

Re: DAO to ADO Recordset Options

Source: <http://www.tech-archive.net/Archive/Data/microsoft.public.data.ado/2006-06/msg00080.html>

- *From:* "Stephen Howe" <stephenPOINThoweATtns-globalPOINTcom>
 - *Date:* Thu, 22 Jun 2006 15:58:07 +0100
-

doing so, and after reserching the many postings, it seems that I need to migrate much of my DAO code to ADO. Mainly, it seems that this greatly improves performance.

DAO was really fast when dealing with Access data (faster than ADO) but was slower for other databases.

ADO generally does well for all Databases but just not as good as DAO when dealing with Access.

ADO is good all-rounder. In some senses it is not ADO but the provider it is attached to.

I have not done the measurement myself

- 1.) Use strict SQL commands vs. ADO for retrieving data.

That does not make sense as you can issue SQL commands when opening a ADO Recordset.

I tend to use Recordsets only for SQL SELECTs.

I hardly use Recordsets if doing INSERTs, UPDATEs or DELETEs.

- 2.) Don't migrate to ADO, continue using DAO.

You can do that but I would not recommend it.

- 3.) Use the following method for opening a recordset:

```
Dim rst As New ADODB.Recordset 'New for SQL server
Set db = CurrentProject.Connection
sqlCode = "SELECT * FROM [PRODUCTCONFIGURATIONS]"
rst.Open sqlCode, db, adOpenStaticOnly, adLockOptimistic 'New for SQL
```

No. If you do

Re: DAO to ADO Recordset Options

Dim rst As New ADODB.Recordset

it is impossible to work with Stored Procedures that return multiple resultsets. rst from now on can NEVER be Nothing as ANY inspection of it causes VB to instantiate the object. That make it fractionally slower.

Better is

```
Dim rst As ADODB.Recordset
Set rst = New ADODB.Recordset
'do any setup stuff
rst.Open
```

```
rst.Close
Set rst = Nothing
```

Now in terms of best practice with ADO:

1. I would read

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnmdac/html/improvperf.asp?frame=true>

It is bit dated as it does not cover using ADO Record objects (introduced in MDAC 2.8) for Singleton SELECTs which are about 15% faster than using a RecordSet.

2. Articles by Bill Vaughn are very good.

His very dark purple book "ADO Examples and Best Practices" was incredible.

See here for an article

<http://www.betav.com/Files/Content/Whitepapers/ADO%20Performance.htm>

3. In general I use RecordSet's for retrieving data and Connection and Command object for doing INSERTs, UPDATEs & DELETEs. I don't use RecordSet's for these (with one exception – later). But it depends precisely on what is being done. You will see what I mean in a minute.

On Recordsets, you should use the cheapest Recordset you need.

Not planning on updating? => Then open with LockReadOnly, anything else is more expensive.

Not planning on doing MovePrevious?=> Then open with ForwardOnly, anything else is more expensive.

In terms of Server–sided versus Client–sided, you should be aware of the differences in functionality. They both have their uses. I tend to use a lot of Server–sided, Forward–only, Read–only Cursors as they are the fastest read cursor of the lot. But it is also provides the least in terms of functionality – RecordCount will return –1 for such a cursor.

4. You should ALWAYS, ALWAYS specify all the parameters to Recordset Open.

It is the most important call of the lot as rs.Open() directly determines what functionality all the methods, properties rs will return after this call. VB programmers are most guilty of leaving off parameters and need their hands chopped off.

Re: DAO to ADO Recordset Options

Re: DAO to ADO Recordset Options

All 5 parameters should be specified.

5. Programmers frequently like to inspect RecordCount to see if they have any records.

This is a bad idea, as for some cursor types RecordCount is -1.

Better is

```
IF rs.BOF AND rs.EOF THEN
' There are no records if BOF and EOF are both true.
END IF
```

This works for all 4 cursor types.

6. Be aware that ADO can coerce CursorType and LockType. You might ask for some combination that is not available. ADO will choose something else. You can print CursorType and LockType _AFTER_ a successful rs.Open to see if what you requested is what you got. ADO will never coerce CursorLocation.

7. For client-sided cursors, there is only Static cursor type regardless as to what is requested.

Also rs.Open() will not return until it has built the entire recordset in your applications memory.

Once rs.Open() has returned, methods like Filter, Sort, Find work – they are designed for client-sided cursors.

They do not work well (if at all) with server-sided cursors.

RecordCount will always be correct.

You can also close the connection and still work with the Recordset

8. For server-sided cursors, what is available in terms of cursors, lock types depends on the provider.

rs.Open() will often return immediately (but the entire recordset has not been read).

For server-sided cursors it is very important to set CacheSize as it determines how often data is fetched from the server. RecordCount can be -1.

8. Returning to what I do for UPDATES

(i) If the UPDATE can be done as 1 line of SQL, I will do so with a Connection Execute

(ii) If the UPDATE is the same fields but different contents then
– with few rows to update, I will do multiple Connection Execute's
– with many rows to update, I will set up a Command object with an adhoc query and then use a for loop, changing the parameters each time round. This is a temporary Stored Procedure.

(iii) If the UPDATE is the different fields for each row there is no choice but a Connection Execute.

Permanent SP's are also good.

There is similar story for DELETES, INSERTs and SELECTs. What I do depends on how many times it is being done, whether it is repeated in any way etc, how many rows are being returned etc.

Re: DAO to ADO Recordset Options

9. As a whole, if it can be done with one line of SQL, it should be done.
For example somebody wanted to add missing rows in a table from an identical table in a different database.
They thought of using 2 Connections and 1 RecordSet.
But it can be done with 1 SQL command.
And that is far faster and less error-prone than 25 lines of VB/VC++ code.
With me?
Make the Jet Engine/Oracle Engine/SQL Server Engine work for you.

10. The sole exception at the moment is I use RecordSet for doing a series of many INSERTs and use BatchUpdate.
To do this, I need an empty Recordset to start with for table in question.
The only way of getting this in ADO is to use a WHERE clause that is always false.
So

```
"SELECT * FROM table1 WHERE 0=1"
```

You get back an empty Recordset – perfect to add new rows.

The only I am doing this is because I could not get Command Execute's to work faster on SQL Server and I found SQL Server BULK INSERT had setup limitations. But I have recently worked out how I might Command Execute's to work faster and in which case I will only be using Recordset for SELECTs.

Hope that helps

Stephen Howe

.