Fig 5.2: Detail Listing of Attributes of the Database

Next to that comes the decision about the blocking field. Two blocking algorithms are implemented i.e. Full Index and Blocking Index [4]. Full indexing does no blocking rather it compares every record with other records in the database. The drawback of full indexing is that as it compares every record with other, the number of comparisons becomes \( n^* (n-1) \) where ‘\( n \)’ is a number of records in a database. If ‘\( n \)’ is large i.e. in range of tens of thousands, i.e. 60000, then comparisons performed will be 60000*59999 which is around 359994000 which is much large and may take several minutes to calculate the result on higher range machines. It should also be considered that the data in a mid range organizations is much large than that is considered in the example and it may become a difficult task for the organizations those who can not afford to buy higher end servers. In order to tackle the problem, generally blocking indexes are used. According to Blocking Indexing, user has to select a field/s and corresponding encoding function/s. The data in the selected field/s is/are encoded according to the encoding function/s selected. The field/s is/are selected such that maximum comparisons are possible in between the records. For example, if “state” is the field which has less number of blanks in it, then selection of the field will have proper groupings as states are generally entered properly. A parameter stating length of encoding string is to be entered by the user.

Here we proposed to use the WordNet ontology and shortcut ontologies developed by us to be used. An option is given here on the GUI for Ontology [5] use. Here one thing is essentially to be remembered is that the user should select the fields that are prone to synonym and/or where shortcuts can be used. The caution is taken considering that the when an ontology searched using binary search, the comparisons required will be maximum as they does not find any match there in the dictionary. In order to get the records in the
same block after checking it for ontology and shortcuts, we calculated the encoding of each synonym after arranging them in ascending order. This ensures the same encode combination for synonyms. For example, in one record the value of the field is say ‘God’ and in other it is ‘divinity’. As both the words means the same thing, their synonyms will be same. Each time the synonyms are extracted, they will be sorted in ascending order. After sorting the synonyms, the list of synonyms will be same in both the cases and hence the encodings. A snapshot giving the design of this GUI is shown in Fig 5.3.

![Fig 5.3: Snapshot Showing Index Selection Page](image)

Blocking of the data creates the groups of the records. The records so grouped will be compared in the groups. If blocking fields have blank or null data then the records with blocking field blank will be compared to all other records irrespective of groups. Now it is a proper time to compare the fields in the record. Snapshot of field matching interface is shown below in Fig 5.4. Use can select as many fields for comparison. Depending on the algorithm selected numbers of parameters are to be entered. Common parameters are Missing Weight, Agree Weight, Disagree Weight and Threshold. Missing Weight is required to be greater than Disagree Weight. This will ensure the record with missing values to put in possible matches in every block and the record will matched based on other fields selected for record matching. Disagree weight is required to be less than the agree weight and threshold should be either equal to or less than agree weight and greater than disagree weight. Several algorithms for field comparison are implemented as discussed in the previous chapters. A user has to select one from the given list for a field. As many fields as user wants can be selected with field matching techniques. After selecting a field and field matching algorithm, the required parameters are to be filled which are different for different algorithms. The user should select the proper matching algorithm for the fields. The GUI is
built such that editing the data in the list boxes showing the fields and their corresponding field matching algorithms and other data are editable. Hence user can select and deselect the list as per the requirement. The selection for ontology support (both WordNet and shortcut) is made available here so that depending on the requirement of the user ontology support is selected.

Fig. 5.4: A Snapshot of Field Comparison Page

Fig 5.5: Debugging View of the Code
After processing each field for matching with other, a list of match score is generated for each field processed for the match in each record pair. This vector of comparison weights is stored in the structure named as RWWeight which contains a list of fields compared in the record set (FIndex1 and FIndex2) which are vectors, identifier of the record which is compared (RID1 and RID2) and the vector (WeightV) storing the weights after application of the comparison algorithm. The structure and its contents are shown above using debugging view in Fig 5.5. The weights so calculated can be used to compare the records for identification as duplicates. Algorithms that are used are Fellegi Sunter Classification, Optimal Threshold Classification and K Means algorithm. The parameters required for classification are minimum threshold and maximum threshold. The record pair with weight less than minimum threshold are identified as non match, the record pair with weight between minimum threshold and maximum threshold are identified as possible matches and record pairs with weight greater than maximum threshold are identified as matches. Fig 5.6 shown below shows the GUI designed for selection of classification algorithm and the requisite parameters. Clicking the process button will start the calculation according to the algorithm designed.

