Chapter -6

Object oriented implementation of Method used
for Normalization of Intuitionistic Fuzzy

Database into IF2NF & IF3NF
6.1 Introduction to Object Oriented Implementation

In this chapter we present an object oriented implementation of method proposed for normalization of an intuitionistic fuzzy relational database into IF2NF and IF3NF. Object oriented implementation of IF1NF is already given by Alam, sharfuddin, Roy and biswas [40-41]. They have used java as their implementation language.

We have also used object oriented language java to implement our proposed algorithm for IF2NF and IF3NF. Java is a well known programming language with important various features like Polymorphism, Inheritance, Portability, Encapsulation etc.

In this chapter we are introducing different functions and classes that we have used in coding. A rough representation of the classes and functions used for coding would look like:

```java
public class input_values
{
    input_values()
    {
```
I

code to accept values

}

display_values()
{
   // code to display values entered

}

public class key extends input_values
{

keyInput()
{
   // Enter Key Type: “1” for composite and “2” for non-composite
   //or single
   //code to accept key from user

}

keyCheck()
{
   // code to check validity of entered key, if valid proceed further,
   //else display message to enter another key and proceed to
   //keyInput() function again.

}

keyDetailsDisplay()
{
   // Gives the type of key i.e. composite or single
   //Give degrees of the key
   //Gives the validity of key
   // if key is non-composite then relation is in IF2NF
   // if key is composite then proceed to class IF2NF to check partial
   //dependency

}
public class IF2NF extends key
{

hesitationValue()
{
    // calculate hesitation if composite key
}

hesitationSig()
{
    // Test the significance of hesitation. if hesitation is insignificant
    // then follow fuzzy approach partial_F_DepnCheck () else proceeds
    // with partial_IF_DepCheck()  
}

partial_F_DepnCheck()
{
    // code to check the partial dependency using fuzzy approach
}

partial_IF_DepCheck()
{
    // check the partial dependency using Intuitionistic Fuzzy approach
}

decomposeRelation()
{
    // this function is called to decompose the relation if partial
    // dependency exist
}

resultIF2NF()
{
    // this function display relations in IF2NF
}
}
public class IF3NF extends IF2NF
{

hesitationValueIF3NF()
{
    // calculate hesitation of the each transitive relations that may occur
    // and further proceed to hesitationSigIF3N function
}

hesitationSigIF3NF()
{
    // Test the significance of hesitation. if hesitation is insignificant
    // then follow fuzzy approach transitive_F_DepnCheckIF3NF() else
    // proceeds with transitive_IF_DepCheckIF3NF()
}

transitive_F_DepnCheckIF3NF()
{
    // code to check the transitivity using fuzzy approach
}

transitive_IF_DepCheckIF3NF()
{
    // check the transitivity Intuitionistic Fuzzy approach
}

decomposeRelationIF3NF()
{
    // this function is called to decompose the relation if transitivity
    // exists
}

resultIF3NF()
{
    // this function display relations in IF3NF
}
In this program we provide IFFD with degree of membership and non membership. Based on the degrees provided, we will check the partial dependency in case of IF2NF and Transitivity in IF3NF. This program will help us to know up to what extends a particular relation belongs to Intuitionistic fuzzy second normal or third normal form and up to what extend it does not belong. We have taken same example that we have used to explain normalization into IF2NF. We have used JCreator LE for developing and checking our code. Here we give brief introduction of functions used in coding.

**public class input_values**: This class is responsible for taking input from user.

**input_values()**: This function will take attributes names from user in the form of dependent and non dependent attributes along with their membership and non membership values.

**display_values()**: This function will display the input attributes with their membership and nonmembership values in a matrix form.

**public class key extends input_values**: This class is responsible for taking key from the user and checking its validity.
**keyInput()**: This function will take prime key name of the relation from user.

**keyCheck()**: This function will check whether the key entered by user can be a permissible prime key for the relation or not.

**keyDetailsDisplay()**: This function will display “key is valid”, if a correct key is entered and “Not a valid key”, if the key entered by user can not be a prime key. It will also display membership and non membership degree of a valid key.

```java
public class IF2NF extends key
```

**hesitationValue()**: This function will calculate the hesitation in a relation.

**hesitationSig()**: This function will check significance of hesitation between relation of the attributes. If the hesitation is insignificant we will call partial_F_DepnCheck() function to check the partial dependency i.e. follow fuzzy approach to normalize the relation. If the hesitation is significant we will follow intuitionistic fuzzy approach i.e. we will call partial_IF_DepCheck() function.
partial_F_DepnCheck() : It will check fuzzy partial dependency.

partial_IF_DepCheck() : It will check intuitionistic fuzzy partial dependency.

decomposeRelation() : This function will divide the relation into smaller relation if the partial dependency exists.

resultIF2NF() : This function will display new relations with their key values.

public class IF3NF extends IF2NF : This class is responsible for checking a relation in 3NF and this class will also inherit the IF2NF and input_values class properties.

hesitationValueIF3NF() : This function will calculate the hesitation in a relation.

hesitationSigIF3NF() : This function will check significance of hesitation between relation of the attributes. If the hesitation is insignificant we will call transitive_F_DepnCheck() function to check the fuzzy transitive dependency i.e. follow fuzzy approach to normalize the relation. If the hesitation is significant we will follow intuitionistic fuzzy approach i.e. we will call transitive_IF_DepCheck() function.
transitive_F_DepnCheckIF3NF() : It will check fuzzy transitive dependency.

transitive_IF_DepCheckIF3NF() : It will check Intuitionistic fuzzy transitive dependency.

decomposeRelationIF3NF() : This function will divide the relation into smaller relation if the transitivity exists.

resultIF3NF() : This function will display new relations with their key values.

