Option Compare Database

Option Explicit

Const GenericSetStatus = acSysCmdSetStatus

Const GenericClearStatus = acSysCmdClearStatus

Function CleanPathname(Pathname As Variant) As String

'

'This function verifies that a path specification begins with

'a valid drive specification, i.e. a letter followed by a colon.

'If the path specification does not begin with a valid drive

'specification, the string "InvalidDriveSpecification" is returned.

'If the path specification begins with a valid drive specification,

'all repeating backslashes are reduced to a single backslash, and the

'cleaned up path specification is returned.

'

Dim strPathname1, strPathname2 As Variant

Dim i As Long

Dim blnPreviousBackslash

strPathname2 = "InvalidDriveSpecification"

If Len(Pathname) > 1 Then

If InStr(1, "abcdefghijklmnopqrstuvwxyz", Left(Pathname, 1), vbTextCompare) And mid(Pathname, 2, 1) = ":" Then

'OK, we have a valid drive specification.

strPathname2 = Pathname

If Len(Pathname) > 2 Then

'Ensure that a backslash immediately follows the drive specification.

If mid(Pathname, 3, 1) <> "\" Then

strPathname1 = mid(Pathname, 1, 2) & "\" & mid(Pathname, 3)

Else

strPathname1 = Pathname

End If

'Reduce any repeating backslashes to a single backslash.

strPathname2 = ""

blnPreviousBackslash = False

For i = 1 To Len(strPathname1)

If Not (mid(strPathname1, i, 1) = "\" And blnPreviousBackslash) Then

strPathname2 = strPathname2 & mid(strPathname1, i, 1)

If mid(strPathname1, i, 1) = "\" Then

blnPreviousBackslash = True

Else

blnPreviousBackslash = False

End If

End If

Next i

End If

End If

End If

CleanPathname = strPathname2

End Function

Function SSURGODBPath() As String

'

'This function returns the directoy in which this database

'file resides. The returned path does not include the name

'of the current database. The returned path has no trailing

'backslash.

'

Dim dbsSSURGO As Database

Dim strPath As String

Dim intStartPos As Integer

Dim intBackSlashPos As Integer

Set dbsSSURGO = CurrentDb()

strPath = dbsSSURGO.Name

'Strip the database name and trailing backslash.

intStartPos = 1

intBackSlashPos = InStr(intStartPos, strPath, "\")

Do While (intBackSlashPos <> 0)

intStartPos = intBackSlashPos + 1

intBackSlashPos = InStr(intStartPos, strPath, "\")

Loop

SSURGODBPath = ""

If intStartPos <> 1 Then SSURGODBPath = Left(strPath, intStartPos - 2)

End Function

Function DropSSA()

'

'This function deletes the data corresponding to one or more soil survey

'areas that were selected by the user via the "Drop Soil Survey Area"

'form. This function has no input arguments and does not return any

'value to indicate success or failure.

'

'The keys of the soil survey areas selected by the user to be deleted

'are written to a file by the name of "SYSTEM - Soil Survey Area", by

'an event procedure of the "Drop Soil Survey Area" form. This function

'then reads that file to get the keys of the soil survey area or areas

'to be deleted.

'

Dim dbs As Database, rst1 As Recordset, rst2 As Recordset

Dim qdfTemp As QueryDef

Dim strSQL1 As String, strSQL2 As String, strSQL3 As String

Dim strMessage As String

Dim lngDistmdkey As Long

Dim lngLegendCount As Long

Dim strNewline As String

Dim rcode As Variant

strNewline = String(1, 13) & String(1, 10)

Set dbs = CurrentDb

'Open a cursor for the legends to be deleted.

strSQL1 = "Select [area symbol], [area name], [legend key] from [SYSTEM - Soil Survey Area]"

Set rst1 = dbs.OpenRecordset(strSQL1)

Do Until rst1.EOF

'For each legend to be deleted, do the following.

rcode = SysCmd(GenericSetStatus, "Dropping data for survey area: " & rst1![area symbol] & " - " & rst1![area name] & "...")