![Classification Page](image)

Fig 5.6: Classification Page

Shown below in Fig 5.7, is a view of the table showing intermediate structure, after application of blocking algorithm and at the time of comparison of fields. It can be seen that two fields are introduced in the table. They are MyRI and my_str_index. MyRI is a field which assures unique identification of the record if my_str_index is not defined in the recordset. And the last field shown in the list associates the primary key is not defined in the block for comparison. Comparison will be done between the records in the block either having same block index or block index is empty.
5.2 SAMPLE CODE

5.2.1 A CODE FOR THESAURUS

Public m_thes As New NHunspell.MyThes("th_en_us_v2.idx", "th_en_us_v2.dat")
Module thesaurus
' A module to get the synonym for the word in the recordset
Public Function getData(ByRef str As String) As String()
    Dim sc As String
    Dim nh As NHunspell.ThesResult = m_thes.Lookup(LCase(str))
    Dim nm As NHunspell.ThesMeaning
    Dim lst As New List(Of String)
    Try
        For Each nm In nh.Meanings
            Dim s As List(Of String) = nm.Synonyms
            For Each sc In s
                If sc.IndexOf(""") > -1 Then
                    sc = sc.Substring(0, sc.IndexOf(""""))
                End If
                If lst.IndexOf(sc) < 0 Then
                    lst.Add(LCase(sc))
                End If
            Next
        Next
        Catch ex As Exception
            End Try
    End Function
End Module

5.2.2 A CODE FOR INDEXING

Private Sub bunifuProcessIndex_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles bunifuProcessIndex.Click
    Dim dff As Double = 0.0
    Me.Cursor = Cursors.WaitCursor
    Dim il As Integer
    Dim cmd As OleDb.OleDbCommand = myConn.CreateCommand
    Dim sTime, endTime As Double

