is a class that at least one of its members has a d.o.m strictly inferior to 1; i.e., \( \exists e \in K \) such that \( \mu_K(e) < 1 \).

The elements of a fuzzy class are called members. In FSM, \( \alpha \)-MEMBERS denotes for a given fuzzy class \( K \) the set \( \{ e : e \in K \land \mu_K(e) \geq \alpha \} \); where \( \alpha \in [0,1] \). It is easy to see that \( \alpha \)-MEMBERS \( \subseteq \beta \)-MEMBERS for all \( \alpha \) and \( \beta \) in \( [0,1] \) and verifying \( \alpha \geq \beta \). Note that 1-MEMBERS may also be refereed to true or exact members. In turn, \( \alpha \)-MEMBERS with \( 0 < \alpha < 1 \) are called fuzzy members. The concept of \( \alpha \)-MEMBERS may be mapped to the concept of \( \alpha \)-cut associated with fuzzy sets and which is defined for a fuzzy subset \( F \) as the set \( F_\alpha = \{ x : \mu_F(x) \geq \alpha \} \) with \( 0 \leq \alpha \leq 1 \).

FSM supports four different relationships: property, decision-rule, membering and interaction. The property relationships relate fuzzy classes to domain classes and each of which creates an attribute. The decision rule relationships are implementation of extent sets. The membering relationships relate fuzzy entities to fuzzy classes through the definition of d.o.m. The interaction relationships relate members of one fuzzy class to other members of one or several fuzzy classes.

FSM contains several complex fuzzy classes that permit to implement the semantics of real-world among objects in terms of generalization, specialization, aggregation, grouping and composition relationships, which are commonly used in purely semantic modelling. Finally, we mention that a full description of FSM can be found in [2].

### 3 Schema definition in FSM

This section provides a proposal for specifying schema of FSM-based databases. All examples of this section rely on the database example of Figure 1. In this example, GALAXY is an aggregate fuzzy class whose members are unique collection from fuzzy grouping class STARS. This last one is an homogenous collection of members from fuzzy class STAR. NOVA and SUPERNOVA are two attribute-defined fuzzy subclasses of STAR basing on type-of-star attribute.

In the generic definitions below, the following conventions are used: “[ ]”: optional parameter; “[ ]”: list of parameters or values; “|”: binary operator “xor”; “<>”: obligatory parameter; and “{ }”: series of parameters connected with the “xor”. The generic definition of a fuzzy class in FSM is as follows:

```
CLASS <class-name> WITH DOM OF <dom>
{
  SUPERCLASS:
  OF <class-name> WITH DOM OF <dom>
  …

INTERACTION CLASS OF <class-list>

EXTENT:
  <ext-pr> WITH WEIGHT OF <w_i> DECISION RULE IS ((<attr-name><op> (<attr-name>| <value>))<op><s-phrase>)
  …

ATTRIBUTES:
  <attr-name>: [FUZZY] DOMAIN <domaine>:
```

```
```
The SUPERCLASS component of the fuzzy class definition enumerates all the subclasses of the class along with their d.o.m relatively to this class. The INTERACTION CLASS OF component is for fuzzy interaction classes only. It permits to specify the list of the participant classes for which the interaction class is defined. The EXTENT part specifies the list of extent properties. For each extent property we indicate the name, the weight and the decision rule on which this extent property is based. As it is quoted earlier, decision rules may be attribute-based or semantic phrase-based. The left-side of the attribute-based decision rule indicates the attribute name on which the decision rule is based. The op operator may be a binary or a set-operator. The right-side of the attribute-based decision rule may be a crisp (e.g. age=21) or fuzzy (e.g. age=young) value. For semantic phrase-based decision rules, the op is an “is-a” operator and the right-side is a semantic phrase. For instance, we may have the following extent properties definitions:

\[ p_1 \text{ WITH WEIGHT OF 0.8 DECISION RULE IS } \text{luminosity} \geq 0.5L_s \]
\[ p_2 \text{ WITH WEIGHT OF 0.3 DECISION RULE IS } \text{weight} \text{ in } [0.01W_s, 1W_s] \]
\[ p_3 \text{ WITH WEIGHT OF 1.0 DECISION RULE IS is-a galaxy} \]

The symbol “\( W_s \)” above is the weight of the sun; it is often used as a measurement unit.

