Cascade Record and Image Deletes

This tutorial is the third of a three part series on image delete options. Unlike the other two tutorials we will be not working with George Petrov's Pure ASP upload behavior on this page. Instead we will look at how to remove related records and the images associated with them when deleting a master record. By controlling orphaned records we keep the database tables clean and our applications run faster with fewer resources consumed. It only makes sense to remove images associated with the deleted records as well, again conserving server resources.

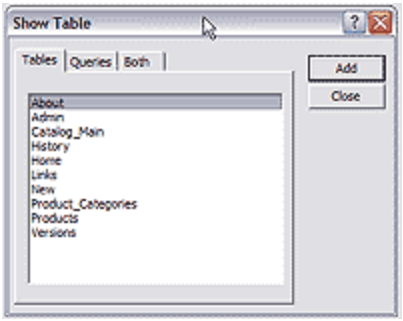
In this example we will look at deleting a category in a typical product catalog with Product_Categories, Products, and Versions. Having more than one table imposes difficulties that can only be overcome by a little hand coding and modification of stock UD behaviors. This tutorial details how to set up the relationships in an Access database for cascade deletes of related records* and how to implement server side code to delete images associated with related records.†

I will set up some conventions to make it easy to follow the tutorial. This tutorial assumes you have created a recordset navigation page to select the record you want to delete and a method to select the desired record. It assumes you have added a form to the delete page with a method to submit the delete. It assumes you have created an Access database with the described table structure or your customized version of it.

Color conventions will be used to make following the tutorial as simple as possible. Stock UltraDev code blocks that are not modified in this tutorial will be black. Tutorial code will be blue. Comments pertinent to the tutorial will be grey. We will begin with the Access database.

Relationships

The catalog database will need the following structure: Tables for Product_Categories, Products, and Versions, each of the tables having an image field so that as many as three images may be associated with any given Product_Category > Product > Version set. The tables are related by primary key values in the following manner: the Category_ID field relates the Products table to the Product_Category records. The Product_ID field relates the versions to products.



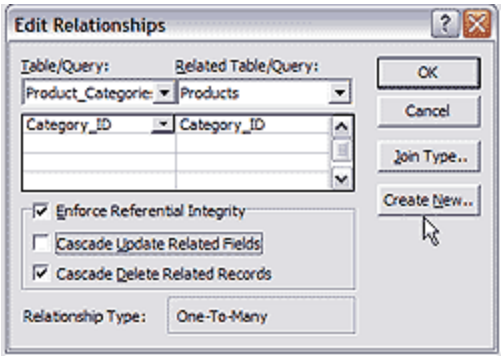
Open the database in Access and select the relationships window using the icon shown in the heading of this section. It is found in the main toolbar. The relationships wizard will appear as shown above.Highlight and add the tables you for which you want to define relationships. Add the Product_Categories, Products, and Versions tables to the relationships layout window. Next click on the Category_ID field of the Product_Categories table and drag the icon that appears at your mouse pointer to the Category_ID field of the Products table. The define relationships window will open.

---

* The Access Upsizing Wizard will convert the relationships to triggers when upsizing the Access database to SQL Server removing any need to write the SQL statements.

† This same method of coding will work with structures that have fewer or more tables by merely adding or taking away recordsets and loops.

As you can see the correct tables and fields are selected in their respective boxes. You will, however, need to check “Enforce Referential Integrity” and the “Cascade Delete Related Records” checkboxes. Now repeat the process for the Products and Versions tables. When you finish the relationships should something like the image to the left. Save your changes and close Access.



Creating the Recordsets

We need to three recordsets on the delete page. One recordset—rs_versions—will be conditional to keep the page from crashing when deleting categories without products. We will also need to custom code the SQL query to use the rs_products("P_ID") value to select the records from the versions table marked for deletion.

The first recordset we’ll create is rs_categories. The only parameter we need to pass from our recordset navigation page to the delete record page is the Category_ID value. This will be the filter. If your navigation page uses links to select records for deletion, then the filter will be a URL querystring. If you use a drop down menu it will be a form variable. The rs_products recordset will use the same filter. For reference see the UD generated recordsets below:

```
<%
Dim rs_categories__MMColParam
rs_categories__MMColParam = "1"
if (Request.QueryString("Category_ID") <> "") then rs_categories__MMColParam =
Request.QueryString("Category_ID")
%>
<%
set rs_categories = Server.CreateObject("ADODB.Recordset")
rs_categories.ActiveConnection = MM_yourdatasource_STRING
rs_categories.Source = "SELECT * FROM Product_Categories WHERE Category_ID = " +
Replace(rs_categories__MMColParam, "'", "''") + ""
rs_categories.CursorType = 0
rs_categories.CursorLocation = 2
rs_categories.LockType = 3
rs_categories.Open()
rs_categories_numRows = 0
%>

<%
Dim rs_products__MMColParam
rs_products__MMColParam = "1"
if (Request.QueryString("Category_ID") <> "") then rs_products__MMColParam =
Request.QueryString("Category_ID")
%>
<%
set rs_products = Server.CreateObject("ADODB.Recordset")
rs_products.ActiveConnection = MM_yourdatasource _STRING
rs_products.Source = "SELECT * FROM Products WHERE Category_ID = " +
Replace(rs_products__MMColParam, "'", "''") + ""
rs_products.CursorType = 0
rs_products.CursorLocation = 2
```

```
rs_products.LockType = 3
rs_products.Open()
rs_products_numRows = 0
%>
```

Nothing fancy there, but it gets interesting now. Create the recordset rs_versions in the data bindings window. Don't bother to set any filter we will do that in code view. After you've created rs_versions, open the Code Inspector. Notice the comment I've added to the code block and the following conditional statement. If rs_products is empty rs_versions is not created for two reasons. First the page will crash if it tries to create rs_versions without a rs_products("P_ID") value. And second there are no version images to delete because there are no versions. The next code change is in the SQL query statement. We add a "Where" directive and set the "Where" condition equal to the rs_products record ID number ("P_ID"). When you're done, close the "if then" statement, save the file, and return to design view.

```
<%
'set condition for versions recordset creation
If Not rs_products.eof  then
set rs_versions = Server.CreateObject("ADODB.Recordset")
rs_versions.ActiveConnection = MM_yourdatasource_STRING
'set versions recordset equal to product ids
rs_versions.Source = "SELECT * FROM Versions WHERE P_ID =" & rs_products("P_ID")
rs_versions.CursorType = 0
rs_versions.CursorLocation = 2
rs_versions.LockType = 3
rs_versions.Open()
rs_versions_numRows = 0
end if
%>
```

When you get back to design view the first thing you should notice is that the rs_versions recordset has vanished from the data bindings window. Don't panic—it's supposed to do this. Go ahead and add the delete records server behavior to the page for rs_categories.

Okay, now that we have the recordsets we need to get all the image names for deletion. To accomplish this we need a repeat region to loop through the Products table and one to loop through the Versions table and collect image values for the delete. You can use any method to create two repeat regions that works for you. I pull a field value from the Products table and apply the repeat region behavior selecting show all records. I then return to code view, copy and paste a a copy of the repeat region and change the variable names as shown. You could also drag field elements from the Products and Versions onto the page before modifying rs_versions and cut and paste the resulting code. Regardless of method you need two repeat regions on the page one for rs_products and one for rs_versions.

```
<%
Dim Repeat1__numRows
Repeat1__numRows = -1
Dim Repeat1__index
Repeat1__index = 0
rs_products_numRows = rs_versions_numRows + Repeat1__numRows
%>
<%
```

```
'repeat behavior modified  for nesting
Dim VRepeat1__numRows
VRepeat1__numRows = -1
Dim VRepeat1__index
VRepeat1__index = 0
rs_versions_numRows = rs_versions_numRows + VRepeat1__numRows
%>
```

The Delete Code

First we need to add the functions for the image deletes. I use the functions from Marcellino Bommezijn's excellent tutorial on image deletes. These can be added to the page or—better yet--in an include file.

```
<%
Function newFileSystemObject()
set newFileSystemObject=Server.CreateObject("Scripting.FileSystemObject")
End Function
%>
<%
Function fileExists(aFileSpec)
fileExists=newFileSystemObject.fileExists(aFileSpec)
End Function
%>
```

Find the UD delete SB and look for the comment in bold grey type below. We create the File Scripting Object (FSO) that will handle the category image deletes, set variable values for the path and image names for rs_categories, then call Marcellino Bommezijn's functions to delete any rs_categories image that exists.

```
' *** Delete Record: construct a sql delete statement and execute it

If (CStr(Request("MM_delete")) <> "" And CStr(Request("MM_recordId")) <> "") Then

  ' create the sql delete statement
MM_editQuery = "delete from " & MM_editTable & " where " & MM_editColumn & " = " &
MM_recordId

If (Not MM_abortEdit) Then
' execute the delete
Set MM_editCmd = Server.CreateObject("ADODB.Command")
MM_editCmd.ActiveConnection = MM_editConnection
```

Next Find the edit connection line of the delete SB shown bold in the line above—it's the last line before the UD delete SB executes--and make some space because here's where the magic happens.

```
' This is where we delete the category image file before we delete the record!
 Set File = CreateObject("Scripting.FileSystemObject")
 ImagePath = Server.MapPath("..\site_images\")
 ImagePath = ImagePath & "\" & (rs_categories.Fields.Item("Category_Image").Value)
 ' check if file exists and if true delete the category image file
 If fileExists(ImagePath) Then
 File.DeleteFile(ImagePath)
```

```
End If
```

Now that we've taken care of the Product_Categories image, we must nest the repeat regions we created earlier and modify them to perform image deletes. This allows us to employ our repeat regions to loop through the rs_products and rs_versions tables and extract values for the image deletes. Insert the opening "While Wend" statement from the rs_products repeat region we created earlier.

```
' loop through rs_products
While ((Repeat1__numRows <> 0) AND (NOT rs_products.EOF))
```

Before we delete our product images, however, we need to delete the rs_versions images. Remember the rs_versions recordset we created above uses the rs_products("P_ID") value to select related rs_versions records marked for deletion and extract the image field values. So we add our rs_versions "While Wend" statement immediately below the rs_products "While Wend."

```
' loop through and delete version images using a nested repeat behavior
While ((VRepeat1__numRows <> 0) AND (NOT rs_versions.EOF))
```

Now that we have collected all the image values, we create our rs_versions (FSO) and set variables for the image path and name values, and call Marcellino Bommezijn's functions to perform the file delete as above. Make sure you make the modifications needed to pull image name values from rs_versions. If the path to your version images is different you will need to change that as well.

```
  Set File = CreateObject("Scripting.FileSystemObject")
  ImagePath = Server.MapPath("..\site_images\")
  ImagePath = ImagePath & "\" & (rs_versions.Fields.Item("Version_Image").Value)
  ' check if file exists and if true delete the file
  If fileExists(ImagePath) Then
  File.DeleteFile(ImagePath)
 End If
'close rs_versions repeat region
  VRepeat1__index=VRepeat1__index+1
  VRepeat1__numRows=VRepeat1__numRows-1
  rs_versions.MoveNext()
Wend
```

Okay, now the we've taken care of the version images we close the rs_versions nested loop and re-enter the rs_products loop to perform the rs_products image delete. Again add the FSO and delete code.

```
'fall out of rs_versions loop and delete rs_product image
Set File = CreateObject("Scripting.FileSystemObject")
 ImagePath = Server.MapPath("..\site_images\")
 ImagePath = ImagePath & "\" & (rs_products.Fields.Item("P_Image").Value)
 ' check if file exists and if true delete the file
If fileExists(ImagePath) Then
File.DeleteFile(ImagePath)
End If
Set  File = Nothing
'close rs_products repeat region
Repeat1__index=Repeat1__index+1
  Repeat1__numRows=Repeat1__numRows-1
```

```
  rs_products.MoveNext()
Wend
%>
```

That's all there is to it. The stock UD generated code deletes the rs_categories record and the relationships we created in our access database cascades the delete to related records in the Products and Versions tables.

```
'Execute record delete
    MM_editCmd.CommandText = MM_editQuery
    MM_editCmd.Execute
    MM_editCmd.ActiveConnection.Close

    If (MM_editRedirectUrl <> "") Then
      Response.Redirect(MM_editRedirectUrl)
    End If
  End If
End If
%>
```