

Re: Homegrown synchronization

Source:

<http://www.tech-archive.net/Archive/Access/microsoft.public.access.replication/2006-12/msg00166.html>

- *From:* "David W. Fenton" <XXXusenet@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 27 Dec 2006 17:07:50 -0600
-

"rdemyan" <rdemyan@xxxxxxxxxxxx> wrote in
<news:1167243172.045423.134430@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>:

[]

b) In order to determine if there are record that were deleted, the code has to check each table. I don't see how a filter can be applied in the current setup. That means that the process for over 100 tables is relatively long.

If you used a DELETED flag and left the deleted records in the tables and just hid them from users in forms and reports and your processing routines, then you wouldn't have to anything at all, since the UPDATE process would propagate the DELETED flags.

*****How do I apply updates to the server
backend*****

[]

But the real question is how do those updates in the ImportDropBox for the server get applied. There is no program running in the background on the server like the Jet Synchronizer. Window's scheduler is not available to us to launch an Access program that could maybe handle this (although I'm not completely sure how).

As I told you, you're just moving the problems around by hand-coding this, as you still need capabilities that your brain-dead IT people are prohibiting you from using. I really think you need to go to management on this and explain that IT is being too restrictive and it's costing extra money and effort in your work.

Re: Homegrown synchronization

In one form or another,
this has to somehow be user initiated.

My current plan is:

- 1) Only users that are connected to the server via a LAN can initiate the updating of the server backend (no WAN users). If I keep the code for this in my app, this is easy to do since I have a user setup screen which administrators use to set permissions to view data. This would just be one more permission.
- 2) Should I use a hidden form in my app with a timer that monitors if new update files have appeared in the server ImportDropBox. Then the user could be alerted to apply the update or this could be done automatically. I could get away with this because in general we don't update data very often. The code would of course check if any of the update files in the dropbox had already been imported (each import is logged to a table).

I think the way I'd do this in your situation is:

1. have LAN users run the update process on the server when they upload their own updates to the server (assuming they are using the same methods as the remote users, which may be wrong -- the LAN users may be updating the server back end directly).
2. on a workstation that stays on all the time, run an Access app that has one form with a timer that simply runs the update code on the server at a specified interval. Or, that looks in the dropbox every N minutes and processes any files there. Running it in a Locked Workstation logon on a workstation removes the IT restrictions. On WinXP, it could be just a "switch user" kind of thing, and as long as the machine is left on, it will just run.

A better alternative might be to launch another program at the same time myapp is launched. This second program would be the one monitoring the server ImportDropBox and would automatically apply updates. But right now, I'm not sure if a timer on a form within in application that does not have the focus, works within Access. I need to research this.

You wouldn't want but one instance of this update program, though.

But either method requires that someone is logged in. Even if I use a separate app to accomplish the server synchronization, it is always going to require that someone is logged into their computer at a minimum.

Re: Homegrown synchronization

Yes. This is where the IT restrictions end up being just exactly as restrictive for your hand-built synch as they are for Jet replication. What i described above is *exactly* equivalent to running your synchronizer on a workstation.

But let's keep things in perspective and look at how we're doing it now (BTW: the app is still in the deployment stage so we are still working out coding and logistic bugs and adding customization; but I need to figure this "synchronization" issue out and deploy it soon). Updates are sent to a central person (typically by spreadsheet or e-mail) and that person has to manually update the server backend. Then after the server backend is updated an e-mail is sent out and remote users have to download the updated backend from the server.

Having a human administrator apply the updates via your Access code should be a huge improvement, no? That way you won't have to worry about all the problems that come with trying to do your updates automatically.

So from that perspective, this is a large improvement. Further, since there are multiple users using the app that are also connected to the LAN where the server resides, each one would be given permissions to update the server backend so there would be some redundancy. There will still be latency, but it will be much shorter than it is now. Of course, Jet indirect would be better yet (but IT won't yet allow it). OTOH, one of my other concerns is that MS will abandon Jet replication as they currently seem to be on that path (I don't know how well it is supported in A 2007).

The new ACCDB format does not support replication. Nor does it support user-level security. However, you can still create MDB files in A2K7 and replicate them and implement user-level security. So, from my point of view, it's just about the same, except for the fact that the new features of the ACCDB format are not available when you use the "old" format.

I don't think that's much of an issue for replication, as you can surely use an ACCDB for your front end and link to a replicated back end. Thus you get most of the new features of the ACCDB (which are almost all UI-related and not data-related, with the exception of the multi-value fields, which I don't believe I'd use, in any case).

--

David W. Fenton <http://www.dfenton.com/>
usenet at dfenton dot com <http://www.dfenton.com/DFA/>

Re: Homegrown synchronization