SQL Object Performance Maintenance

Syniti Software Support recommendations for Master Data Management (dspConduct)

Table of Contents

1 - Introduction

Syniti Software Support is creating this document in order to provide SQL Server Object performance recommendations to anybody who might be implementing or using a Master Data Management (formerly dspConduct) component in the DSP. While much of the information provided here will be applicable to other components within the DSP, please note that the material has been written with Master Data Management in mind.

For this document, a SQL object will be considered "slow" if it frequently causes a user to wait more than 1 second for an action to complete without warning the user. It's important to note the following implications of this definition:

- a 10-second stored procedure that runs on a button click is not slow if a pop-up is warning the user that the button event will take some time to complete
- a 30-minute stored procedure is not slow if it only runs in the background (never slowing down the user experience)
- a 5-second validation view is not slow if it runs on a DSP page that is only used once a week

When dealing with any performance problem, there are 2 steps. The first step is identifying the scope, or the list of items that are slow. The second step is speeding up those items. Oftentimes, it is necessary to review the items 1 by 1, but there are sometimes broad changes that can help speed up multiple items.

NOTE: Some of these broad changes are detailed in Appendix A, Appendix B, and Appendix C. You should check these before moving into the next section.

2 - Identifying Slow SQL Objects

Syniti Software Support recommends 2 methods for identifying SQL Objects that are running slow enough to negatively impact the DSP user experience. While each method is technically sufficient to complete the task, using both methods works best.

The first method we call the "user approach". It relies on educating the DSP users on how to properly use the DSP Page Performance Monitor and how to report the results they find. This approach may seem passive since it allows users to identify slow objects, but we find it is very important in establishing a strong, reliable channel of communication with the users. Without using this method, admins of the DSP and/or SQL Server are often unaware of exactly what users mean by "slow".

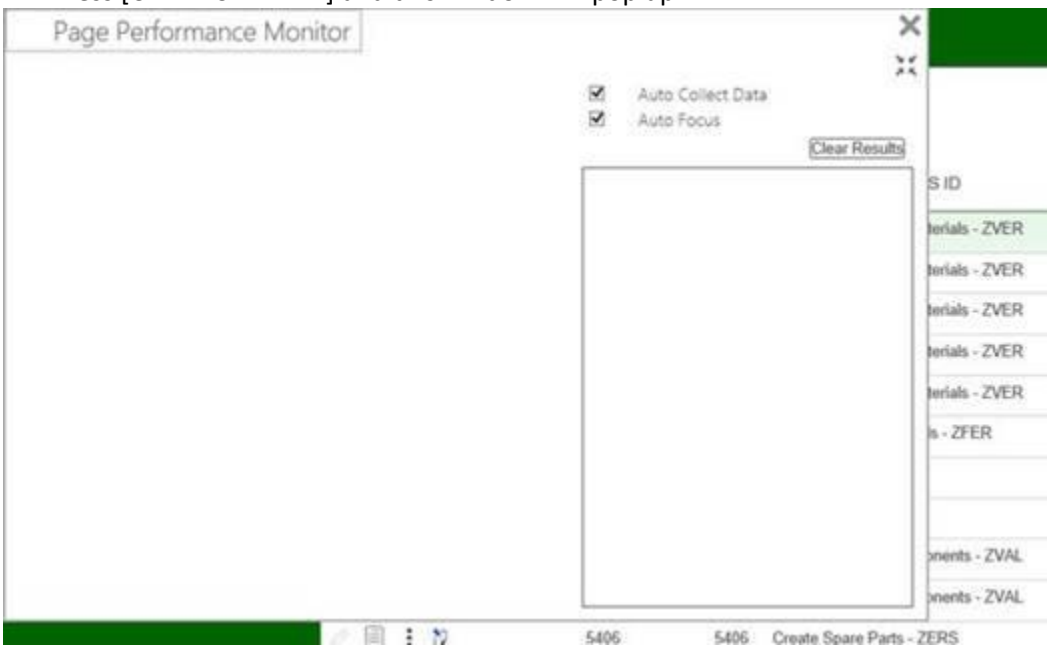
The second method we call the "admin approach". It utilizes the SQL Server Management Data Warehouse (MDW) reports. Instructions on configuring MDW can be found in Appendix D. This approach is more active, making it possible to prevent users from ever reporting slow objects. However, there will realistically be times when a particular slow object is difficult to identify in the MDW reports alone.

NOTE: When using the "admin approach", it will be necessary to make an individual (or team) responsible for reviewing the Query Statistics MDW report at some regular interval. If the rate of change in the SQL Server is low, then a weekly or even monthly review is probably sufficient.