'Determine the primary key of the distribution metadata record associated with this legend.

strSQL2 = "Select [distmdkey] from [distlegendmd] where lkey = '" & rst1![legend key] & "'"

Set rst2 = dbs.OpenRecordset(strSQL2)

If rst2.RecordCount = 0 Then

rst2.Close

strMessage = "Unable to determine the primary key of the distribution metadata record associated with soil survey area: " \_

& strNewline & rst1![area symbol] & " - " & rst1![area name] & strNewline & strNewline & "One or more soil survey areas could not be dropped."

MsgBox strMessage, vbOKOnly + vbExclamation, "Drop Soil Survey Area - Error"

Exit Function

Else

lngDistmdkey = rst2![distmdkey]

rst2.Close

End If

'Determine if there is currently more than legend associated with this distribution metadata record.

strSQL2 = "Select count(\*) as LegendCount from distlegendmd where [distmdkey] = '" & lngDistmdkey & "'"

Set rst2 = dbs.OpenRecordset(strSQL2)

lngLegendCount = rst2![LegendCount]

rst2.Close

If lngLegendCount = 1 Then

'This legend is the only remaining legend for the corresponding distribution metadata record.

'Delete the corresponding distribution metadata record and all of its children. It is sufficient

'to delete the parent distribution metadata record because all relationships within the distribution

'metadata object are set to "delete/cascade".

Set qdfTemp = dbs.CreateQueryDef("", "Delete from [distmd] where distmdkey = '" & lngDistmdkey & "'")

qdfTemp.Execute

Else

'This legend is not the only remaining legend for the corresponding distribution metadata record.

'Just delete this legend from the legend distribution metadata table.

Set qdfTemp = dbs.CreateQueryDef("", "Delete from [distlegendmd] where lkey = '" & rst1![legend key] & "'")

qdfTemp.Execute

End If

'Delete this legend and all of its children. We use to just delete the legend record and let cascade delete

'rules take care of child tables, but we have since discovered that this approach exhausts all system resources

'for large soil survey areas. Accordingly, we now delete data for a survey in 4 steps. Step 1 - Delete everything

'at the chorizon level and below, for the appropriate survey area. Step 2 - Delete everything in cointerp, for the

'appropriate survey area. Step 3 - Delete everything at the component level and below, for the appropriate survey

'area. Step 4 - Delete everything at the legend level and below, for the appropriate survey area.

'This is arguably a little Mickey Mouse, but I didn't want to delete from each table individually. This approach

'does succeed on my PC, for the largest soil survey area, MT627, with a full set of interps, where loaded database

'size is approximately 388MB.

'Step 1 - Delete from chorizon and below.

strSQL3 = "DELETE chorizon.\*, legend.lkey "

strSQL3 = strSQL3 & "FROM legend INNER JOIN (mapunit INNER JOIN (component INNER JOIN chorizon ON component.cokey = chorizon.cokey) ON mapunit.mukey = component.mukey) ON legend.lkey = mapunit.lkey "

strSQL3 = strSQL3 & "WHERE (((legend.lkey)='" & rst1![legend key] & "'));"

Set qdfTemp = dbs.CreateQueryDef("", strSQL3)

qdfTemp.Execute

'Step 2 - Delete from cointerp.

strSQL3 = "DELETE cointerp.\*, legend.lkey "

strSQL3 = strSQL3 & "FROM legend INNER JOIN (mapunit INNER JOIN (component INNER JOIN cointerp ON component.cokey = cointerp.cokey) ON mapunit.mukey = component.mukey) ON legend.lkey = mapunit.lkey "

strSQL3 = strSQL3 & "WHERE (((legend.lkey)='" & rst1![legend key] & "'));"

Set qdfTemp = dbs.CreateQueryDef("", strSQL3)

qdfTemp.Execute

'Step 3 - Delete from component and below.

strSQL3 = "DELETE component.\*, legend.lkey "

strSQL3 = strSQL3 & "FROM legend INNER JOIN (mapunit INNER JOIN component ON mapunit.mukey = component.mukey) ON legend.lkey = mapunit.lkey "

strSQL3 = strSQL3 & "WHERE (((legend.lkey)='" & rst1![legend key] & "'));"

Set qdfTemp = dbs.CreateQueryDef("", strSQL3)

qdfTemp.Execute

'Step 4 - Delete from legend and below.

strSQL3 = "DELETE legend.\*, legend.lkey "

strSQL3 = strSQL3 & "FROM legend "

strSQL3 = strSQL3 & "WHERE (((legend.lkey)='" & rst1![legend key] & "'));"

Set qdfTemp = dbs.CreateQueryDef("", strSQL3)

qdfTemp.Execute

'The following step was added for SSURGO version 2.1, which includes additional tables.