In the ATTRIBUTES component we specify the list of the attributes of the fuzzy class.

We note that attributes definition is inspired from [6]. This definition of attributes apply for both exact and fuzzy ones. An exact attribute requires the definition of a datatype (e.g. integer, string) and a domain as a range of possible values for the attribute. A fuzzy attribute requires the definition of a fuzzy type and a fuzzy domain. In addition, attributes may be specified as required, unique or multi-valued. For example, we may have the following declarations:

```
age: FUZZY DOMAIN (very old, old, young, very young):
TYPE OF integer WITH DOM OF 1.0
```

```
star-name: TYPE OF string WITH DOM OF 1.0: REQUIRED
```

Figure 1: Example database

The next component is specific for fuzzy composite and grouping classes. It permits to specify the members of fuzzy composite and/or grouping classes. The INTERACTION component indicates the eventual interaction relationship(s) of the fuzzy class. The following is an illustrative example.

```
CLASS star WITH DOM OF dom
{
    SUPERCLASS:
    OF supernova WITH DOM OF dom
    OF nova WITH dom
    EXTENT:
    p_4 WITH WEIGHT OF 0.8 DECISION RULE IS luminosity \geq 0.5L_s
    p_5 WITH WEIGHT OF 0.3 DECISION RULE IS weight \geq 0.05W_s
```

The following is an illustrative example.

```
CLASS star WITH DOM OF dom
{
    SUPERCLASS:
    OF supernova WITH DOM OF dom
    OF nova WITH dom
    EXTENT:
    p_4 WITH WEIGHT OF 0.8 DECISION RULE IS luminosity \geq 0.5L_s
    p_5 WITH WEIGHT OF 0.3 DECISION RULE IS weight \geq 0.05W_s
```

The next component is specific for fuzzy composite and grouping classes. It permits to specify the members of fuzzy composite and/or grouping classes. The INTERACTION component indicates the eventual interaction relationship(s) of the fuzzy class. The following is an illustrative example.
The generic definition of fuzzy subclasses is similar to that of fuzzy classes. In particular, they may have SUPERCLASS components that indicate the list of their own subclasses. In turn, they have a specific component, SPECIALIZATION, that is designed to map to their fuzzy superclasses. Its generic definition is as follows:

```
SUBCLASS <class-name> WITH DOM OF <dom>
{  
  SPECIALIZATION :
  OF <class-name> WITH DOM OF <dom> : 
  [BY ENUMERATION <members-list>]  
  [ON ATTRIBUTES <attr-list>]        
  [BY INTERSECTION WITH <class-list>]  
  [BY DIFFERENCE WITH <class-name>] 
  ...                                    
}
```

For each superclass, we indicate the name of the superclass and the d.o.m of the subclass in this superclass. We remark that in FSM, a subclass may be defined in four ways: simply by enumeration, basing on some selection attributes, through the set intersection operator, or through the set difference operator. The following is an example.

```
SUBCLASS supernova WITH DOM OF dom
{  
  SPECIALIZATION :
  OF star WITH DOM OF dom:
  ON ATTRIBUTES type-of-star 
  EXTENT:
  p:\text{\textit{p}}_{7} \text{WITH WEIGHT OF 0.5 DECISION RULE IS } \text{luminosity} \geq \text{high} 
}
```

4 Implementation issues

This section first shows how different kinds of imperfect information are represented and implemented. Then, it discusses the mapping of a FSM-based model to a fuzzy relational object (FRO) database model. Finally, shows how binary and set-operators should be extended to operate on imperfect information.

4.1 Representing imperfect information

FRO supports a rich set of imperfect data types. Here, we mention three ones (the complete list of these data types are available in [1]). First, it is important to mention that for facilitating data manipulation and for computing efficiency while giving the maximum flexibility to the users, the different types of attributes values in FRO are uniformly represented through possibility distribution. Each of these data types has one, two, three or four parameters permitting to generate its possibility distribution.