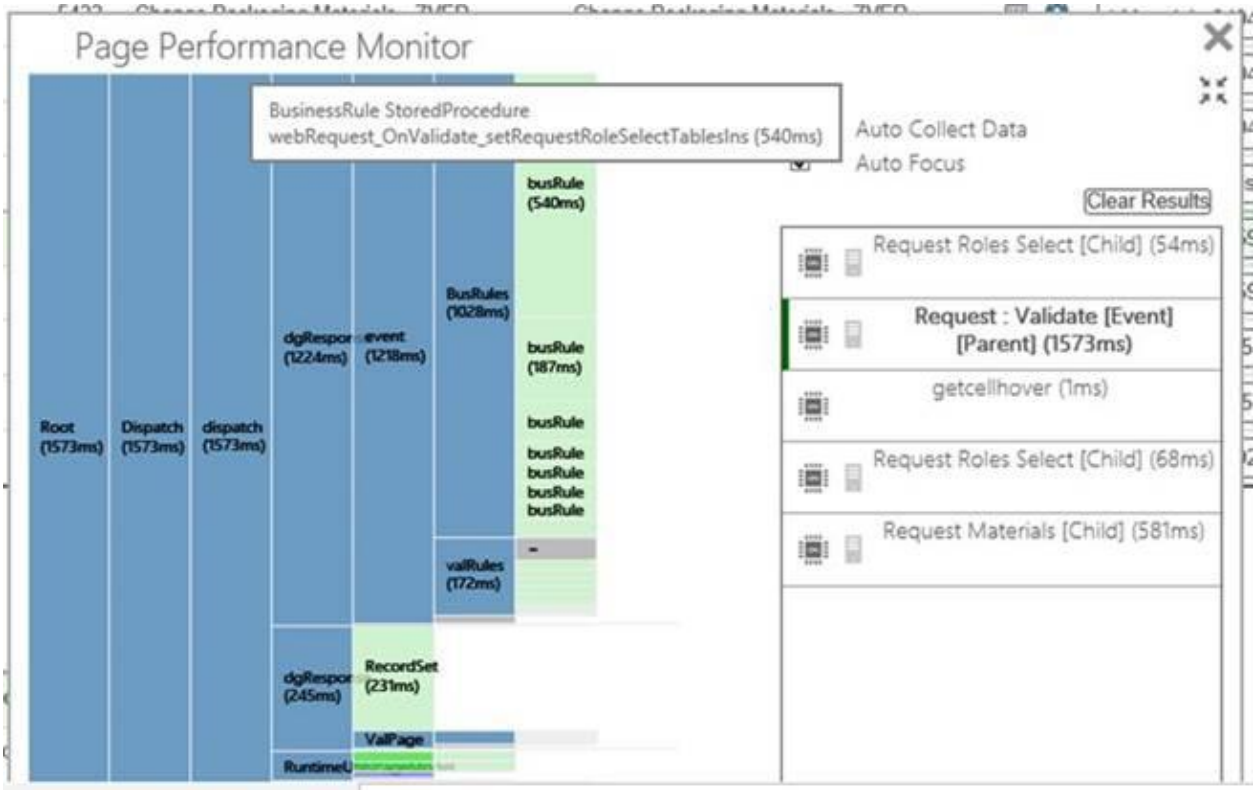
2.1 - DSP Page Performance Monitor (user approach)

The following instructions should be communicated to users, ideally through demos.

