**ALTER Procedures When Using Query Store**

* [Erin Stellato](https://www.sqlskills.com/blogs/erin/author/erin/)
* [October 8, 2020](https://www.sqlskills.com/blogs/erin/2020/10/08/)

FROM: <https://www.sqlskills.com/blogs/erin/alter-procedures-when-using-query-store/?utm_source=ssc&utm_medium=pubemail>

When I talk about Plan Forcing I always discuss how users should ALTER procedures when using Query Store, and not use DROP and CREATE. This is valid beyond Plan Forcing cases; it’s a best practice I recommend *however* you are using Query Store. Every query stored in Query Store has an object\_id associated with it, which ties it back to its object (stored procedure, function, etc.). This is critical not just for plan forcing, but also when you want to look at historical performance for a query *after* a change to the object.

**Setup**

Within the WideWorldImporters database, create a stored procedure with multiple statements:

USE [WideWorldImporters];  
GO

CREATE OR ALTER PROCEDURE [Sales].[usp\_GetCustomerDetail]  
@CustomerName NVARCHAR(100)  
AS

CREATE TABLE #CustomerList (  
[RowID] INT IDENTITY (1,1),  
[CustomerID] INT,  
[CustomerName] NVARCHAR (100)  
);

INSERT INTO #CustomerList (  
[CustomerID],  
[CustomerName]  
)  
SELECT  
[CustomerID],  
[Customername]  
FROM [Sales].[Customers]  
WHERE [CustomerName] LIKE @CustomerName  
UNION  
SELECT  
[CustomerID],  
[CustomerName]  
FROM [Sales].[Customers\_Archive]  
WHERE [CustomerName] LIKE @CustomerName;

SELECT  
[o].[CustomerID],  
[o].[OrderID],  
[il].[InvoiceLineID],  
[o].[OrderDate],  
[i].[InvoiceDate],  
[ol].[StockItemID],  
[ol].[Quantity],  
[ol].[UnitPrice],  
[il].[LineProfit]  
INTO #CustomerOrders  
FROM [Sales].[Orders] [o]  
INNER JOIN [Sales].[OrderLines] [ol]  
ON [o].[OrderID] = [ol].[OrderID]  
INNER JOIN [Sales].[Invoices] [i]  
ON [o].[OrderID] = [i].[OrderID]  
INNER JOIN [Sales].[InvoiceLines] [il]  
ON [i].[InvoiceID] = [il].[InvoiceID]  
AND [il].[StockItemID] = [ol].[StockItemID]  
AND [il].[Quantity] = [ol].[Quantity]  
AND [il].[UnitPrice] = [ol].[UnitPrice]  
WHERE [o].[CustomerID] IN (SELECT [CustomerID] FROM #CustomerList);

SELECT  
[cl].[CustomerName],  
[si].[StockItemName],  
SUM([co].[Quantity]) AS [QtyPurchased],  
SUM([co].[Quantity]\*[co].[UnitPrice]) AS [TotalCost],  
[co].[LineProfit],  
[co].[OrderDate],  
DATEDIFF(DAY,[co].[OrderDate],[co].[InvoiceDate]) AS [DaystoInvoice]  
FROM #CustomerOrders [co]  
INNER JOIN #CustomerList [cl]  
ON [co].[CustomerID] = [cl].[CustomerID]  
INNER JOIN [Warehouse].[StockItems] [si]  
ON [co].[StockItemID] = [si].[StockItemID]  
GROUP BY [cl].[CustomerName], [si].[StockItemName],[co].[InvoiceLineID],  
[co].[LineProfit], [co].[OrderDate], DATEDIFF(DAY,[co].[OrderDate],[co].[InvoiceDate])  
ORDER BY [co].[OrderDate];

GO

Enable Query Store with most of the default settings; this is a demo, I’m not worried about configuration. This is **not** what I would do for production, see [this post](https://www.sqlskills.com/blogs/erin/query-store-settings/) for more details on settings.. I also would not remove all data from Query Store in production, but again…this is a demo.

ALTER DATABASE [WideWorldImporters]  
SET QUERY\_STORE = ON;  
GO

ALTER DATABASE [WideWorldImporters]  
SET QUERY\_STORE (  
OPERATION\_MODE = READ\_WRITE,  
INTERVAL\_LENGTH\_MINUTES = 10  
);  
GO

/\*  
Do not run in a Production database unless you want  
to remove all Query Store data  
\*/

ALTER DATABASE [WideWorldImporters]  
SET QUERY\_STORE CLEAR;  
GO

Then, execute the stored procedure a few times:

EXEC [Sales].[usp\_GetCustomerDetail] N'Alvin Bollinger';  
GO 10

EXEC [Sales].[usp\_GetCustomerDetail] N'Tami Braggs';  
GO 10

EXEC [Sales].[usp\_GetCustomerDetail] N'Logan Dixon';  
GO 10

EXEC [Sales].[usp\_GetCustomerDetail] N'Tara Kotadia';  
GO 10

**Understanding the Data**

It’s critical to understand what data exists in Query Store for this stored procedure. The query below interrogates the Query Store views to list all the statements in this stored procedure (note that there is no way to see this with any of the default reports):

SELECT  
[qsq].[query\_text\_id],  
[qsq].[query\_id],  
[qsq].[object\_id],  
[qsq].[context\_settings\_id],  
[qst].[query\_sql\_text]  
FROM [sys].[query\_store\_query] [qsq]  
JOIN [sys].[query\_store\_query\_text] [qst]  
ON [qsq].[query\_text\_id] = [qst].[query\_text\_id]  
WHERE [qsq].[object\_id] = OBJECT\_ID(N'Sales.usp\_GetCustomerDetail');  
GO

Query Information in QS

There are multiple columns that, when combined, create a unique query in Query Store. Three of those columns are query\_text\_id, context\_settings\_id, and object\_id.