'Step 5 - Delete from sacatalog and below.

'At the current time, the sacatalog hierarchy is independent of the legend hierarchy.

strSQL3 = "DELETE sacatalog.\*, sacatalog.areasymbol "

strSQL3 = strSQL3 & "FROM sacatalog "

strSQL3 = strSQL3 & "WHERE (((sacatalog.areasymbol)='" & rst1![area symbol] & "'));"

Set qdfTemp = dbs.CreateQueryDef("", strSQL3)

qdfTemp.Execute

'The following step was added for SSURGO version 2.1, which includes additional tables.

'Step 6 - Delete from featdesc.

'The featdesc table is not related to any other table in the database, at least via

'engine level referential integrity.

strSQL3 = "DELETE featdesc.\*, featdesc.areasymbol "

strSQL3 = strSQL3 & "FROM featdesc "

strSQL3 = strSQL3 & "WHERE (((featdesc.areasymbol)='" & rst1![area symbol] & "'));"

Set qdfTemp = dbs.CreateQueryDef("", strSQL3)

qdfTemp.Execute

rcode = SysCmd(GenericClearStatus)

rst1.MoveNext

Loop

'Reset map unit sort table to remove references to mapunits that have been deleted.

Set qdfTemp = dbs.CreateQueryDef("", "Delete from [SYSTEM - Mapunit Sort Specifications]")

qdfTemp.Execute

'Reset interp depth sort table to remove references to components that have been deleted.

Set qdfTemp = dbs.CreateQueryDef("", "Delete from [SYSTEM - Interp Depth Sequence]")

qdfTemp.Execute

'Reestablish map unit sort data for all remaining mapunits.

Call SortAllMapunits

'Reestablish interp depth sort data for all remaining components.

Call SortInterpsByDepth

End Function

Function PluralGeomorphicFeatureName(varGeomftname As Variant, varGeomfname As Variant) As Variant

'

'This function provides the plural form of a geomorphic feature name.

'

' Arguments:

'

' varGeomftname - The corresponding geomorphic feature type name.

' Needed because geomorphic feature names are only

' unique within their corresponding type.

'

' varGeomfname - The singular form of the geomorphic feature name.

'

' Returns:

'

' The plural form of geomfname, or Null, if geomfname is not a

' valid instance of type varGeomftname.

'

Dim varPluralname As Variant

Dim varGeomftiid As Variant

varPluralname = Null

varGeomftiid = DLookup("[geomftiid]", "[SYSTEM - Geomorphic Feature Type]", "[geomftname] = """ & varGeomftname & """")

If Not IsNull(varGeomftiid) Then varPluralname = DLookup("[geomfnamep]", "[SYSTEM - Geomorphic Feature]", "[geomftiidref] = " & varGeomftiid & " and [geomfname] = """ & varGeomfname & """")

PluralGeomorphicFeatureName = varPluralname

End Function

Function ReadQueryWriteFile()

Dim dbsSSURGO As Database

Dim ds As Recordset

Dim tv As Recordset

Dim qdfTemp As QueryDef

Set dbsSSURGO = DBEngine.Workspaces(0).Databases(0)

Set ds = dbsSSURGO.OpenRecordset("Input - Mapunit Sort Specifications", DB\_OPEN\_DYNASET)

Set qdfTemp = dbsSSURGO.CreateQueryDef("", "Delete from [Insert File Name Here]")

qdfTemp.Execute

Set tv = dbsSSURGO.OpenRecordset("SYSTEM - Mapunit Sort Specifications", DB\_OPEN\_TABLE)

Do Until ds.EOF

tv.AddNew

tv![Insert Field Name Here] = ds![Insert Field Name Here]

tv.Update

ds.MoveNext

Loop

ds.Close

tv.Close

ReadQueryWriteFile = 0

End Function