1. Press [CTRL + SHIFT + K] and this window will pop up:



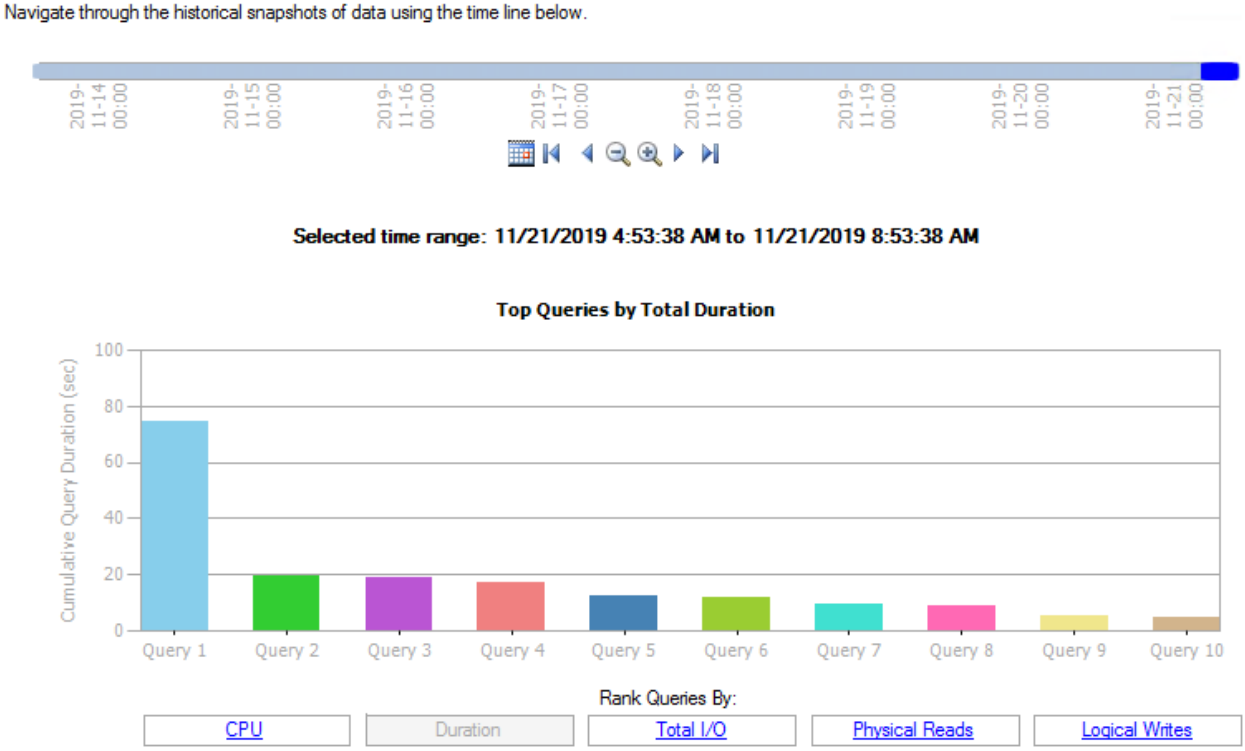
2. Then perform your action like Submit a Request, Finish a Role, open/refresh a page .
3. You now have the processing time in milliseconds.
4. If you find a green block that takes significant time, please hover the mouse on that and send a screenshot like this to whoever is collecting the performance information.



2.2 - SQL Server MDW Query Statistics (admin approach)

NOTE: Instructions on configuring MDW can be found in Appendix D.

1. Open the Query Statistics report in MDW.
2. Set the time interval so that it covers a period of high DSP user activity.
3. Rank the queries by Duration.

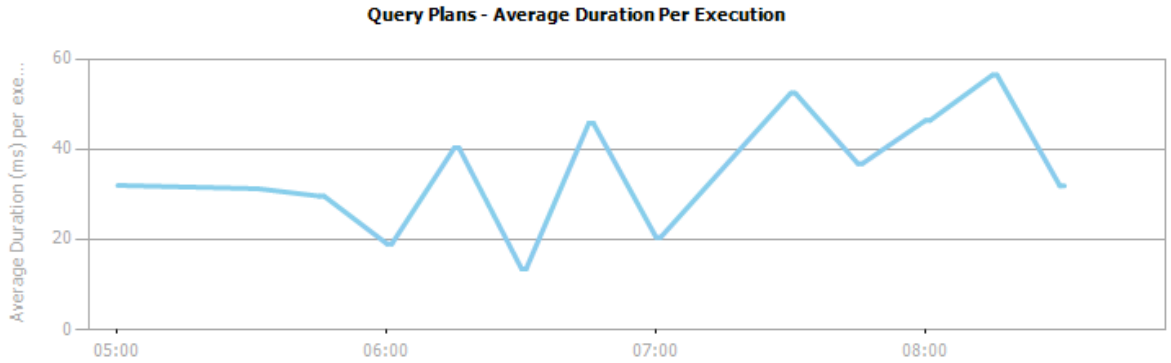


4. Click into the query with the greatest total duration and review the following information
    a. Average Duration (ms) per Execution – anything over 500 is a possible problem
    b. Average Executions per Minute – anything over 1 indicates that user activity is likely triggering the query
    c. DatabaseID – if available, it can identify the database when looked up via SELECT * FROM sys.databases
    d. Query – helps identify the objects involved in the query

Query Execution Statistics

Average CPU (ms) per Execution:	0.1	Total CPU (sec):	0.3
Average Duration (ms) per Execution:	35.0	Total Duration (sec):	74.6
Average Physical Reads per Execution:	0	Total Physical Reads:	0
Average Logical Writes per Execution:	0	Total Logical Writes:	3
		Total Executions:	2133
Average Executions per Min:	8		
Average CPU (ms) per Sec:	0	Query Plan Count:	1
Average Duration (ms) per Sec:	5		
Average Physical Reads per Sec:	0		
Average Logical Writes per Sec:	0		

View sampled waits for this query



5. Click on 'view sampled waits' to see if the query is being slowed by locking or some other kind of wait

3 - Speeding Up Slow SQL Objects

Syniti Software Support recommends 3 methods for speeding up SQL Object, though there are certainly more. The first method requires no knowledge of the object's intended function, while the other two are best performed by somebody familiar with the purpose of the object.

