20 key Points rising, or Inferred, From “Working with large lists in MOSS 2007” Paper

http://blogs.devhorizon.com/reza/?p=790

Suggestion:

I would name the data access methods a bit differently. I agree that my names are longer and may not be a good choice for captions, but they could pick up short names for each method such as “approach “ or “Method B” and use them instead of longer names This way, readers don’t have to scan the entire code to find out what the actual data access method is all about

Instead of “SPList with For/Each” I would say: For/Each Looping through SPListItemCollection of SPList.Items
Instead of “SPList with SPQuery” I would say Looping through SPListItemCollection of SPList.GetItems (SPQuery)
Instead of “SPList with DataTable” I would say Looping though the DataView (w/ Filtering) of SPListItemCollection.GetDataTable ()
Instead of “SPListItems with DataTable” I would say Looping through each DataRow of SPListItemCollection.GetDataTable ()

20 Key Points:

1. There are eight different ways to query data from lists (same order as they appear in the paper, but using my names):

    I. Browser Access

    II. SPList Methods:

    -For/Each Looping through SPListItemCollection of SPList.Items

    -Looping through SPListItemCollection of SPList.GetItems (SPQuery)

    -Looping though the DataView (w/ Filtering) of SPListItemCollection.GetDataTable ()

    -Looping through each DataRow of SPListItemCollection.GetDataTable ()

III. Lists Web Service

IV. Search Object Model

V. PortalSiteMapProvider

2. Methods you should stay away when querying large lists both in WSS and MOSS :

I. Browser View

II. Looping though the DataView (w/ Filtering) of SPListItemCollection.GetDataTable ()

III. For/Each Looping through SPListItemCollection of SPList.Items

3. Fastest methods to query large lists in WSS (Not explicitly mentioned in the paper) :

I. Looping through SPListItemCollection of SPList.GetItems (SPQuery)

II. Lists Web Service

III. Looping through each DataRow of SPListItemCollection.GetDataTable ()

4. Fastest methods to query large lists in MOSS :

I. PortalSiteMapProvider. Depending on the column you use in the WHERE clause of the SPQuery object used in this method and whether or not that column is indexed there might be some performance degradation. More about this in tip #10.

II. Search API

III. Looping through each DataRow of SPListItemCollection.GetDataTable (). Pleas e be note d that, for smaller sets of data, for example 1500 list items, this approach may outperform querying using Search API technique.

5. A list, in SharePoint, is capable of hosting up to 5 million items, but how to distribute these items in that list is an IMPORTANT trick! More importantly if you would like to go over 5000 items you'd better watch potential SQL Server locking issues discussed in tip # 17

6. Root folder of a list is considered as a **container**, as well as each folder created in the root or any subfolders. Think recursive here!

7. 2000 list items limitation **per list** that some folks try to forewarn you is absolutely bullshit. Bucketize your list (by creating folders and nesting them), then you can potentially have more than 2000 list items per list.

8. There is a 2000 list items limitation **per container**, but why is that the case?

    1. Out of the box browsing facility buckles under the pressure when a list grows over 2000 items per container That's mainly due to how "Views" work in SharePoint. If you ever come up with your own optimize d UI, you can go past this hard limit as well. Chances are, even if you come up with your own UI , you still find it much better to have max 2000 list items per fetch from the underlying list.
    2. Going over 2000 list items per container may impact other common operation such as adding and deleting items, even if you already came up with your own optimized UI to provide a good browsing experience.

9. Let me give you an example. If you have 4007 items and you want to store the m all in one list, best practices recommend that you place 1998 list items in one container (let's say in the root) then create two folders and distribute the rest of items (in this example remaining 2009 items).

10. SPQuery won't work across multiple folders in a single list unless the ViewAttributes property includes Scope="Recursive" That being said all the data access methods which rely on this object potentially have the same behaviour. For the same reason these methods can't be used to query data from multiple lists in one SPWeb or across multiple SPWeb(es).

11. ID column is a built-in numeric indexed field in all SharePoint lists.

12. Using an indexed column in a WHERE clause of a C ML query used in SPQuery object doesn't necessarily boost the performance. For instance when used in PortalSiteMapProvider, Seach API and Lists Web service , indexed columns result in performance degrade. You can use the ID column instead if possible. For other methods that involve SPList object, that is exactly opposite. Index columns always outperform the ID column.

13. PortalSiteMapProvider comes with some bake d caching capabilities which make it the fastest way of querying large lists in SharePoint. This class is used in navigation and navigation is almost everywhere so IT GOTTA BE FAST!

14. PortalSiteMapProvider class uses a method name d GetCachedListItemsByQuery and a SPQuery object as a parameter to the method call to check to see if the result exists in the cache. If they do , the method returns the cached result , and if not , here is what happens :

    I. Queries the list

    II. Stores the results in the cache

    III. Spit the result out to the caller

    As you can tell, caching is not that cheap and It comes at its own cost!

15. Although PortalSiteMapProvider is the fastest method for querying large lists in MO SS, it can potentially lead you to troubles if not used carefully.

    I. PortalSiteMapProvider uses the same caching facility (Object cache), available to many things scoped at each site collection. Obviously, you can go ahead and increase the amount of object cache (default is 100 MG) via browser, but bear in mind that object cache comes out of the same shared memory allocated to the application pool. In particular in 32-bit installations of SharePoint where you can only allocated 2G of memory to each app pool, you may want to be cautious how you would us e your memory.

    II. The same caching rules that apply in .Net are applicable here as well. If the underlying data frequently changes you'd better not use PortalSiteMapProvider at all. Remember, from previous tip that caching has its own cost.

16. The same rules that apply when you over-index a table in SQL Server, applies to SharePoint lists as well. Indexing may improve querying operations, but it also may negatively impact the performance of adding new records due to index re builds.

17. Always limit the number of the affected rows in various operations against a list. This is because:

    I. SharePoint stores all the lists of a given site collection in one table in the content DB.

    II. Lists with approximately more than 5000 items can get you trapped into locking incidents. For a smaller lists, SQL Server locks all the rows of a particular list (list items) when the above operations are performed against it.

However, If the affected rows are more a certain number (approximately 5000) , the entire table is locked and it won't be released until the transaction is completed. With this being said, you can potentially affect other lists in the same site collection.

Remember, this locking can be the result of fetching unreasonable number of records (list items) or updating (adding, editing and deleting) data in a large list. RowLimit on the SPQuery class is your friend to control the number of returned records May be that's reason you should stay away from storing 5000 list items in a list.

18. If you have too many deletion operations in a large list you'd better d o it in batches during off-peak periods.

19. Although using search object model is one of the quickest ways of retrieving data out of large lists , it definitely calls for some extra work to be done firs t.

    I. Managed properties must be configured to pick up custom properties in the list.

    II. Indexing has to be completed otherwise search method doesn't work

20. Even if you have solved all the issues mentioned above when working with large lists, make sure you test the crawler to see if it performs the indexing with no time out! Do it before you deploy anything to the production and before you spend any time on extra work to support large lists in your environment!

For comments please see: **http://blogs.devhorizon.com/reza/?p=790**