Drop and recreate the stored procedure, using:

DROP PROCEDURE IF EXISTS [Sales].[usp\_GetCustomerDetail];  
GO

Then execute the **exact same** CREATE OR REPLACE code from the previous section, and run the stored procedure again. Check the data in Query Store with the previous query:

Query Information in QS after DROP AND CREATE

Notice that the query\_id and object\_id have changed. The query\_text\_id is the same for all three queries (as is the context\_settings\_id), but the object\_id changed – because we DROPPED the stored procedure, rather than ALTERing it – and therefore the query\_id *also* changed.

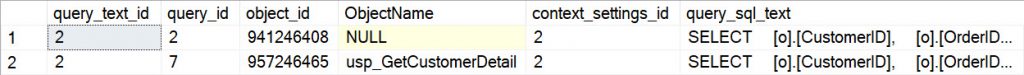
The good news is that the original queries (and their plans and runtime statistics) are still in Query Store; they’re just harder to find:

SELECT  
[qsq].[query\_text\_id],  
[qsq].[query\_id],  
[qsq].[object\_id],  
OBJECT\_NAME([qsq].[object\_id]) AS [ObjectName],  
[qsq].[context\_settings\_id],  
[qst].[query\_sql\_text]  
FROM [sys].[query\_store\_query] [qsq]  
JOIN [sys].[query\_store\_query\_text] [qst]  
ON [qsq].[query\_text\_id] = [qst].[query\_text\_id]  
WHERE [qsq].[object\_id] > 0;  
GO

All queries in QS after DROP AND CREATE

Unless the object\_id for the queries is saved/noted somewhere *before* the procedure is dropped and created, the only way to find the original queries is to query for object\_id > 0 (which likely returns *a lot* of data) or search by the query\_sql\_text (which is not going to be fast):

SELECT  
[qsq].[query\_text\_id],  
[qsq].[query\_id],  
[qsq].[object\_id],  
OBJECT\_NAME([qsq].[object\_id]) AS [ObjectName],  
[qsq].[context\_settings\_id],  
[qst].[query\_sql\_text]  
FROM [sys].[query\_store\_query] [qsq]  
JOIN [sys].[query\_store\_query\_text] [qst]  
ON [qsq].[query\_text\_id] = [qst].[query\_text\_id]  
WHERE [qst].[query\_sql\_text] LIKE '%INTO #CustomerOrders%';  
GO

Queries returned based on a search of query\_sql\_text

**Action Item: Use ALTER**

If you’re using Query Store, or planning to use it, you really need to ALTER procedures, rather than running DROP and CREATE statements. Note: You need to use ALTER PROCEDURE specifically. If you use CREATE OR ALTER procedure, while the object\_id stays the same…a new query\_id shows up in Query Store (even with no change to the query text). This is something I’m following up on – I don’t think this is expected behavior.





Bottom of Form

**Other articles**

[**ALTER Procedures When Using Query Store**](https://www.sqlskills.com/blogs/erin/alter-procedures-when-using-query-store/)

When I talk about Plan Forcing I always discuss how users should ALTER procedures when using Query Store, and not use DROP and CREATE. This

[Explore](https://www.sqlskills.com/blogs/erin/alter-procedures-when-using-query-store/)

[**The security\_error\_ring\_buffer\_recorded event and why you don’t need it**](https://www.sqlskills.com/blogs/erin/the-security_error_ring_buffer_recorded-event-and-why-you-dont-need-it/)

  I recently ran through a health audit for a customer, and in reviewing their system\_health output I was reminded of the unnecessary security\_error\_ring\_buffer\_recorded event.

[Explore](https://www.sqlskills.com/blogs/erin/the-security_error_ring_buffer_recorded-event-and-why-you-dont-need-it/)

[**Capturing DBCC CHECKDB Output**](https://www.sqlskills.com/blogs/erin/capturing-dbcc-checkdb-output/)

If you haven’t experienced database corruption in your SQL Server career and had to pore over DBCC CHECKDB output, you’re a lucky soul.  If you

[Explore](https://www.sqlskills.com/blogs/erin/capturing-dbcc-checkdb-output/)

[**Working From Home with Kids: 7 Weeks In**](https://www.sqlskills.com/blogs/erin/working-from-home-with-kids-7-weeks-in/)

I’ve been meaning to write a follow up post to my original to discuss what life is REALLY like during quarantine.  I re-read my original

[Explore](https://www.sqlskills.com/blogs/erin/working-from-home-with-kids-7-weeks-in/)

[**Troubleshooting Performance in Azure SQL**](https://www.sqlskills.com/blogs/erin/troubleshooting-performance-in-azure-sql-database/)

At some point, whether you’re a DBA, developer, or application administrator, you’re going to find yourself troubleshooting performance in Azure.  Within the Azure Portal you

[Explore](https://www.sqlskills.com/blogs/erin/troubleshooting-performance-in-azure-sql-database/)

[**Using relog: Creating a smaller file**](https://www.sqlskills.com/blogs/erin/using-relog-creating-a-smaller-file/)