APPENDIX 1

TABLE DETAILS

Mainly three tables namely Teacher, Student and Class for small database of a school are used. The snapshots of all three tables are shown below.

Details of Class table are shown below. The primary key of this table is ClassID. The other dependent attributes are Class_Name, Strength, Section and Incharge. The field Incharge is the foreign key for this table in the Teacher table.
Table below shows the data contained in the Teacher table. The primary key of this table is TeacherID. The other dependent attributes are First_Name, Last_Name, Address, Phone, Age, Sex and Salary. Fuzzy conditions have been implemented on the fields Age and Salary using membership functions described in section 5.1. Flexible queries can be implemented on these fields. The field TeacherID is the primary key of the table Teacher and it is foreign key corresponding to the Incharge field of Class table.
Table below shows the data contained in the Student table. The primary key of this table is Roll_No. The other dependent attributes are First_Name, Last_Name, Address, Phone, Age, Sex, Class, Height and Weight. Fuzzy conditions have been implemented on the fields Age, Height and Weight using membership functions. Flexible queries can be implemented on these fields. The field ClassID is the primary key of the table Class and it is foreign key corresponding to the Class field of Student table.
// Import Namespaces that needs reference within the program

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Configuration;

using System.Collections;

// Keeping all our code for Fuzzy SQL Generation under one namespace
namespace FuzzySQLGeneration
{

    // Create Class for our code
    public partial class FuzzyForm : Form
    {
    
        // Constructor to initialize
        public FuzzyForm()
        {
        }
    
}
InitializeComponent();

}  

// Form Load method of windows application. This will execute first
private void FuzzyForm_Load(object sender, EventArgs e)
{
    // try catch will handle default exceptions
    try
    {
        cmbTableNames.Items.Clear();
        cmbWhereField.Items.Clear();

        // Prepare SQL Connection to retrieve data from database server
        SqlConnection con = new SqlConnection();
        con.ConnectionString = ConfigurationSettings.AppSettings["connectionstring"].ToString();

        // SqlCommand cmd = new SqlCommand("Select name from sysobjects
        where type = 'u'", con);

        SqlCommand cmd = new SqlCommand("Select distinct TableName from FuzzyFields", con);

        con.Open();
        SqlDataReader dr = cmd.ExecuteReader();

        // Initialize the values within the drop downs on form load
while (dr.Read())
{
    cmbTableNames.Items.Add(dr.GetString(0));
}

cmbTableNames.SelectedIndex = 0;
}
catch (Exception ex)
{
    string exception = ex.Message;
}

// choose different tables from the drop down. This method will handle the required changes
private void cmbTableNames_SelectedIndexChanged(object sender, EventArgs e)
{
    cmbWhereField.Items.Clear();
    lstFields.Items.Clear();

    string tableName = cmbTableNames.SelectedItem.ToString();

    // Prepare SQL Connection to retrieve data from database server
using (SqlConnection con = new SqlConnection())
{
    con.ConnectionString =
    ConfigurationSettings.AppSettings["connectionstring"].ToString();
    con.Open();

    using (SqlCommand cmd = new SqlCommand("SELECT FieldName FROM FuzzyFields WHERE TableName = "+ tableName + ",", con))
    {
       SqlDataReader dr = cmd.ExecuteReader();

       while (dr.Read())
       {
           cmbWhereField.Items.Add(dr.GetString(0));
       }
    con.Close();
    }  
}

con.Open();

using (SqlCommand cmd = new SqlCommand("SELECT column_name from INFORMATION_SCHEMA.COLUMNS where TABLE_NAME = "+ tableName + ",", con))
{
   SqlDataReader dr = cmd.ExecuteReader();
// Populate the Fields drop down for the selected table name

while (dr.Read())
{
    lstFields.Items.Add(dr.GetString(0));
}
con.Close();
}

cmbWhereField.SelectedIndex = 0;
}

// After the selections have been done. User will click on the button
// GenerateSQLFromFuzzy. This method will implement the algorithm to
determine fuzzy
// values and display them in the grid

private void btnGenerateSQLFromFuzzy_Click(object sender, EventArgs e)
{
    try
    {
        gvResults.Columns.Clear();