common.v_indexType = Me.cmbIndexSelect.Text
If v_indexType.Equals("FullIndex") Then
    common.v_indexField = Nothing
    common.v_indexEncode = Nothing
    common.v_indexmaxLen = Nothing
Else
    ReDim common.v_indexField(Me.lstIndexField.Items.Count - 1)
    ReDim common.v_indexEncode(Me.lstIndexEncode.Items.Count - 1)
    ReDim common.v_indexmaxLen(Me.lstIndexMaxLen.Items.Count - 1)
For il = 0 To Me.lstIndexField.Items.Count - 1
    common.v_indexField(il) = Me.lstIndexField.Items(il)
    common.v_indexEncode(il) = Me.lstIndexEncode.Items(il)
    common.v_indexmaxLen(il) = Me.lstIndexMaxLen.Items(il)
Next
'Process the indexing
If v_indexType.Equals("BlockingIndex") Then
    Dim ddr As Odbc.OdbcDataReader
    Try
        myCom.CommandText = "drop table tempindex"
        myCom.ExecuteNonQuery()
        Catch ex As Exception
            'tempindex is code generted table. It may not be in the database. Hence no need to catch exception
        End Try
    'copy the fields from blocking into tempindex table
    Dim cmdstr As String = "create table tempindex (MYRI integer,"
    Dim cmdstr1 As String = "insert into tempindex select MYRI,"
    For il = 0 To common.v_indexField.Length - 1
        cmdstr += common.v_indexField(il) + " text,"
        cmdstr1 += common.v_indexField(il) + ","
    Next
    cmdstr = cmdstr.Substring(0, cmdstr.Length - 1) + ")"
    myCom.CommandText = cmdstr
    myCom.ExecuteNonQuery()
    cmdstr1 = cmdstr1.Substring(0, cmdstr1.Length - 1) + " from temp"
    myCom.CommandText = cmdstr1
    myCom.ExecuteNonQuery()
    Try
        myCom.CommandText = "Alter table tempindex add my_str_index text"
        myCom.ExecuteNonQuery()
        Catch ex As Exception
            Debug.Print(ex.Message)
        End Try
    myCom.CommandText = "select * from tempindex"
    ddr = myCom.ExecuteReader
    Dim enc, encmb As String
    Dim en As New encode
    enc = ""
    Dim thld As Integer
    Dim xid As Integer = 0
    If Me.chkIndexOnto.Checked Then
        t_a.thesaurus = True
        stTime = Now().TimeOfDay.TotalMilliseconds
    While ddr.Read
        xid = xid + 1
        Me.Text = Str(xid)
encmb ="
For il = 0 To common.v_indexField.Length - 1
  enc = dr.Item(common.v_indexField(il))
  Dim oList() As String = getData(LCase(enc))
  If enc.Length > 0 Then
    If common.v_indexEncode(il).Equals("Double Metaphone") Then
      enc = en.dmetaphone(enc, CInt(common.v_indexMaxLen(il)))
      For thld = 0 To oList.Length - 1
        enc = enc + "," + en.dmetaphone(oList(thld), CInt(common.v_indexMaxLen(il)))
      Next
    ElseIf common.v_indexEncode(il).Equals("Fuzzy Soundex") Then
      enc = en.fuzzysoundex(enc, CInt(common.v_indexMaxLen(il)))
      For thld = 0 To oList.Length - 1
        enc = enc + "," + en.fuzzysoundex(oList(thld), CInt(common.v_indexMaxLen(il)))
      Next
    ElseIf common.v_indexEncode(il).Equals("NYSIIS") Then
      enc = en.nysis(enc, CInt(common.v_indexMaxLen(il)))
      For thld = 0 To oList.Length - 1
        enc = enc + "," + en.nysis(oList(thld), CInt(common.v_indexMaxLen(il)))
      Next
    ElseIf common.v_indexEncode(il).Equals("Phonex") Then
      enc = en.phonex(enc, CInt(common.v_indexMaxLen(il)))
      For thld = 0 To oList.Length - 1
        enc = enc + "," + en.phonex(oList(thld), CInt(common.v_indexMaxLen(il)))
      Next
    ElseIf common.v_indexEncode(il).Equals("Phonix") Then
      enc = en.phonix(enc, CInt(common.v_indexMaxLen(il)))
      For thld = 0 To oList.Length - 1
        enc = enc + "," + en.phonix(oList(thld), CInt(common.v_indexMaxLen(il)))
      Next
    ElseIf common.v_indexEncode(il).Equals("Soundex") Then
      enc = en.soundex(enc, CInt(common.v_indexMaxLen(il)))
      For thld = 0 To oList.Length - 1
        enc = enc + "," + en.soundex(oList(thld), CInt(common.v_indexMaxLen(il)))
      Next
    ElseIf common.v_indexEncode(il).Equals("Modified Soundex") Then
      enc = en.mod_soundex(enc, CInt(common.v_indexMaxLen(il)))
      For thld = 0 To oList.Length - 1
        enc = enc + "," + en.mod_soundex(oList(thld), CInt(common.v_indexMaxLen(il)))
      Next
    ElseIf common.v_indexEncode(il).Equals("None") Then
      enc = enc
      For thld = 0 To oList.Length - 1
        enc = enc + "," + oList(thld)
      Next
  End If
  encmb += enc
Next
' code for sequencing for theraus
If encmb.Length > 0 Then
  Dim encmbAt() As String = encmb.Split("",").ToCharArray(),
  System.StringSplitOptions.RemoveEmptyEntries()
End If
Dim encmbA1(encmbA1.Length) As String
Dim enci, exx As Integer
exx = 0
For enci = 0 To encmbA1.Length - 1
    If Array.IndexOf(encmbA1, encmbA1(enci)) < 0 Then
        encmbA1(exx) = encmbA1(enci)
        exx += 1
    End If
Next
ReDim Preserve encmbA1(exx - 1)
Array.Sort(encmbA1)
encmb = ""
For enci = 0 To exx - 1
    encmb += encmbA1(enci) + ","
Next
encmb = encmb.Substring(0, encmb.Length - 1)
End If
comm.CommandText = "update temp set my_str_index='" + encmb.Replace(" ", ",") + ","
where MYRI='" + CStr(drr.Item(0))
comm.ExecuteNonQuery()
End While
endTime = Now().TimeOfDay.TotalMilliseconds
MsgBox(Str(endTime - stTime))
Else
    t_a.thesaurus = False
    stTime = Now().TimeOfDay.TotalMilliseconds
    While drr.Read
        xid = xid + 1
        Me.Text = Str(xid)
        encmb = ""
        For il = 0 To common.v_indexField.Length - 1
            enc = drr.Item(common.v_indexField(il))
            If enc.Length > 0 Then
                If common.v_indexEncod(il).Equals("Double Metaphone") Then
                    enc = dmetaphone(enc, CInt(common.v_indexmaxLen(il)))
                ElseIf common.v_indexEncod(il).Equals("Fuzzy Soundex") Then
                    enc = en.fuzzysoundex(enc, CInt(common.v_indexmaxLen(il)))
                ElseIf common.v_indexEncod(il).Equals("NYSIS") Then
                    enc = en.nysiis(enc, CInt(common.v_indexmaxLen(il)))
                ElseIf common.v_indexEncod(il).Equals("Phonex") Then
                    enc = en.phonex(enc, CInt(common.v_indexmaxLen(il)))
                ElseIf common.v_indexEncod(il).Equals("Phonix") Then
                    enc = en.phonix(enc, CInt(common.v_indexmaxLen(il)))
                ElseIf common.v_indexEncod(il).Equals("Soundex") Then
                    enc = en.soundex(enc, CInt(common.v_indexmaxLen(il)))
                ElseIf common.v_indexEncod(il).Equals("Modified Soundex") Then
                    enc = en.mod_soundex(enc, CInt(common.v_indexmaxLen(il)))
                ElseIf common.v_indexEncod(il).Equals("None") Then
                    enc = enc
                End If
            End If
        Next il
        comm.CommandText = "update temp set my_str_index='" + enc
End If
End If
    enc = enc.Replace("", ")
    comm.CommandText = "+update tempindex set " + common.v_indexField(il) + "=" + enc + "+ where MYRI=" + CStr(drr.Item(0))
    comm.ExecuteNonQuery()
End If
enccmb += enc
Next
    comm.CommandText = "+update temp set my_str_index=" + enccmb.Replace(" ", ") + " where MYRI=" + CStr(drr.Item(0))
    comm.ExecuteNonQuery()
End While
endTime = Now().TimeOfDay.TotalMilliseconds
MsgBox(Str(endTime - stTime))