The first method is simply allowing SQL Server to recommend Table Indexes and then adding those that it recommends. This is a fully supported activity on both custom objects (created onsite) and delivered objects (shipped with the DSP). There is no risk of breaking the object's intended function or violating any license agreement.

The second method is to rewrite the SQL Object to run more quickly (simpler joins, fewer functions, etc). The SQL Server execution plan can be reviewed to identify areas of improvement. This method is often only possible when the original developer of the custom object (created onsite) is available to review the design of the object. If they are not available, then it is necessary to assign a new developer, making them responsible for learning the original object's function and then writing a new object to perform the same function (more quickly, of course).

WARNING: It is not appropriate to rewrite a delivered object (shipped with the DSP). If you're uncertain whether an object is custom or delivered, please open a ticket at support.syniti.com to check.

The third method is to remove the offending SQL Object from any places where it is registered in the DSP. This task must be performed by somebody who understands the object and where it would be found in the DSP page design registrations. The obvious risk here is that the object must first be deemed non-critical and the removal should first be tested in a development environment.

3.1 - Add Table Indexes

1. Take the entire query or just the portion from the SQL Object that you suspect is the slowest and plug it into a 'new query' in SSMS.
2. Enter sample values into the query if it contains any @variables. Make sure to use sample values that are realistic or else the query may complete very quickly on account of returning no results.
3. Press [Ctrl + M] on the keyboard to 'Include Actual Execution Plan'
4. Execute the query [F5]
5. After it completes, note the duration
6. Go to the {Execution plan} tab (to the right of {Results})
7. Right-click in the white space and select 'Missing Index Details...'
8. A 'new query' tab should appear in SSMS with a commented out query in it. Enter a name for the index in the query that appeared. A common naming format is IX_Table_Columns
9. Uncomment the query and execute it
10. Run the execution plan again to see if an additional table index is recommended
11. After it completes, note the duration

3.2 - Rewrite Object

WARNING: It is not appropriate to rewrite a delivered object (shipped with the DSP). If you're uncertain whether an object is custom or delivered, please open a ticket at support.syniti.com to check.

There are many ideas that can be followed to rewrite a SQL Object so that it runs faster. It is ideal to rely on the developer of the object to choose the best ideas. If the object at issue is delivered (shipped with the DSP), then you should consider Syniti the developer and open a ticket at support.syniti.com to request that we improve the performance of the delivered object. Here are some examples of query changes that can make a SQL Object run faster:

- Eliminate scalar functions from joins (where possible)
- Insert the results of complex views into staging tables and then join with the staging tables instead of the complex views
- Eliminate any uses of DISTINCT or GROUP BY (where possible)
- Eliminate operations that might cause locking when the object is run multiple times at once (or add the NOLOCK hint)

You can also use some tools to identify query improvements:

- The SQL Server Execution Plan will identify the most expensive operations in queries run by the object
- The MDW Query Statistics report can highlight locking

3.3 - Remove Object from DSP

WARNING: It is not appropriate to remove a delivered object (shipped with the DSP). If you're uncertain whether an object is custom or delivered, please open a ticket at support.syniti.com to check.

WARNING: It is important to only remove the slow SQL Object from the DSP, not from the SQL Server. If the object is discovered to be critical in some way, then it will be necessary to quickly add it back to DSP. Adding it back will not be quick if it has been removed from the SQL Server.

The idea here is that sometimes you ask what a slow object does and nobody believes it is important. Of course, you need to be sure you are asking the right people. Even then, the safest way to test eliminating the object is to unregister it from the DSP in a development environment. If the object is found to be non-critical, then it can simply be removed instead of being rewritten.

If the object appears to still be running in the development environment after it is unregistered, then there are probably some additional registrations that were missed. One trick to finding all the registrations is to rename the object in SQL Server. This should begin to produce error messages in the DSP wherever the object was still registered. The errors will go away after all the remaining registrations have been removed in the DSP.

Appendix A – SQL Server Database Settings

A.1 - Snapshot Isolation and Read Committed Snapshot

Syniti Software Support has found that the following list of databases often experience deadlocks in a Master Data Management (formerly dspConduct) implementation.

