

Re: dataset Performance Issue

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.adonet/2005-08/msg00164.html>

- *From:* "Sahil Malik [MVP]" <contactmethrumyblog@xxxxxxxxxxx>
 - *Date:* Wed, 3 Aug 2005 13:27:41 -0400
-

Nigel,

It is true that .NET 2.0's dataset has been really improved from within to handle greater quantities of data, but the justification of that is the desire to abuse it as a database. It shouldn't be interpreted as a nod from Microsoft that a DataSet is okay to abuse as a DataBase. (I am not Microsoft mind you).

There are a number of reasons for this.

1. First of all, Dataset is guess what – Managed Code, and to send all the ..NET lovers in tailspin, Managed Code can never be as fast and as optimized as native code. That is reasonable to expect and I am certainly not saying that unmanaged is better and drop managed code for good – NO WAY, there are a lot of benefits of using Managed code, but when it comes to raw performance, Managed code sucks.
2. Secondly, the Garbage collector is that animal that makes things very very good for 90% of the situations i.e. normal memory usage, but when you start storing many megabytes or close to a gigabyte of information completely in RAM – it will actually hurt your application performance. In those scenarios, you don't want an external policeman who doesn't understand the specific needs of your app. In that situation, you want fine control on the memory where you specify when it gets cleaned, or serialized to the disk etc. You need paging mechanisms etc. which are possible to write for the dataset but are a real royal pain to write and even then they don't work quite as well as – guess what – native code (i.e. most of what SQL Server).
3. SQL Server and any database comes with a "Query Engine". The number of optimizations built into that is the work of many Phds (or dudes with similar smarts and specialization), they have written up SQL Server's query engine to take advantage of automatic paging, locking algorithms, spilling over to the disk when needed, "query plans", caching those query plans – when you compare the object model of a Dataset (or any biz object for that matter), the comparison is like comparing a candle with the sun.
4. The algorithms in a DataSet is rudimentary, they rely on simple techniques such as string matching, string manipulation – that level of

Re: dataset Performance Issue

simplicity. They work on an "Object structure", every value they access goes over a dereferenced segment calculation. SQL Server and any standard database is written with that level of optimization (Not like I have looked at their code, but it has to be written that way). Maybe in certain instances DataSet might be smarter than I try to bring out here, but a DataSet is an in memory object – that is how simple it is. It's just an "object". It's worse than MS–Access when it comes to managing data for you. (Don't get me wrong, I still love datasets, I just don't think they replace databases). Even MS–Access will manage 4 GB, a DataSet doesn't even have a clear upper limit defined. But of course MS Access isn't an in memory, xml convertible, serializable in memory cache that lets you extract and merge changes – so comparing dataset with access is like comparing apples and oranges (but hey the argument is all about a DataSet is an Orange (in memory cache), not an Apple (Database)).

5. Lets not forget transactional locks and many other such points, I blogged about it earlier over here –

<http://codebetter.com/blogs/sahil.malik/archive/2005/01/23/47547.aspx>

6. Datasets are or any such object – AN IN MEMORY disconnected cache of data. Being completely in memory lends them to the disadvantage of a 32 bit OS's 2 GB memory allocation limit, there are ways around that but I personally see those ways as bandaids rather than a true solution. Secondly being disconnected leaves you with a WORLD of problems to solve when trying to persist relational data back into the database. If you think `DataAdapter.Update(DataSet)` will save your entire dataset into the database, and also take care of concurrency issues, and transactional deadlocks – you are sorely mistaken. A simple 3 table hierarchy will require you to write pages and pages of code to save properly into the database in every scenario – it is NOT a trivial task. And then you have to worry about not sending too much over the wire in web service like environments, so `Merge/GetChanges` – and oh lets not forget keeping your disconnected cache fresh. How do you resolve Deleted Rows? .NET 1.1 leaves you with very few choices, .NET 2.0 has a new `IndexOf` method which doesn't work in every circumstance.

Now of course you could argue that #6 proves that simply don't use a database, only use a disconnected cache i.e. prevayler etc. For that argument, re–read items #1 thru 5.

In short, while DataSets will continue to improve, or lets say, they will continue to try and be as good as a full fledged database is, they will NEVER reach that point. While even though datasets will improve, who knows you may even be able to run a SQL Query against them directly, but hey you could always store a little tiny such database on a RAMDisk, or a memory mapped file in SQL Server, and pretty much get what you need including the heavy duty research that has already gone in making "a database".

Again, I strongly and vehemently disagree with an architecture that puts 1 GB data into a DataSet. That is complete stupidity in both .NET 1.1 and 2.0.

Re: dataset Performance Issue

Whew .. this was a long reply .. gotta go !!

– Sahil Malik [MVP]

ADO.NET 2.0 book –

<http://codebetter.com/blogs/sahil.malik/archive/2005/05/13/63199.aspx>

"Nigel Norris" <nospam@xxxxxxxxxx> wrote in message
news:ugx7MDBmFHA.2860@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

>

> "Sahil Malik [MVP]" <contactmethrumblog@xxxxxxxxxx> wrote in message

> news:Of7iYB%23IFHA.576@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

>

>> But what I am certainly not a fan of is storing 400 MB in a DataSet –

>> that object is just not designed for such a heavy amount of in memory

>> data.

>>

> Sahil,

>

> You make that assertion – can you elaborate as to why you believe that? I

> don't see anything in the documentation, or the services that DataSets

> provide, that would make me believe that they are not designed to handle

> large amounts of data (given the inherent constraints of memory, garbage

> collecting very large heaps, etc).

>

> Now it appears to be the case that the V1.1 implementation has some

> performance problems in some areas with large tables, but at least some of

> these are fixed in V2.0. So Microsoft are making efforts to ensure that

> large volumns of data are supported. See the following article for some

> information:

>

> <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnadonet/html/datasetenhance.asp>

>

> While I would agree that the main use and focus for DataSets is

> *selective* caching of data, I can certainly envisage situations where I

> might want to hold quite a large extract of a table in a dataset (where

> connectivity is not always available, for instance).

>

> -----

> Nigel Norris

>

>

>

.

- **Follow-Ups:**
 - ◆ **Re: dataset Performance Issue**
 - ◇ *From:* Cor Ligthert [MVP]
 - ◆ **Re: dataset Performance Issue**
 - ◇ *From:* Nigel Norris

- **References:**
 - ◆ **dataset Performance Issue**
 - ◇ *From:* Ashishthaps
 - ◆ **Re: dataset Performance Issue**
 - ◇ *From:* Sahil Malik [MVP]
 - ◆ **Re: dataset Performance Issue**
 - ◇ *From:* Adrian Moore
 - ◆ **Re: dataset Performance Issue**
 - ◇ *From:* Sahil Malik [MVP]
 - ◆ **Re: dataset Performance Issue**
 - ◇ *From:* Nigel Norris

- Prev by Date: **WinForm datagrid**
- Next by Date: **Re: MySql ado.net connector problem**
- Previous by thread: **Re: dataset Performance Issue**
- Next by thread: **Re: dataset Performance Issue**
- Index(es):
 - ◆ **Date**
 - ◆ **Thread**