

Re: Cached ADO

Source: <http://www.tech-archive.net/Archive/Data/microsoft.public.data.ado/2005-06/msg00255.html>

- *From:* "Anwar Shafiev" <AnwarShafiev@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Thu, 30 Jun 2005 11:49:08 -0700
-

Stephen, thanks for the information. This is a kind of experience, which we cannot get just reading books. :) I agree with you. Any solution must be reasonable. In my case, I used a main map (primary key, cached object) and a vector of multimaps for indexes. As a result, searching on indexed fields takes some nanoseconds. I use static binding to work with recordsets. I wrote a wizard (btw, that's what I miss too in classic ADO) to generate both a static binding and a cached object classes out of database structures. So working with the layer is pretty easy. The cached recordset class is impemented in a template. API to the cached recordset is similar to ADO. I don't want to go deep into details, but I am satisfied with the performance, although the functionality is very limited. Well, I am almost satisfied. I think, any programmer will never be satisfied with the results. :) The perfomance is acceptable, but I really feel a lack of functionality. I cannot believe that nothing similar exists. At least, I know many developers who are forced to create their own caching layers like me. :-/

Anwar

"Stephen Howe" wrote:

- > > Stephen, my caching layer is based on STL too. I use maps, as they provide
- > > much faster indexed searching on a large emount of data.
- >
- > I also use maps but there are some riders to this:
- > (i) I only use maps if I have true key-value pairs that are logically
- > distinct. If I am dealing with a single entity (or class) then I use a set.
- > It does not make sense to break a class apart or repeat the key element
- > `_just_` to use a map. It took me a long time (months) to realise this.
- > (ii) For 2000 items or less, a vector will beat a map. Every time you insert
- > into a map, that is a memory allocation. That happens less often for vector.
- > And if you know the size in advance vectors `reserve()` will cut down on
- > memory reallocation. The fact that vectors insert is $O(N)$ and maps is $O(\log$
- > $N)$ barely matters if copying objects is cheap. map only really comes into
- > its own when copying is not cheap and there is a large number of items. You
- > have to have a very large number of items before map's $O(\log N)$ beats
- > vectors $O(N)$ due to the memory allocation/deallocation acting as a drag to
- > map's performance .
- > That 2000 items could be 5000 for say integers and less for heavy-duty

Re: Cached ADO

> classes.
> (iii) If a container once populated rarely changes, a sorted vector searched
> using lower_bound() is far faster than maps find. If it changes once in a
> blue moon, sorted vector vector still wins.
>
> This comes down to : Roughly I use map/set when > 2000 items and number of
> items is changing constantly over lifetime of container. Iterator validity
> is also a consideration.
>
>> As I already
>> described in my previous post to Bill, the basic functionality exists. But
>> that's not enough for me. I need to have powerful filtering capabilities,
>> extremely fast searching and data manipulation. The layer must be
> thread-safe
>> and must solve all collisions in multi-user environment. I would like to
> have
>> transactions, referential integrity, relationships between tables on the
>> caching layer too.
>
> Tough requirements. I don't see how "collisions" are solved intelligently.
> Problem is your describing a seamless world – which we do not have.
>
> In C++, you can write thread-safe code but it is not an automatic given.
> The programmer usually has to supply some design intelligence to make it
> thread-safe and of course bugs can creep in. I hate that. There is no way a
> newbie C++ programmer can _easily_ write thread-safe code. It requires an
> experienced programmer apply intelligence. And C++, as a language, does not
> make guarantees on thread-safety. A real problem.
>
> In the database world, SQL Server and some of Microsoft's client-sided ADO
> supplies searching, filtering, transactions, referential integrity,
> relationships but these relationships are not maintained once you have the
> data inside the client app.
>
> So it is a mish-mash. You ruled out .NET but I believe it makes some inroads
> in presenting a seamless environment. I don't think you will get a seamless
> environment with C++ and classic ADO.
>
>>Well, now when I think about all these things, which I
>> have to implement, I decide to buy an existing solution if it exists, of
>> course. :)
>
> Sure :-)
>
>>If not... Then I will continue developing my own layer.
>
> It sounds very similar to what we have :-)
> Of late I am using a class to represent a table and it is the class's
> responsibility to perform any read/updates to the table. The class caches
> the table as it see fit. All other parts of the app, deal with the class and
> its interface.

Re: Cached ADO

