MAPPING OF CONCEPTUAL FUZZY OBJECT ORIENTED DATA
MODEL INTO LOGICAL SCHEMA

*S Shivam Singh, Kavita Agarwal and Jameel Ahmad
Department of Computer Science Integral University, Lucknow, India
*Author for Correspondence

ABSTRACT
With the growing need in organizations to store and manipulate complex data and relationship to meet this purpose. In this paper, we have discussed method of mapping of conceptual fuzzy object oriented data model into a logical schema so that in spite of the mapping of conceptual fuzzy object oriented data model into logical schema is yet to be thoroughly studied.

Keywords: Object Definition Language, Fuzzy Class, Fuzzy Object

INTRODUCTION
Conceptual modeling is an important phase in designing any database model. For designing the conceptual model of object-oriented database, unified modeling language (UML) has been extensively used because of its efficiency in specifying complex objects (Oscar et al., 1997) to satisfy the need of modeling complex objects with imprecision and ambiguity.

In my previous paper, a generic conceptual model has proposed for fuzzy object-oriented database which uses class diagram feature of UML, so that fuzzy objects can be modeled in the most efficient manner. In the continuation of that, we have tried to map conceptual fuzzy object oriented model into logical schema despite mapping of conceptual object oriented model into logical schemas has been extensively used so that it can be directly implemented using an object database management system (ODBMS).

In this paper, we have developed logical schema from conceptual fuzzy object oriented model with the help of object definition language. In section 2 we have proposed a new way of conceptual modeling of fuzzy Object-Oriented databases (Ma and Yan, 2010) with the help of UML. In section 3, we have discussed how to map conceptual fuzzy object oriented model into logical schema. Section 4 presents the conclusion of the proposed work.

Conceptual Modeling of Fuzzy Object Oriented Databases
- Fuzzy Object and Classes
Due to the lack of information, an object is fuzzy. More general, objects which have at least one attribute whose value is a fuzzy set are fuzzy objects. A class may be fuzzy because of the following reasons. Firstly some objects are fuzzy, which have similar properties. These objects belong to the class with the membership degree of [0,1]. Secondly, when a class is intentionally defined. The domain of an attribute may be fuzzy and a fuzzy class is created. For example, a class Old Syllabus is a set of fuzzy values such as long, very long, and about 30 year. Thirdly the sub class produced by a fuzzy class of specialization and the super class produced by some classes (in which at least one class is fuzzy) by means of generalization is also fuzzy.

![Example of UML fuzzy class diagram with two classes](image)

In the Figure 1, membership function is attached to the student and course which denotes the fuzzy classes. Here the attribute year in the student class is fuzzy because of set of fuzzy values {first, second, third}.

© Copyright 2014 | Centre for Info Bio Technology (CIBTech)
third, fourth). In the same way, attribute credit_hrs of course class is also a fuzzy attribute. Thus we can say that a fuzzy class provides a templates or schema for its instances.

Unlike fuzzy class diagram, in fuzzy object diagram there are only two rectangular boxes with dotted lines. The name of object and its class is underlined and shown in top box and an object attribute and their fuzzy values are shown in second compartment as shown in Figure 2.

![Figure 2: Fuzzy object diagram with two instances](image)

A function or a service that is provided by all instances of a fuzzy class is called fuzzy operation. The fuzzy operation thus provides an external interface of a fuzzy class. The interface presents outside view of fuzzy class without showing its internal structure or how its fuzzy operation are implemented. This technique of hiding the internal implementation detail of a fuzzy object from its external view is known as encapsulation or information hiding.

- **Association**

An association is a named relationship between or among instances of fuzzy classes. It can be of unary, binary and ternary nature. We have shown the association relation of degree two (binary) among instances of fuzzy classes in Figure 3. The binary relationship could be any of the following types.

![Figure 3: Association of fuzzy classes](image)

- **Generalization**

On generalizing a set of fuzzy classes into a more general fuzzy class we abstract not only the common attribute (i.e. may be fuzzy) and relationship, but the common operation as well the operation also may be...
Research Article

fuzzy. For example, consider a fuzzy class employee $E_f$. Here $E_f$ represents possibility of fuzzy membership for employee and there can be three type of employee i.e. hourly employee (HE), salaried employee (SE) and consultant (S). And $HE_f$, $SE_f$ and $S_f$ have the same meaning like $E_f$. The feature that will be shared by all employee is emp_name, emp_number, address, age, date hired and print_label are stored in fuzzy employee superclass. Following depicts the generalization.

Figure 5: Example of fuzzy generalization, inheritance and constraints

Here in the Figure 5, attribute will be inherited from the superclass that attribute may be of the fuzzy like age= \{young, middle, old\}. Generalization and inheritance are transitive across any number of levels of superclasses and subclass hierarchy.

Mapping of Conceptual Fuzzy Object Oriented Model into Logical Schema

Object Definition Language

Object definition language (ODL) allow us to specify a logical schema (Jeffery et al.,). It is independent specification programming language for defining OODB schema.

Defining a Class

In ODL, a class is represented using the class keyword and attribute using the attribute keyword here we have defined the student class and course class assuming that there no membership function associated with it i.e. non fuzzy class for the sake of simplicity.

class Student{
    attribute string name;
    attribute date dateOfBirth;
    attribute string address;
    attribute string phone;
    // plus relationship and operations…
};
class Course{
    attribute string crse_code;
    attribute date crse_title;
    attribute short credit_hrs;
    // plus relationship and credit_hrs and operations….
};

Here we have denoted ODL keyword in bold italic and we have added comment preceded by the “//” sign just to indicate we can add relationships and operations.