4.3 Sample output screen for checking IF2NF

```
Enter all attributes: Press XX to stop-
A
B
C
E
XX

Enter the non-dependent attributes:
A
B
C
AB
XX

Enter dependent attribute for “A”
C
E
XX

Enter dependent attribute for “B”
C
E
XX
```
Enter dependent attribute for “C”
E
XX
Enter dependent attribute for “AB”
C
E
XX

Enter membership and non-membership value for “A to C” separated by space 0.3 0.4
Enter membership and non-membership value for “A to E” separated by space 0.3 0.5
Enter membership and non-membership value for “B to C” separated by space 0.6 0.2
Enter membership and non-membership value for “B to E” separated by space 0.5 0.4
Enter membership and non-membership value for “C to E” separated by space 0.5 0.1
Enter membership and non-membership value for “AB to C” separated by space 0.8 0.1
Enter membership and non-membership value for “AB to E” separated by space 0.7 0.1

Displaying Input values:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1,0</td>
<td>0,0</td>
<td>.3,.4</td>
<td>.3,.5</td>
</tr>
<tr>
<td>B</td>
<td>0,0</td>
<td>1,1</td>
<td>.6,.2</td>
<td>.5,.4</td>
</tr>
<tr>
<td>C</td>
<td>0,0</td>
<td>0,0</td>
<td>1,1</td>
<td>.5,.1</td>
</tr>
<tr>
<td>AB</td>
<td>1,1</td>
<td>1,1</td>
<td>.8,.1</td>
<td>.7,.1</td>
</tr>
</tbody>
</table>

Enter 1 for single key and 2 for composite key:
2

Enter key value
AB
Checking key validity please wait . . . .
Entered key
“AB” is Valid
“AB” is 0.7,0.1 degree key of the relation
“AB” is a composite
6.3 Conclusion

In the previous chapters we have studied methods to normalize Intuitionistic fuzzy relational database into 2NF and 3NF using intuitionistic fuzzy functional dependencies, pq keys and other related concepts. In this chapter we have presented an object oriented implementations of the approach define for normalization of IF database into IF2NF and IF3NF. We have also given a brief introduction of the functions and classes that we have used in coding.
CONCLUSION & FUTURE SCOPE
Databases are one form of modeling parts of the real world. They may contain information related to enterprise, scientific activities, graphical entities, geographical information system or other fields. World of database is the world of digital computers, one of the most unexpected and exploratory system. It is therefore, not surprising that the type of storage is not always crisp rather fuzzy or intuitionistic fuzzy is nature. So to deal with imprecise data Fuzzy relational databases system (FRDBS) are introduced. FRDBS are the generalization of classical database model presented by E. F. Codd by allowing uncertain and imprecise information to be represented and manipulated.

Later on in some situation it was noticed that fuzzy databases are not sufficient to deal with deterministic part of a relation so the concept of Intuitionistic fuzzy database came into existence that can efficiently deal with deterministic part.

In the starting chapter of this thesis we have studied basic concept about fuzzy sets and intuitionistic fuzzy sets. Further we have also investigated Fuzzy and Intuitionistic fuzzy databases.
Similar to classical database, the problems of data redundancy and update anomalies do occur, if fuzzy/intuitionistic fuzzy databases are not properly designed. As our main objective is to reduce redundancy from fuzzy database we emphasized on normalization theory.

In this thesis we have presented a method to normalize an intuitionistic fuzzy relation into second normal form. This method will check the requirements for a relation to be in IF2NF if the relation fulfills the requirement it is said to be IF2NF with some degree of membership and non membership. Higher the degree of membership means a relation is strongly in IF2NF and less redundant as compare with lesser degree of membership. IF2NF mainly check Intuitionistic fuzzy partial dependency in a relation. If partial dependency exists between attributes then that relation can not be in IF2NF and further decomposition of that relation is required to remove intuitionistic fuzzy partial dependency.

As we know that the normal forms of relational database theory provide criteria for determining a table's degree of vulnerability to redundancies and anomalies. The higher the normal form applicable to a table, the less vulnerable it is to redundancies and anomalies. Each table has a "highest
normal form" (HNF): by definition, a table always meets the requirements of its HNF and of all normal forms lower than its HNF.

So to make our relation more redundancy free we further proposed a method to normalize an intuitionistic fuzzy relation into third normal form i.e. a higher normal form. And if a table in third normal form (3NF), then it is consequently in second normal form (2NF) as well; but the reverse is not necessarily the case. This method will check the requirement for a relation to be IF3NF if the relation fulfills the requirement it is said to be IF3NF with some degree of membership and non membership. Higher the degree of membership means a relation is strongly in IF3NF and less redundant as compare with lesser degree of membership. IF3NF mainly check Intuitionistic fuzzy Transitive dependency in a relation. If Transitive dependency exists between attributes then that relation can not be in IF3NF and further decomposition of that relation is required to remove intuitionistic fuzzy transitive dependency.
At the end of the thesis we have also given the object oriented implementation of method proposed for Intuitionistic fuzzy database normalization.

**Future Scope**

As we have already mentioned that higher normal forms successively impose higher degrees of strictness to reduce further redundancies and inconsistencies from the relation. So we feel that there is a good scope for further normalizing the Intuitionistic fuzzy database into BCNF, fourth and fifth normal form.