The graphical representation of possibility distribution of the fuzzy range data that handles the “more or less” information is provided in Figure 2a. For instance, we may have: “age = more or less between 20 and 30”. The d.o.m of any z in the fuzzy set $A$ on which the attribute is defined is computed through Equation (2):

$$
\mu_A(z) = \begin{cases} 
1, & \text{if } \beta \leq z \leq \gamma; \\
\frac{z - \beta}{\gamma - \beta}, & \text{if } \gamma < z < \lambda; \\
\frac{\lambda - z}{\lambda - \beta}, & \text{if } \alpha < z < \beta; \\
0, & \text{otherwise}. 
\end{cases} \tag{2}
$$
The parameters $\beta$ and $\gamma$ represent the support of the fuzzy set associated with the attribute values and $\alpha$ and $\lambda$ represent the limits of the transition zones.

The parameters required here are: the limits of the central range $a_1$ and $a_2$; and the left and right transition zones $b_1$ and $b_2$. Note that $a_1$ and $a_2$ are the crossover (or transition) points defined such that $\mu(a_1) = \mu(a_2) = 0.5$.

4.2 Implementing imperfect information

To store the specificity of all the attributes, we define a meta-relation, called ATTRIBUTES with the following attributes: (i) attr-id: it uniquely identifies each attribute; (ii) attr-name: it stores the name of the attribute; (iii) class-name: denotes the fuzzy class to which the attribute belongs; and (iv) data-type: which is a multi-valued attribute that stores the attribute type. For crisp attributes, this attribute works as in conventional databases (it may take the values of integer, float, etc.). For fuzzy attributes, the data-type attribute stores the fuzzy data type itself and the basic crisp data type on which the fuzzy data type is based. The ATTRIBUTES meta-relation associated with the model in Figure 1 is as follows:

<table>
<thead>
<tr>
<th>attr-id</th>
<th>attr-name</th>
<th>class-name</th>
<th>data-type</th>
</tr>
</thead>
<tbody>
<tr>
<td>attr-16</td>
<td>star-name</td>
<td>STAR</td>
<td>[string]</td>
</tr>
<tr>
<td>attr-17</td>
<td>type-star</td>
<td>STAR</td>
<td>[symbolic]</td>
</tr>
<tr>
<td>attr-18</td>
<td>age</td>
<td>STAR</td>
<td>[linguistic label, real]</td>
</tr>
<tr>
<td>attr-20</td>
<td>weight</td>
<td>STAR</td>
<td>[linguistic label, real]</td>
</tr>
</tbody>
</table>

The parameters associated with linguistic terms that appear in the domain of any linguistic data type are stored in meta-relation PARAMETERS. This meta-relation contains one line for each linguistic value. Its attributes are: (i) attr-id: which identifies the attribute; (ii) label: that stores a linguistic term belonging to the attribute domain; and (iii) parameters: which is a multi-valued attribute used to store the parameters required for generating the possibility distribution of the linguistic term. The following is an example:

<table>
<thead>
<tr>
<th>attr-id</th>
<th>label</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>attr-17</td>
<td>very young</td>
<td>{0.0, 0.1, 0.2, 0.3}</td>
</tr>
<tr>
<td>attr-17</td>
<td>young</td>
<td>{0.8, 0.9, 1.0, 1.1}</td>
</tr>
<tr>
<td>attr-17</td>
<td>old</td>
<td>{2.3, 5.0, 10.15}</td>
</tr>
<tr>
<td>attr-17</td>
<td>very old</td>
<td>{12.0, 17.5, 50.6}</td>
</tr>
</tbody>
</table>

The information required to define the extent properties of fuzzy classes are stored in two
meta-relations called A-DECISION-RULES and S-DECISION-RULES. A-DECISION-RULES is devoted to store attribute-based extent properties. It has the following attributes: (i) ext-property: stores the name of the extent property; (ii) c-name: denotes the name of the fuzzy class for which the extent property is defined; (iii) based-on: references the attr-id on which the extent property is based; (iv) decision-rule which is a composite attribute defined as follows: (a) operator: contains a binary or a set operator; and (b) operand: is a crisp or fuzzy value from the attribute domain; and (v) weight: stores the weight of the extent property. The following is an example:

<table>
<thead>
<tr>
<th>ext-property</th>
<th>c-name</th>
<th>based-on</th>
<th>decision-rule</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>ext-star-1</td>
<td>STAR</td>
<td>attr-18</td>
<td>[≤, 0.05L_s]</td>
<td>0.9</td>
</tr>
<tr>
<td>ext-star-2</td>
<td>STAR</td>
<td>attr-20</td>
<td>[≥, 0.5W_s]</td>
<td>0.3</td>
</tr>
</tbody>
</table>

The symbol “L_s” above is the luminosity of the sun; it is often used as a measurement unit. The S-DECISION-RULES is devoted to store extent properties based on common semantics. Its structure is similar to that of A-DECISION-RULE meta-relation but with no based-on attribute and the first component of the composite attribute decision-rule is an “is-a” operator.

At the extent definition of fuzzy classes, each attribute is mapped into a new composite with three component attributes: (i) attr-value: stores the value of the attribute as provided by the user; (ii) data-type: stores the data type of the value being inserted; and (iii) parameters: is a multi-valued attribute used to store parameters associated with the attribute value. The data-type attribute is used both at the extent definition and in the intent definition to allow users insert values of different data types, which may have different number of parameters. The mapping of the weight attribute of fuzzy class STAR is as follows:

<table>
<thead>
<tr>
<th>weight</th>
<th>attr-value</th>
<th>data-type</th>
<th>parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>{low, linguistic label H.I. (2.0..6)}</td>
<td>0.1L_s, real([nil])</td>
<td>0.3</td>
</tr>
<tr>
<td>0.3</td>
<td>{more than 10L_s, linguistic label (7.5L_s, 10)}</td>
<td>0.5L_s, real([nil])</td>
<td>0.75</td>
</tr>
</tbody>
</table>

Consider now the fuzzy subclass/superclass relationships transformation. A fuzzy subclass B of a fuzzy superclass A is mapped into a relation which inherits all the attributes of the relation transformed from A. In addition to the attribute dom, the relation B contains a new attribute, denoted by dom-A, which is used to store the d.o.m of one entity from fuzzy subclass B in its fuzzy superclass A. The same reasoning is used for their respective domains. Proximity relations are stored is meta-relation PROXIMITY. There are also several other meta-relations devoted to maintain information concerning interaction, subclass/superclass, composition, aggregation and grouping relationships. These meta-relations are needed to store the d.o.m of one fuzzy (sub)class in the corresponding fuzzy interaction, superclass, composite, aggregate or grouping class.

4.3 Mapping of a FSM-based model

As mentioned above, FSM-based model is mapped into a fuzzy relational object (FRO) database one. The FRO is being implemented as a front-ends of the relational object database system PostgreSQL. Here we provide the transformation of only simple classes and fuzzy subclass/superclass relationships. Each fuzzy class in the FSM model is mapped into a relation in the database level. The fuzzy attributes are mapped into composite ones as explained above. The crisp attributes are treated as in conventional databases. An additional non printable attribute, dom, used to store the global d.o.m is systematically added into the new relation. In addition, the information relative to the extent properties of the fuzzy class are automatically introduced in the A-DECISION-RULES and/or S-DECISION-RULES meta-relations. As example, the mapping of the fuzzy class STAR in Figure 1 is as follows:
fuzzy subclasses with more than one fuzzy superclass. Note particularly that the relation mapped from fuzzy class $B$ will contain several $dom-A$, one for each fuzzy superclass. For instance, the mapping of the fuzzy subclass SUPERNOVA in Figure 1 is as follows:

<table>
<thead>
<tr>
<th>snova-name</th>
<th>type-of-snova</th>
<th>$dom$</th>
<th>$dom$-star</th>
</tr>
</thead>
<tbody>
<tr>
<td>SN1987Va</td>
<td>Ib</td>
<td>0.95</td>
<td>1.0</td>
</tr>
<tr>
<td>SN1604</td>
<td>Ic</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>SN1006</td>
<td>Unknown</td>
<td>0.7</td>
<td>0.9</td>
</tr>
</tbody>
</table>

4.4 Computing the d.o.m

In FSM, an attribute-based extent property is associated with a condition of the form:

<left-hand-operand> <op> <right-hand-operand>

The left-side parameter indicates the attribute name on which the rule is based. The right-side parameter may be a crisp or fuzzy value. The parameter $op$ is a binary or a set operator. For instance, we may have the following decision rules: $luminosity \geq high$; and $age \in [17-21]$. These operators may be associated with the negation operator “not” denoted “$\neg$” below. Basing on the work of [7], we have extended all the operators that may be used in the definition of the extent properties of fuzzy classes and for query processing. The extended operators apply both for crisp and imprecise data. In this second case, Zadeh’s extension principle is used. Two examples are cited here.

The fuzzy “$\equiv$” and “$\approx$” operators. The fuzzy “$\equiv$” operator models the equality concept for precise as well as imprecise data values. Four cases can be distinguished:

$$
\mu_{\equiv}(\tilde{x}, \tilde{y}) = \begin{cases} 0, & |\tilde{x} - \tilde{y}| > m; \\ 1 - \frac{|\tilde{x} - \tilde{y}|}{m}, & |\tilde{x} - \tilde{y}| \leq m. 
\end{cases}
$$

The parameter $m$ represents the common support of fuzzy numbers $\tilde{x}$ and $\tilde{y}$. The fuzzy “$\sim \equiv$” operator is computed as the complement of “$\equiv$” operator, i.e. $\mu_{\sim \equiv} = 1 - \mu_{\equiv}(\tilde{x}, \tilde{y})$.

Finally, we mention that the proposed calculus delivers a index of possibility and the computed degrees are values of possibility obtained through zadeh’s extension principle. A degree of necessity can be also computed.

5 Related work

This section discusses some recent proposals of fuzzy database models. Based on fuzzy set and possibility distribution, the author in [5] introduces fuzziness in the different constructs of the semantic IFO data model, including printable type, abstract type, free type, grouping, aggregation, fragment and ISA relationship. The obtained system, denoted IF$2O$, is mapped into a relational fuzzy database schema. Its fruitful to remark that several fuzzy constructs of IF$2O$ (e.g. abstract types with the fuzziness at the schema level) can not be mapped into the relational fuzzy database schema.

Another proposal for extending the well-known ER data model to support fuzziness is reported in [3]. The obtained possibility-based F uzzy ER data model supports fuzziness and uncertainty at attribute, entity, relationships and instance/entity relationships. The paper includes a fuzzy entity-relationship methodology for the design and development of fuzzy relational databases. Note that fuzzy ER does not support fuzziness at subclass/superclass relationship level.

In [9], based on similarity relations, the IFO model was extended to the ExIFO (Extended IFO) to represent uncertainty as well as precise information. ExIFO supports uncertainty at the attribute, entity, and instance/class levels. The authors provide also an algorithm for mapping the schema of the ExIFO model to an extended NF$2$ database model.

Another extension of IFO model is provided
In this paper, the authors introduce proposals for handling ill-defined values. The uncertainty is supported at attribute, object and class levels. The paper includes also a mapping of IFO to a fuzzy object-oriented database model. However, the paper lacks a discussion of several important concepts such as object/class and subclass/superclass relationships.

Compared to these proposals, FSM is semantically richer and supports fuzziness within all of its constructs and within all levels. In addition, our mapping approach permits to gather advantages of both relational and object-oriented database models. Indeed, FRO: (i) supports a rich set of fuzzy, imprecise and uncertain data types ensuring its high flexibility; (ii) allows values on attributes to be of different data types which further enhances its flexibility; (iii) guarantees a high level of data/application independency thanks to its relational part; and (iv) supports all the constructs of FSM.

6 Conclusion

We have provided a proposal for specifying FSM schemas. We have also addressed some issues related to the implementation of FSM. More specifically, we have showed how different kinds of imperfect information are represented and internally implemented. Then we have briefly described a formal approach to map FSM-based model to a fuzzy relational object database model. Finally, we have given some insights concerning the extension of binary and set-operators to operate on imperfect information.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions, which improved the technical and the presentation of the paper.

References