Defining an Attribute

An attribute value may be either a literal or an object identifier (Jeffery et al.,). It can also have fuzzy value if it have fuzzy value the class will be fuzzy. But here I have focused mainly on literal values which do not have identifier, it seems as constant. These are embedded inside object. literal values can be of...
atomic literal(string,char(character),float(real),etc.), collection literal (set,bag,list,array,etc.), structured literal(consists of a fixed no. of named elements).

Defining User Structures
ODL defines standard Data type structure, it can be defined by using keyword struct. For example:

```plaintext
Struct Address{
    string street_address;
    string city;
    string state;
    string zip;
};
```

Suppose if a student have more than one phone number the phone attribute can be defined by

```plaintext
attribute set < phone> phones;
```

Defining Operations
In ODL operation can be represented by using parenthesis after its name. Result of operation may be of fuzzy type suppose using age(); operation we can calculate the age and result may denote {child,young,old}.so we can say age can be act as fuzzy operation.ODL definition for student:

```plaintext
class Student{
    attribute string name;
    attribute date dateOfBirth;
    // user-defined structured attributes
    attribute string Address address;
    attribute string Phone phone;
    // plus relationship
    // operations
    short age();
    float gpa();
    boolian register_for(string crse,short sec,string term);
};
```

Defining a Range for an Attribute
If all the possible values that an attribute can have is known we can enumerate those values in ODL. For example if we know that the maximum number of sections for a course is 3, we can use the keyword enum before the attribute name for example:

```plaintext
class CourseOffering{
    attribute string term;
    attribute enum section{1,2,3,4};
    short enrollment();
};
```

Defining Relationships
Object database management group (ODMG) object model (Cattell et al., 2000) allows only unary and binary relationship in Figure 4 we have given name of relation for both direction. We use keyword relationship to specify a relation.

```plaintext
class Student{
    attribute string name;
    attribute date dateOfBirth;
    // user-defined structured attributes
    attribute string Address address;
    attribute string Phone phone;
    // many-to-many relationship between student and course offering
    relationship set <courseOffering> takes inverse courseOffering::taken_by;
    // operations
```
short age();
float gpa();
boolean register_for(string crse, short sec, string term);
};
Here the name of relationship is preceded by the target class in this case name of relation is takes and target class is courseOffering and the relationship is many to many because number of students can take multiple courseOffering. We have also used the keyword “set” to tell that student object is related to a set of courseOffering objects. In ODL the inverse key word is used to specify the relationship in reverse direction. Here the inverse of take is taken_by.

Class Extent
In ODL, we have used keyword extent, Extent of a class is the set of all instances of the class within the database (Cattell et al., 2000). For example, the extent students refer to all the student instances in database.

class Student{
    (extent students)
    attribute string name;
    attribute date dateOfBirth;
    attribute string Address address;
    attribute string Phone phone;
    relationship set <courseOffering> takes inverse courseOffering::taken_by;
    short age();
    float gpa();
    boolean register_for(string crse, short sec, string term);
};
class CourseOffering{
    (extent courseOffering)
    attribute string term;
    attribute enum section{1,2,3,4};
    relationship set <student> taken_by inverse student::takes;
    relationship course belongs_to inverse course::offers;
    short enrollment();
};
class Course{
    (extent courses)
    attribute string crse_code;
    attribute date crse_title;
    attribute short credit_hrs;
    relationship set <course> has_prereqs inverse course::is_prereq_for;
    relationship set <course> is_prereq_for inverse course::has_prereqs;
    relationship list < courseOffering > offers inverse courseOffering::belongs_to;
    short enrollment();
};
ODL Schema for University Database

Defining Generalization
As we know ODL support unary and binary association relationship but not of higher degree. It allow you to represent generalization relationship using the extend keyword. If we consider non fuzzy data in Figure 5(E-employee, HE-hourly employee, SE-salaried employee) then the ODL schema corresponding to class diagram is as follow:
class Employee{
    (extent employees)

Research Article

attribute short emp_number;
attribute string name;
attribute date dateOfBirth;
attribute string Phone phone;
attribute string Address address;
attribute float salary;
attribute date date_hired;
void print_label();
};
class HourlyEmploye extends Employee{
    extent hrly_emps
attribute float hourly_rate;
float wages();
};
class SalariedEmployee extends Employee{
    extent salaried_emps
attribute float annual_salary;
attribute Boolean stock_option;
void contribute_pension();
};
class consultant extends Employee{
    extent consultants
attribute short contract_number;
attribute float billing_rate;
float fees();
};

Defining an Abstract Class
ODL does not currently support the abstract keyword for classes and operations.

CONCLUSION AND FUTURE SCOPE
In this paper we have discussed how mapping will be done from conceptual fuzzy object oriented model into logical schema.
Over the past thirty years many researchers have proposed fuzzy data modeling for relational database and object oriented database with the help of ER model, EER model, IFO model, Ex-IFO model and Ex-IFO2 model yet there is very little research done in modeling fuzziness at conceptual data level in fuzzy object-oriented database with the help of UML So the future scope of our work will be to implementing this model to create a fuzzy database and retrieval of fuzzy information from these type of databases with the help of fuzzy object query languages (FOQL).

REFERENCES