ADO and Subform Performance

Peter Vogel

Peter Vogel answers some thorny questions about using ADO to update a view (you can't) and setting the RecordSource property of a subform dynamically to improve an application's performance.

I' m trying to update a recordset, but every time I run the code, I get the following error: "Run Time Error 3251: Current recordset doesn't support updating. This may be a limitation of the provider, or the selected locktype." When I researched this error with Microsoft, they said it was occurring because I didn't specify the locktype and that, by default, a recordset is read-only. I checked, and I did specify the locktype as adLockPessimistic, but it still doesn't work. You should know that my recordset is based on a command object that calls a stored procedure.

The problem is probably occurring because you're using a stored procedure. When you change the value of a field in a recordset, under the hood ADO generates a SQL statement to perform the update. That SQL statement is then sent to whatever database management system actually owns the table (typically Jet). To figure out what that update SQL statement should be, ADO looks at the SQL statement that you used to generate the recordset. Since all that ADO has to go on in your example is the name of the stored procedure, ADO can't figure out how to generate a SQL statement to do the update.

I'm afraid that you have only two solutions:

- Create another stored procedure (and command object) that you can use for updates.
- Issue the update SQL yourself.

I recommend the second choice, as the update statements generated by ADO aren't very good. For instance, assume that ADO will create a SQL statement that updates every field retrieved in the original Select statement even if you only change one field in the recordset. This creates the following scenario:

1. You retrieve fields A and B.
2. Another user retrieves fields B and C.
3. You update field A.
4. The other user updates fields B and C (which are related).
5. The other user puts his data back before you.
6. Your update goes through with your new value for field A and the original value for field B.

The record now has your new value for field A, the original value for field B, and the other user's new value for field C. Not only has the other user lost his change without notification, but fields B and C (which I said were related) may be in conflict. Many developers solve this problem by locking a record when they retrieve it (pessimistic locking), which reduces an application's scalability. However, if each of you had only updated the fields that you changed, there would be no conflict and no need for pessimistic locking.

Does ADO update unchanged fields? You don't know. And even if you did, can you guarantee that future versions of ADO won't change that behavior? If your application is installed on a system with an older version of ADO, do you know that previous versions didn't work this way? From a performance point of view, it probably doesn't make any difference whether you issue the SQL statement or ADO does, so this is one area where you shouldn't give up control.

ADO.NET solves this problem, to a certain extent, by allowing you to specify the update, insert, and delete statements for a recordset. You can also get ADO.NET to generate the statements as ADO does if you don't want to do it yourself or are happy with what ADO.NET creates for you.

I'm trying to implement the general approach of passing SQL strings into forms and reports that you described in your March 2002 article, "Access Efficiency." Setting the RecordSource of my forms to retrieve only the data that I want has sped up my application. However, I can't figure out how to set the RecordSource of a subform. I tried passing a string into the main form and then setting things like this:

```
Me.<subformName>.SourceObject = lstOpenArgs
Forms!<subformName>.RecordSource = lstOpenArgs
Me!<subformName>.RecordSource = lstOpenArgs
```

All gave me different varieties of errors. The subform works fine when I put a query name in the RecordSource and filter it, so the problem isn't with form.

A subform is a kind of object within a form—really just another kind of control, a control that's used to hold forms. The subform control has a bunch of interesting properties, including one that allows you access to the form within the subform control. That property is cleverly

called Form. As a result, this code should let you get at the properties of the form within the subform control:

```
me.<subform name>.Form
```

So, for instance, to get at the RecordSource property of the subform from the main form, you'd use this code:

```
me.AccessLinks_Sheet1_List1.Form.recordsource = Me.lstargs
```

Unfortunately, you can't get at the subform before it's opened within the host form without opening the form in design view (and, even then, I don't know if you'd be able to work with the subform). The best that you can do is to open the main form and, in the Load event in the main form, set the RecordSource property of the subform. ▲

Peter Vogel (MBA, MCSD) is a principal in PH&V Information Services. PH&V specializes in system design and development for systems that use Microsoft technologies. Peter has designed, built, and installed intranet and component-based systems for Bayer AG, Exxon, Christie Digital, and the Canadian Imperial Bank of Commerce. He's also the editor of Pinnacle's *Smart Access* and *XML Developer* newsletters and wrote *The Visual Basic Object and Component Handbook* (Prentice Hall, currently being revised for .NET). In addition to teaching for Learning Tree International, Peter wrote its Web application development, ASP.NET, and technical writing courses, along with being technical editor of its COM+ course. His articles have appeared in every major magazine devoted to VB-based development, can be found in the Microsoft Developer Network libraries, and will be included in Visual Studio .NET. Peter also presents at conferences around the world. peter.vogel@phvis.com.

Smart Search...

Continued from page 11

Progressive searches

In progressive searches, the search criteria specified by the users will be added to the existing criteria. For this, the current SQL statement attached to the pass-through query needs to be evaluated. This is done as follows:

```
Public Function fnGetSQLQuery(strQuery As String) _
                                    As String

   Dim db As DAO.Database
   Dim qdf As DAO.QueryDef

   Set db = CurrentDb
   Set qdf = db.QueryDefs(strQuery)

   fnGetSQLQuery = qdf.SQL

   qdf.Close

   Set qdf = Nothing
   Set db = Nothing

End Function
```

Once the SQL statement is evaluated, it will be treated as before, with minor adjustments as described earlier.

So with Smart Search, you can now try your hand at power searching, the simple way. Happy searching. ▲

DOWNLOAD SQLSRCH.ZIP at www.smartaccessnewsletter.com

Nirmala Sekhar runs her own software consultancy in Singapore. She's currently working on Access, ASP, XML, and SQL Server 2000 platforms. nirmala@saicomsystems.com.

Starting the Interface...

Continued from page 16

set on the server side: Server Filter and Server Filter By Form. However, if you're using a stored procedure in the record source for a form, the Server Filter By Form isn't available. To set the Server Filter property, simply put a WHERE clause in the property (without the word "WHERE"). For example, to restrict my auto Staff form to those staff members whose forenames begin with A, I'd set the Server Filter property to this:

```
StaffForeName Like 'A%'
```

The property can also be set via code. For example, by adding the following code to a command button, I can change the server filter currently being used:

```
Me.ServerFilter = "StaffSurname Like 'R%'"
Me.Refresh
```

In my next article, I'll be looking at building several forms for the project and looking at stored procedures as a means to filter records returned to the client. I'll also look at how common Access form objects behave in the world of SQL Server. ▲

Martin Reid is an analyst with The Queens University of Belfast. He's the co-author, with Susan Sales Harkins, of *SQL: Access to SQL Server*, published by Apress (www.apress.com) and has worked on several major database titles as a technical editor. Martin and Susan (and some of the best Access developers around) can usually be found on the AccessD Developers List. See www.databaseadvisors.com for details.