

## Re: Recordset.AddNew and the recordset object's data retaining

**Source:** <http://www.tech-archive.net/Archive/Data/microsoft.public.data.ado/2004-12/0113.html>

---

**From:** Jiho Han ([jhan\\_at\\_infinityinfo.com](mailto:jhan_at_infinityinfo.com))

**Date:** 12/10/04

Date: Fri, 10 Dec 2004 08:18:01 -0800

Mark,

I am in an environment where bulk insert isn't an option—even though the underlying database is SQL server, I am using a third party OLE DB provider—and DTS may be an option but in a lot of the scenarios, the script needs to be run in the application host rather than in a DTS environment.

I agree with you on doing batch inserts/updates. But there are cases, where that's not possible or even desired even if the performance took a big hit.

Thanks for your input.

Jiho

> *If your design is such that you really are planning to insert 10,000  
> records via an ADO recordset, then I'd strongly suggest you redesign  
> your solution.*  
>  
> *Even building a number of insert statements and executing them all in  
> one hit via an ADO command would be an option. Not to mention bulk  
> insert(assuming you are using SQL server) or DTS.*  
>  
> *"Jiho Han" wrote:*  
>  
>> *A couple of observations to your approach...*  
>>  
>> *The statement you make about the individual updates not hitting the  
>> database until the transaction is committed is totally different from  
>> my understanding of how ADO works. The behavior you specify, if I'm  
>> correct, has nothing to do with the updates being in a transaction  
>> but rather on whether you specify LockType to be  
>> adLockOptimisticBatch and use UpdateBatch method at the end.*  
>>  
>> *But even if things worked as you say it does, it doesn't resolve my  
>> issue with the client retaining all 10,000 records in memory(I don't  
>> mean physical memory only – in fact, if they were, it wouldn't be*

>> *that bad, it's when things spill over and the memory starts paging  
>> out to disk that concerns me). Because until you set the recordset  
>> to Nothing, all records will sit on the client machine. Maybe I'm  
>> overly concerned for nothing. But what if it were 100s of thousands  
>> of records.*  
>>  
>> *I have been thinking about this for a bit last night and thought of  
>> an alternate  
>> solution that may be ok for me.  
>> The reason I wanted to avoid Recordset.AddNew, albeit its  
>> convenience/power  
>> of ease, is that in order to avoid the above issue of retaining  
>> records in  
>> memory, I'd indeed have to set the recordset to nothing, to dispose  
>> them  
>> somehow.  
>> Instead, I am thinking to recycle it every certain number of records.  
>> Obviously this isn't something revolutionary, it is how batch jobs are  
>> traditionally done anyway.*  
>>  
>> *So, I would do something like this:*  
>>  
>> *' somewhat pseudocode follows*  
>>  
>> *Dim lngCount = 0  
>> Set objRS = "SELECT \* FROM ACCOUNT WHERE 1 = 0" ' This lets me get  
>> the schema  
>> quickly and without fuss.  
>> Do While Not Source.EOF  
>> objRS.AddNew  
>> objRS.Fields("NAME").Value = Source("NAME")  
>> ...  
>> Source.MoveToNextLine  
>> If lngCount = 3000 Then ' assuming we recycle every 3000 records  
>> objRS.Requery  
>> End If  
>> Loop  
>> That requery line is the key I think. That is a Close/Dispose and  
>> Open in one line. I'd adjust the interval depending on how the app  
>> performs. I haven't tested this out yet.*  
>>  
>> *What do you think?*  
>>  
>>> *How about this:*  
>>>  
>>> *'-----  
>>> Set rsMyAdodbRecordset = blah, blah, blah  
>>> '-----  
>>> cnMyAdodbConnection.BeginTrans  
>>> '-----  
>>> Until MyBigFatLoop = IsDone*

```
>>> rsMyAdodbRecordset.AddNew
>>> rsMyAdodbRecordset("Field1") = blah
>>> rsMyAdodbRecordset("Field2") = blah
>>> rsMyAdodbRecordset("Field2") = blah
>>> rsMyAdodbRecordset.Update
>>> Loop
>>> '-----
>>> cnMyAdodbConnection.CommitTrans
>>> '-----
>>> rsMyAdodbRecordset.Close
>>> Set rsMyAdodbRecordset = Nothing
>>> '-----
>>> Whaddaya think?
>>> I believe you will find that the records are added locally at high
>>> speed because the Update won't go back to the database during the
>>> transaction. When you close the recordset and set it to Nothing, all
>>> memory is reclaimed. You can use transactions with most providers,
>>> but not all.
>>>
>>> Just one thing, you might not want to open the recordset against the
>>> entire table or it just might return the whole table to local
>>> memory! (Wow! if it is a big table.) Can you do "SELECT * FROM
>>> MyTable WHERE MyTable.MyKey = 'nevercanhavethisvalue';" ????
>>>
>>> Good luck
>>>
>>> Jim Rodgers
>>>
>>> "Jiho Han" <jihohan@gmail.com> wrote in message
>>> news:1102458848.649336.188410@c13g2000cwb.googlegroups.com...
>>>> I would like to use Recordset object's AddNew method to insert a
>>>> new record. However, I am afraid, whenever I do so, the new row
>>>> I've inserted will be retained in memory on my client machine on
>>>> subsequent inserts.
>>>>
>>>> Imagine, inserting 10,000 records using either AddNew/Update or
>>>> UpdateBatch. AddNew is very convenient and easier to code than
>>>> creating a Command object and setting its Parameters collection.
>>>> With the former, I don't even have to worry about optimistic
>>>> concurrency since it's supported by default although in a batch
>>>> scenario such as above, I'd likely turn off concurrency support.
>>>>
>>>> What I don't want is though, memory bloat because of the new
>>>> records I've inserted. Requery or Close/Open will result in
>>>> additional trips to the DB which isn't the case with a Command
>>>> object. Can't have the cake and eat it too?
>>>>
>>>> Can someone clarify?
>>>> Thanks
```