End If
Debug.Print("over")
End If

End If
Me.Cursor = Cursors.Default
End Sub

5.2.3 A CODE FOR FIELD MATCHING

Public Sub ProcessCompare()
    Me.Cursor = Cursors.WaitCursor
    Dim fl As Integer = Me.lstCompareFieldName.Items.Count
    Dim il, i As Integer
    Dim alg As String
    Dim sc As New StrCmpar
    Dim dt As New DateTime
    Dim fields(1 - 1) As String
    Dim drr, drr1 As Odbc.OdbcDataReader
    Dim com1, com As Odbc.OdbcCommand
    Dim cmdstr As String = "select MyRI,"
    For i = 0 To fields.Length - 1
        fields(i) = Me.lstCompareFieldName.Items(i)
        cmdstr += Trim(fields(i)) + ","
    Next
    com1 = New Odbc.OdbcCommand()
    com1.Connection = myConn
    cmdstr += " my_str_index from temp"
    com1.CommandText = cmdstr
    drr = com1.ExecuteReader
    While drr.Read 'get the the field
        com = New Odbc.OdbcCommand()
        com.Connection = myConn
        com.CommandText = cmdstr + " where my_str_index=" + Trim(drr.Item("my_str_index")) + ""
        drr1 = com.ExecuteReader()
        While drr1.Read
            If drr(0) <> drr1(0) Then
                Dim r1 As RWWeight
                r1.RID1 = drr(0)
                r1.RID2 = drr1(0)
                ReDim r1.Findex1(fl - 1)
                ReDim r1.Findex2(fl - 1)
                ReDim r1.WeightV(fl - 1)
                For il = 0 To fl - 1