- CranSoft
- databases used by the content webapps, such as:
  - dgeCustomer
  - dgeMaterial
  - dgeVendor
- DGE
- databases that hold target request data, such as:
  - dgSAP

The following query can be executed when the SQL Server is down for maintenance or any time when there is almost no activity on the SQL Server (from DSP or SQL logins). It sets the [Allow Snapshot Isolation] and [Is Read Committed Snapshot On] database flags to TRUE, which helps to significantly reduce deadlocking.

```
USE [master]
ALTER DATABASE [CranSoft] SET ALLOW_SNAPSHOT_ISOLATION ON
ALTER DATABASE [CranSoft] SET READ_COMMITTED_SNAPSHOT ON WITH NO_WAIT

ALTER DATABASE [dgeCustomer] SET ALLOW_SNAPSHOT_ISOLATION ON
ALTER DATABASE [dgeCustomer] SET READ_COMMITTED_SNAPSHOT ON WITH NO_WAIT

ALTER DATABASE [dgeMaterial] SET ALLOW_SNAPSHOT_ISOLATION ON
ALTER DATABASE [dgeMaterial] SET READ_COMMITTED_SNAPSHOT ON WITH NO_WAIT
```

```
ALTER DATABASE [dgeVendor] SET ALLOW_SNAPSHOT_ISOLATION ON
ALTER DATABASE [dgeVendor] SET READ_COMMITTED_SNAPSHOT ON WITH NO_WAIT

ALTER DATABASE [DGE] SET ALLOW_SNAPSHOT_ISOLATION ON
ALTER DATABASE [DGE] SET READ_COMMITTED_SNAPSHOT ON WITH NO_WAIT

ALTER DATABASE [dgSAP] SET ALLOW_SNAPSHOT_ISOLATION ON
ALTER DATABASE [dgSAP] SET READ_COMMITTED_SNAPSHOT ON WITH NO_WAIT
```

A.2 - Compatibility Level

Over time, after multiple upgrades of the SQL Server, there is a possibility of having databases with different Compatibility Levels. There is quite a bit of documentation online regarding the changes between the different Database Compatibility levels. One of the changes is in regard to the way the query plans are created between the different compatibility levels. Some of the queries that run in the DSP, both delivered and custom, may run slower depending on the compatibility levels of the databases.

In general, the idea is that newer (higher) Compatibility Levels create more efficient query plans. However, that is not always the case. Some queries do perform better under older (lower) Compatibility Levels.

In some cases, Syniti Software Support has seen queries run incredibly slow when they involve objects from multiple databases and those databases have different Compatibility Levels. For that reason, it is our recommendation that customers ensure all of their SQL Server databases (in the DSP SQL Server, at least) use the same Compatibility Level.

Appendix B – SQL Server Maintenance Plans

B.1 - Rebuilding Indexes and Updating Statistics

Syniti Software Support recommends a DBA create a SQL Server Maintenance Plan that rebuilds indexes and updates statistics on the following critical databases every 12 hours in a Master Data Management (formerly dspConduct) implementation.

- CranPort
- CranSoft
- databases used by the content webapps, such as:
  - dgeCustomer
  - dgeMaterial
  - dgeVendor
- DGE
- databases that hold target request data, such as:
  - dgSAP
- DSPCommon

Tables Indexes quickly become fragmented and SQL Statistics quickly become old in these critical databases where a high volume of activity occurs. The 12-hour maintenance plan helps reduce the negative effects of this.

B.2 - Timing Backups

In general, Syniti Software Support recommends an awareness of backup timings, so that the critical databases (listed in B.1) are never being backed up during the periods of highest user activity or during periods of large data extracts. Complete detail on how often to back up the DSP's SQL Server databases can be found in the "Recovery Model and Backup Recommendations" section of this Syniti Software Support Knowledge Base article:

https://support.syniti.com/hc/en-us/articles/115000353327

Appendix C - SQL Server Performance Tuning Queries

Syniti Software Support recommends some general performance-tuning queries that are also commonly used in the SQL Server DBA community. While all the queries are useful, the 2 queries we use the most often are:

- Find "Missing" Indexes for the entire instance of SQL Server
- Look for possible bad indexes inside the entire current database

The queries can all be downloaded from this Syniti Software Support Knowledge Base article:

https://support.syniti.com/hc/en-us/articles/206450467

Appendix D – Configuring SQL Server Management Data Warehouse (MDW)

Instructions are available in this Syniti Software Support Knowledge Base article:

https://support.syniti.com/hc/en-us/articles/360041323373