Data Mover Help

A very grateful “Thank You” to Albert Kallal who simplified the code brilliantly by showing me a way to move the attachments directly from one table to another without the complexity of saving them to disk first.

The intent of this tool is to allow one migrate data from an older back end to a newer one. At least that is what I will use it for. As a developer, I distribute a split database architecture application during the 12 – 18 month development cycle improve the software via added functionality in the front end and associated changes to the back end.

The normal major upgrade involves retrieving user’s back end databases and moving the data into a current database. The typical procedure involves code to all myself or an assistant or a sophisticated end-user to browse from a new front end to an older back end file then code drives the following sequence:

Remove sample data from the new back end.

Link to the tables in the old back end

Run append queries to move the data from the old back end tables to the new ones

Unlink the old tables

An issue arose when I jumped on the attachment field as a new convenience for user and added attachment fields. I discovered quickly that Microsoft didn’t allow any way to move attachments directly from one table to another. This rendered the whole process impractical and I had to remove the attachment “feature” from our software.

This code solves the problems and fixes the issue.

First, it can be a direct replacement for all the hard-coded queries we currently use since one simply calls the movedata2 function and supplies the two table names (source and target) as strings.

Second, it solves the attachment field issue since it allows one to optionally specify the name of an attachment field and it loops through the source table’s attachment fields, saving each attachment to disk and then loading the attachments into the corresponding field in the target table.

Additionally, I suspect this code will “move” data more quickly than queries, but have not tested that hypothesis.

Other functionality and demonstrations:

In order to determine the names of the fields in the source table, we create a table of field definitions (modified from code supplied and accredited in our code).

A simple help system using rich text fields and a dedicated form that provides attractive output at no cost, other than the time involved in creating the document content (I use MS Word) and paste it into the form or associated table..

Constraints:

If an attachment field is specified, both tables must have an attachment field and the field name must be identical in both tables

In use, I'm assuming that in the course of time the Target table may have a greater number of fields and that the added fields are not required fields – the code should handle them (by ignoring them), but if you find better ways of supplying needed data, please let me know.

The source table cannot have a field named "no attachment" :)

Contents of Package

Forms

frmHelp
frmDataMover
frmDataMoverSub1
frmDataMoverSub2
General Function Test (not required but perhaps useful, provided originally by Allen Browne to convert Query SQL into Code-readable SQL)

Tables

Table_Field_Definitions (not required since this code will automatically create it for you)
SourceTable
TargetTable

SourceTable2 (shows how code can handle tables without an attachment field)
TargetTable2 (pairs with SourceTable2)

Queries

qryFieldName (only one required for the system in your application
A few other queries used to drive the demo form frmDataMover

Reports

Help

Modules

Utilities (mostly used in other sample databases) but needed to drive the help system

Functions In Module basRecordHandling

1. Function MoveData2(FromTableName As String, ToTableName As String, Optional AttachmentFieldName As String = "No Attachment") This is the heavy lifter. It calls the next two functions during processing

2. CreateTableofFieldDefs("Table_Field_Definitions") 'this will blow away the table if it exists and recreate it, ensuring only fresh field definitions and no dups.

3. Call GetSpecificTableDefinitions(FromTableName)

4 Function fExistTable(strTableName As String). Is used to check if the table exists that we need to delete and recreate.

Comments and a call for help:

1. Yes, I know I overdo typing of the tempvars. The reason is that I've found them temperamental in queries and when capturing the data in form controls. Since sometimes they need the typing and sometimes not, it's just easier for me to always type them.

2. Yes, I know when I use a string in a function call, I don't have to type it – just a bad habit brought on by addiction to tempvars.

3. I'd love to see what some of you do with this – please feel free to comment, and send improvements. With your permission, I'd be happy to post improvements and give you accreditation.

4. For all you who do great things and never find time to give back to the community – all I can say is that the process of giving helps you more than you would think. Please consider helping us out.