r1.FlIndex1(i) = il
r1.FlIndex2(il) = il
alg = Me.lstCompareAlgorithm.Items(il)
If Me.chkOntoSCuts.checked And Also Module1.checkOntoSCut(drr(fields(il)),
drr1(fields(il)), True) Then
r1.WeightV(il) = CDb(Me.lstCompareAgree.Items(il))
ElseIf alg.Equals("Bag Distance") Then
   r1.WeightV(il) = sc.BagDistance(drr(fields(il)), drr1(fields(il)),
   CDb(Me.lstCompareThreshold.Items(il)), CDb(Me.lstCompareAgree.Items(il)),
   CDb(Me.lstCompareDisAgree.Items(il)), CDb(Me.lstCompareMissing.Items(il)))
ElseIf alg.Equals("Character Histogram") Then
   r1.WeightV(il) = sc.ComparatorCharHistogram(drr(fields(il)), drr1(fields(il)),
   CDb(Me.lstCompareThreshold.Items(il)), CDb(Me.lstCompareAgree.Items(il)),
   CDb(Me.lstCompareDisAgree.Items(il)), CDb(Me.lstCompareMissing.Items(il)))
ElseIf alg.Equals("Contains String") Then
   r1.WeightV(il) = sc.ComparatorContainsString(drr(fields(il)), drr1(fields(il)),
   CDb(Me.lstCompareMissing.Items(il)), CDb(Me.lstCompareAgree.Items(il)),
   CDb(Me.lstCompareDisAgree.Items(il)))
ElseIf alg.Equals("Editext") Then
   r1.WeightV(il) = sc.ComparatorEditext(drr(fields(il)), drr1(fields(il)),
   CDb(Me.lstCompareThreshold.Items(il)), CDb(Me.lstCompareAgree.Items(il)),
   CDb(Me.lstCompareDisAgree.Items(il)), CDb(Me.lstCompareMissing.Items(il)))
ElseIf alg.Equals("Encode String") Then
   r1.WeightV(il) = sc.ComparatorEncodeString(drr(fields(il)), drr1(fields(il)), "Soundex", 
   CInt(Me.lstCompareCharLen.Items(il)), CDb(Me.lstCompareAgree.Items(il)),
   CDb(Me.lstCompareDisAgree.Items(il)), CDb(Me.lstCompareMissing.Items(il)))
ElseIf alg.Equals("Exact String") Then
   r1.WeightV(il) = sc.ComparatorExactString(drr(fields(il)), drr1(fields(il)),
   CDb(Me.lstCompareMissing.Items(il)))
ElseIf alg.Equals("Key Difference") Then
   r1.WeightV(il) = sc.ComparatorKeyDiff(drr(fields(il)), drr1(fields(il)),
   CInt(Me.lstCompareCharLen.Items(il)), CDb(Me.lstCompareAgree.Items(il)),
   CDb(Me.lstCompareDisAgree.Items(il)), CDb(Me.lstCompareMissing.Items(il)))
ElseIf alg.Equals("Numeric Absolute") Then
   r1.WeightV(il) = sc.ComparatorNumericAbs(drr(fields(il)), drr1(fields(il)),
   CDb(Me.lstCompareCharLen.Items(il)), CDb(Me.lstCompareAgree.Items(il)),
   CDb(Me.lstCompareDisAgree.Items(il)), CDb(Me.lstCompareMissing.Items(il)))
ElseIf alg.Equals("Numeric Percentage") Then
   r1.WeightV(il) = sc.ComparatorNumericPerc(drr(fields(il)), drr1(fields(il)),
   CDb(Me.lstCompareCharLen.Items(il)), CDb(Me.lstCompareAgree.Items(il)),
   CDb(Me.lstCompareDisAgree.Items(il)), CDb(Me.lstCompareMissing.Items(il)))
ElseIf alg.Equals("Ontology LCS") Then
   Dim sd As String
   sd = Me.lstCompareCharLen.Items(il)
   Dim sind As Integer = sd.IndexOf("\"",")
   Dim cl As Integer = CInt(sd.Substring(0, sind))
   sd = sd.Substring(sind + 1, sd.Length - (sind + 1))
   r1.WeightV(il) = sc.ComparatorOnToLCS(drr(fields(il)), drr1(fields(il)),
   CDb(Me.lstCompareThreshold.Items(il)), sd, cl, CDb(Me.lstCompareAgree.Items(il)),
   CDb(Me.lstCompareDisAgree.Items(il)), CDb(Me.lstCompareMissing.Items(il)))
ElseIf alg.Equals("Truncate String") Then
   r1.WeightV(il) = sc.ComparatorTruncateString(drr(fields(il)), drr1(fields(il)),
   CInt(Me.lstCompareCharLen.Items(il)), CDb(Me.lstCompareMissing.Items(il)))
ElseIf alg.Equals("Two Level Jaro") Then
