The Ultimate Relinker

Paul Strauss Paul@PStrauss.net

Why Yet Another Relinker

Anyone who has been working with MS Access for a few years has either written for themselves, or has found any number of, procedures that relink the Back-End to the Front-End. They are all very similar, all of them being slight variations on a theme, and they are all seem to be good solutions to the problem. There is only one problem with all of these procedures that I have written or that I have found – *they require some user interaction*. The user typically has to know where the Back-End is and what it is called. For some of my clients, even finding the Back-End with a normal navigation pane is more than they are capable of doing. *This Relinker solves that problem because the user doesn't have to know where the Back-End is nor what it is called*. It doesn't require any interaction with the user at all. The only restriction is that the Back-End file name can't be changed by the user.

How It Works

The Ultimate Relinker first starts out by finding a linked table. It then gets the Connection String and extracts the file path and file name. If that file at that path exists, then the Connection String is pointing at a valid Back-End, which means that the Back-End is correctly linked, and so it just exits because there is nothing to do1.

If the Back-End doesn't exist at that path, then it assumes it exists somewhere on the user's computer and it starts to look for it. It recursively looks on every hard drive and in every folder and subfolder until it finds the present location of the Back-End. It then builds the new Connection String and updates the Connection String Property for every linked table. Since scanning an entire computer can take some time, it opens a form and displays the current folder it is scanning which server as sort of a progress meter. When the process completes, the form is closed. The user doesn't have to do anything except wait.

It is true that if the application has to relink the Back-End it can take several minutes but this delay only happens once unless a new Front-End is delivered or the location of the Back-End changes. Checking for and finding the Back-End where the Connection String says it ought to be causes no perceptible delay in starting up the application.

If you need to change the name of the Back-End, then you have to manually relink, one time only, to the renamed Back-End. This is because the file name in the Connection String is used to find the name of the Back-End. I considered adding a utility function to change the Back-End file name, but since it is so easy manually relink and changing the Back-End file name is probably a rare event, I decided that writing such a utility function is probably not worth the effort.

Caveats

1) As written, there can be only one Back-End file. If there were two or more Back-End files, it would have to look for each Back-End and keep track of the Connection String for each table. Since scanning the hard drive can take such a long time, I would first find all of the linked tables, their

[1] A linked table simply seems to be a table which has a Connection String beginning ";DATABASE=" specifying the path and file name of the linked file. It appears that when MS Access attempts to open the table, it simply grabs the connection string and uses it to find the Back-End. Therefore, for a Back-End to be properly linked, it only requires that the Connection string is properly pointing to Back-End. It is only necessary, therefore, to make sure that the Back-End is named and is located where the Connection string indicates. Trying to open a linked table and then looking for an Open Error works, but I never liked using that approach as it seems to me to be inelegant at least.

Connection Strings, and their Back-End file names and scan for all of the missing Back-Ends at the same time. So far, I don't need such a feature, I don't expect to, and so I didn't add this feature. I have worked with an application for which there was more than one Back-End, but from my experience, I believe this to be a rather rare circumstance.

2) **I think it will scan a networked drive if it is mapped to a drive letter**. I don't have a network to test this on so I can't guarantee it. I also think that there is one place where it a test is made for the file/folder type and this test may need to be modified to allow mapped drives. If you need this feature, I'll work with you to implement it.
3) The usual: It works for me but if it blows up your computer, you have been warned.