        StringBuilder sqlString = new StringBuilder();

        string tableName = cmbTableNames.SelectedItem.ToString();
string fieldNames = "*";

IEnumerator fieldsEnum;

// logic to choose the selected fields and also to keep the field with fuzzy
// applied at the last with column ‘Satisfying_Degree’
if (lstFields.SelectedItems.Count > 0)
{
    fieldsEnum = lstFields.SelectedItems.GetEnumerator();
}
else
{
    fieldsEnum = lstFields.Items.GetEnumerator();
}
fieldsEnum.Reset();

StringBuilder fieldsBuilder = new StringBuilder();

while (fieldsEnum.MoveNext())
{
    {
        cmbWhereField.SelectedItem.ToString().Trim())
    {
}
if (fieldsBuilder.Length > 0)
{
    // Will put ' , ' comma at the last of every field name
    fieldsBuilder.Append(" , ");
}

}
}

fieldNames = fieldsBuilder.ToString();

if (!fieldNames.Contains(cmbWhereField.SelectedItem.ToString()))
{
    fieldNames = fieldNames + " , " + cmbWhereField.SelectedItem.ToString();
}

sqlString.Append("Select " + fieldNames.Trim() + " From " + tableName.Trim() + " order by " + (string)cmbWhereField.SelectedItem);

if ((string)cmbWhereField.SelectedItem != "" &&
cmbWhereField.SelectedItem != null)
{
    sqlString = ApplyFuzzy(sqlString);
}
lblSQL.Text = sqlString.ToString();

    // Prepare SQLConnection and talk to database server
SqlConnection con = new SqlConnection();
con.ConnectionString =
    ConfigurationSettings.AppSettings["connectionstring"].ToString();
SqlCommand cmd = new SqlCommand(sqlString.ToString(), con);
    con.Open();

    using (SqlDataAdapter sqlAdapter = new SqlDataAdapter(cmd))
    {
        // Use DataAdapter to fill DataTable
        using (DataTable t = new DataTable())
        {
            sqlAdapter.Fill(t);

            // Render data onto the screen on Datagrid
            gvResults.DataSource = t;
        }
    }
    
    catch (Exception ex)
    {
    
    }
string message = ex.Message;

} }

// This method will take SQL and then apply fuzzy algorithm and return the changed SQL

private StringBuilder ApplyFuzzy(StringBuilder FuzzySQL)
{
    int LowS = 0, HighS = 0, LowValue = 0, HighValue = 0, IsFirst = 0, IsLast = 0;

    // Prepare SQL Connection and retrieve values from database server
    using (SqlConnection con = new SqlConnection())
    {
        con.ConnectionString = ConfigurationSettings.AppSettings["connectionstring"].ToString();

        SqlCommand cmd = new SqlCommand("select fromRange, toRange, FromCmplSatisDegree,
ToCmplSatisDegree ,IsFirst, IsLast FROM FieldValues INNER JOIN FuzzyFields ON
FieldValues.FuzzyFieldID = FuzzyFields.FuzzyFieldID WHERE
(FuzzyFields.FieldName = "")
+ (string)cmbWhereField.SelectedItem + " and TableName = " +
(string)cmbTableNames.SelectedItem + " and FieldValueName = " +
(string)cmbFuzzyValues.SelectedItem + " )", con);

con.Open();
SqlDataReader dr = cmd.ExecuteReader();
while (dr.Read())
{
    LowValue = dr.GetInt32(0);
    HighValue = dr.GetInt32(1);
    LowS = dr.GetInt32(2);
    HighS = dr.GetInt32(3);
    IsFirst = dr.GetInt32(4);
    IsLast = dr.GetInt32(5);
}
}

string calculatedString = "";
string fieldName = ((string)cmbWhereField.SelectedItem).Trim();

// This is the code to implement algorithm logic

calculatedString = ",cast(case when (" + IsFirst + " = 1) then case when (" +
fieldName + "<=" + HighS + ") then 1.0 when ((" +

108
fieldName + "">" + HighS + ") and (" + fieldName + "\leq" + HighValue + ")
then ((" + HighValue + "," + fieldName + ")*1.0)/((" + HighValue + "," + HighS + ")*1.0) when (" +
fieldName + "">" + HighValue + ") then 0.0 end when (" +
IsFirst + "=!1) and (" + IsLast + "=!1)) THEN case when (" + fieldName + 
"\leq" + LowValue + ") then 0.0 when (" +
fieldName + "">" + LowValue + ") and (" + fieldName + "\leq" + LowS + ") then ((" +
fieldName + "," + LowValue + ")*1.0)/((" + LowS + "," + LowValue + ")*1.0) when (" +
fieldName + "," + HighS + ") then 1.0 when (" +
fieldName + "">" + HighS + ") and (" + fieldName + "\leq" + HighValue + ")
then ((" + HighValue + "," + fieldName + ")*1.0)/((" + HighValue + "," + HighS + ")*1.0) when (" +
fieldName + "">" + LowS + ") then 1.0 when (" +
fieldName + "">" + LowS + ") and (" + fieldName + "\leq" + HighValue + ")
then 0.0 end when (" +
IsLast + "=1) then case when (" + fieldName + "\leq" + LowValue + ")
then 0.0 when (" +
fieldName + "">" + LowValue + ") and (" + fieldName + "\leq" + LowS + ") then ((" +
fieldName + "," + LowValue + ")*1.0)/((" + LowS + "," + LowValue + ")*1.0) when ("+
fieldName + "," + LowS + ") then 1.0 end end as decimal(3,2)) as 'Satisfying_Degree' ;
public void SetParentTable(string parentTable)
{
    FuzzySQL.Insert(FuzzySQL.ToString().IndexOf("From"), calculatedString);

    return FuzzySQL;
}

// Method to implement code when one changes the where field
private void cmbWhereField_SelectedIndexChanged(object sender, EventArgs e)
{
    cmbFuzzyValues.Items.Clear();

    SqlConnection con = new SqlConnection();

    con.ConnectionString =
            ConfigurationSettings.AppSettings["connectionstring"].ToString();

    SqlCommand cmd = new
            SqlCommand("SELECT FieldValues.FieldValueName FROM FieldValues INNER JOIN FuzzyFields ON FieldValues.FuzzyFieldID = FuzzyFields.FuzzyFieldID where TableName = ", con);

    con.Open();
    SqlDataReader dr = cmd.ExecuteReader();

    // Populate the fuzzy fields drop down with fuzzy titles available
    }
while (dr.Read())
{
    cmbFuzzyValues.Items.Add(dr.GetString(0));
}

cmbFuzzyValues.SelectedIndex = 0;
}

// On the fuzzy titles changed, populate the respective range values
private void cmbFuzzyValues_SelectedIndexChanged(object sender, EventArgs e)
{
    txtFromRange.Text = "";
    txtToRange.Text = "";

    SqlConnection con = new SqlConnection();
    con.ConnectionString =
        ConfigurationSettings.AppSettings["connectionstring"].ToString();

    SqlCommand cmd = new
        SqlCommand("SELECT FieldValues.FromRange, FieldValues.ToRange,
            FieldValues.Metrics, FieldValues.FromCmplSatisDegree,
            FieldValues.ToCmplSatisDegree FROM FieldValues INNER JOIN FuzzyFields ON
            FieldValues.FuzzyFieldID = FuzzyFields.FuzzyFieldID where TableName = "");
+ (string)cmbTableNames.SelectedItem + " and FieldName = " +
(string)cmbWhereField.SelectedItem
+ " and FieldValues.FieldValueName = " +
(string)cmbFuzzyValues.SelectedItem + "", con);

    con.Open();
    SqlDataReader dr = cmd.ExecuteReader();

    while (dr.Read())
    {
        txtFromRange.Text = Convert.ToString(dr.GetInt32(0));
        txtToRange.Text = Convert.ToString(dr.GetInt32(1));
        lblFromMetrics.Text = lblToMetrics.Text =
        lblFromMetricsCompSatisfy.Text = lblToMetricsCompSatisfy.Text = dr.GetString(2);
        txtCompleteSatisfyFromRange.Text = Convert.ToString(dr.GetInt32(3));
        txtCompleteSatisfyToRange.Text = Convert.ToString(dr.GetInt32(4));
    }

    // This method will show the generated SQL details in a message box
    private void btnShowSQLDetails_Click(object sender, EventArgs e)
    {
        MessageBox.Show(lblSQL.Text, "SQL Details");
    }
}
APPENDIX 3
SCREEN SHOTS

Result of selecting Table – Student, Condition Field = Age, Fuzzy Value = Toddlers
Result of selecting Table – Student, Condition Field = Height, Fuzzy Value = Medium

```
Select First_Name, Age, Weight, Height, cast(case when (Height <= 135) then 1.0 when (Height > 135) and (Height <= 155) then ((155-Height)*1.0)/((155-135)*1.0) when (Height > 135) then 0.0 end when (01=1) and (01=1)) THEN case when (Height <= 80) then 0.0 when ((Height > 80) and (Height <= 100)) then ((Height-80)*1.0)/((100-80)*1.0) when ((Height > 100) and (Height <= 135) then 1.0 when ((Height > 135) and (Height <= 155)) then ((155-Height)*1.0)/((155-135)*1.0) when (Height > 155) then 0.0 end when (01=1) then case when (Height <= 80) then 0.0 when ((Height > 80) and (Height <= 100)) then ((Height-80)*1.0)/((100-80)*1.0) when (Height > 100) then 1.0 end as Satisfying_Degree From Student order by Height
```
Result of selecting Table – Student, Condition Field = Weight, Fuzzy Value = Heavy

Select: First_Name, Age, Height, Weight, cast(case when (0 = 1) then case when (Weight<=$9999) then 1.0 when (Weight>$9999) and (Weight<=$9999) then (09999-Weight)*1.0) when (Weight>$9999) then 0.0 end when ((0=1) and (1=1)) THEN case when (Weight<=$55) then 1.0 when ((Weight>$55) and (Weight<=$65)) then ((Weight-$55)*1.0)/(65-$55)*1.0 when (Weight>$65) and (Weight<=$9999) then 1.0 when ((Weight>$9999) and (Weight<=$9999)) then 0.0 end when (1=1) THEN case when (Weight<=$55) then 0.0 when ((Weight>$55) and (Weight<=$65)) then ((Weight-$55)*1.0)/(65-$55)*1.0 when (Weight>$65) then 1.0 end as Satisfying_Degree' from Student order by Weight

OK
Result of selecting Table–Teacher, Condition Field = Age, Fuzzy Value = Middle Age

```
Select First_Name, Last_Name, Salary, Age ,cast(case when (0 = 1) then case when (Age<=50) then 1.0 when ((Age>=50) and (Age<=50)) then (100-Age)*1.0)/((50-50)*1.0) when (Age>=50) then 0.0 end when ((0=1) and (0=1)) THEN case when (Age<=30) then 0.0 when ((Age>=30) and (Age<=38)) then ((Age-30)*1.0)/((38-30)*1.0) when ((Age<38) and (Age>=30)) then 1.0 when ((Age<50) and (Age<=50)) then ((50-50)*1.0)/((50-50)*1.0) when (Age>50) then 0.0 end when (0=1) then case when (Age<=30) then 0.0 when ((Age>30) and (Age<=38)) then ((Age-30)*1.0)/((38-30)*1.0) when (Age>38) then 1.0 end and when (0=1) then end as decimal(3,2) as 'Satisfying_Degree' From Teacher order by Age
```

OK
Result of selecting Table – Student, Condition Field = Age, Fuzzy Value = Teenagers

SQL Details

Select First_Name, Height, Weight, Age, cast(case when (0 = 1) then case when (Age<=16) then 1.0 when ((Age>16) and (Age<=20)) then ((20-Age)*1.0)/((20-15)*1.0) when (Age>20) then 0.0 end when (0)=1 and (0)=1) THEN case when (Age<=11) then 3.0 when ((Age>11) and (Age<=14)) then ((Age-11)*1.0)/((14-11)*1.0) when ((Age>14) and (Age<=16)) then 1.0 when ((Age>16) and (Age<=20)) then ((20-Age)*1.0)/(20-16)*1.0) when (Age>20) then 0.0 end when (0)=1 then case when (Age<=11) then 0.0 when ((Age>11) and (Age<=14)) then ((Age-11)*1.0)/(14-11)*1.0) when (Age>14) then 0.0 end and as `Satisfying_Degree` from Student order by Age

OK