   r1.WeightV(il) = sc.ComparatorTwoLevelJaro(drr(fields(il)), drr1(fields(il)),
   CDb(Me.lstCompareThreshold.Items(il)), CDb(Me.lstCompareAgree.Items(il)),
   CDb(Me.lstCompareDisAgree.Items(il)), CDb(Me.lstCompareMissing.Items(il)), "Jaro")
ElseIf alg.Equals("Da Le Distance") Then
r1.WeightV(il) = sc.DaLeDistance(drr(fields(il)), drr1(fields(il)),
CDbl(Me.lstCompareThreshold.Items(il)), CDbl(Me.lstCompareAgree.Items(il)),
CDbl(Me.lstCompareDisAgree.Items(il)), CDbl(Me.lstCompareMissing.Items(il))
ElseIf alg.Equals("Edit Distance") Then
r1.WeightV(il) = sc.EditDistance(drr(fields(il)), drr1(fields(il)),
CDbl(Me.lstCompareThreshold.Items(il)), CDbl(Me.lstCompareAgree.Items(il)),
CDbl(Me.lstCompareDisAgree.Items(il)), CDbl(Me.lstCompareMissing.Items(il))
ElseIf alg.Equals("Jaro") Then
r1.WeightV(il) = sc.jaro(drr(fields(il)), drr1(fields(il)),
CDbl(Me.lstCompareThreshold.Items(il)), CDbl(Me.lstCompareAgree.Items(il)),
CDbl(Me.lstCompareDisAgree.Items(il)), CDbl(Me.lstCompareMissing.Items(il))
ElseIf alg.Equals("Jaro Approximate MultiWord") Then
r1.WeightV(il) = sc.JaroApproxMultiWord(drr(fields(il)), drr1(fields(il)), False,
CDbl(Me.lstCompareThreshold.Items(il)), CDbl(Me.lstCompareAgree.Items(il)),
CDbl(Me.lstCompareDisAgree.Items(il)), CDbl(Me.lstCompareMissing.Items(il))
ElseIf alg.Equals("Positional Q-Gram") Then
Dim pd As Boolean = False
Dim cd As String
cd = Me.lstCompareCharLen.Items(il)
If Clnt(cd(0).ToString) = 2 Then
    pd = True
End If
    cd = cd.Substring(2)
r1.WeightV(il) = sc.PosQGram(drr(fields(il)), drr1(fields(il)),
CDbl(Me.lstCompareThreshold.Items(il)), pd, cd, CDbl(Me.lstCompareAgree.Items(il)),
CDbl(Me.lstCompareDisAgree.Items(il)), CDbl(Me.lstCompareMissing.Items(il))
ElseIf alg.Equals("Q-Gram") Then
Dim pd As Boolean = False
Dim cd As String
cd = Me.lstCompareCharLen.Items(il)
If Clnt(cd(0).ToString) = 2 Then
    pd = True
End If
    cd = cd.Substring(2)
r1.WeightV(il) = sc.QGram(drr(fields(il)), drr1(fields(il)),
CDbl(Me.lstCompareThreshold.Items(il)), Me.lstCompareAgree.Items(il),
CDbl(Me.lstCompareDisAgree.Items(il)), Me.lstCompareMissing.Items(il))
ElseIf alg.Equals("Smith Waterman Distance") Then
r1.WeightV(il) = sc.SWDistance(drr(fields(il)), drr1(fields(il)),
CDbl(Me.lstCompareThreshold.Items(il)), CDbl(Me.lstCompareAgree.Items(il)),
CDbl(Me.lstCompareDisAgree.Items(il)), CDbl(Me.lstCompareMissing.Items(il)),
Me.lstCompareCharLen.Items(il))
ElseIf alg.Equals("Syllable Alignment Distance") Then
r1.WeightV(il) = sc.SyllADistance(drr(fields(il)), drr1(fields(il)),
CDbl(Me.lstCompareThreshold.Items(il)), Me.lstCompareCharLen.Items(il),
CDbl(Me.lstCompareDisAgree.Items(il)), CDbl(Me.lstCompareMissing.Items(il))
ElseIf alg.Equals("Wrinkler") Then
r1.WeightV(il) = sc.wrinkler(drr(fields(il)), drr1(fields(il)),
CDbl(Me.lstCompareThreshold.Items(il)), CDbl(Me.lstCompareDisAgree.Items(il)),
CDbl(Me.lstCompareMissing.Items(il))
ElseIf alg.Equals("Age") Then
r1.WeightV(il) = dt.ComparatorAge(drr(fields(il)), drr1(fields(il)), 0.9, 0.2, "mm/dd/yyyy", 10)
ElseIf alg.Equals("Date") Then
r1.WeightV(il) = dt.ComparatorDate(drr(fields(il)), drr1(fields(il)), 0.9, 0.2, "mm/dd/yyyy", 20, 20)
ElseIf alg.Equals("Time") Then
r1.WeightV(il) = dt.ComparatorTime(drr(fields(il)), drr1(fields(il)), 0.9, 0.2, 10, 10)
End If
Next
REFERENCES:


