Conquering the Monster Proc: How to Combat Legacy Code

Deborah Melkin

She\Her

Data Engineer

Advisor360


Microsoft
AMD
PURESTORAGE®
redgate
rev.io
three tree tech
NORTH BANK INNOVATIONS
INTERJECT
OMNIDATA
SQL Saturday
2024
Sponsors

Deborah Melkin

She\Her

Data Engineer

Advisor360

@dgmelkin

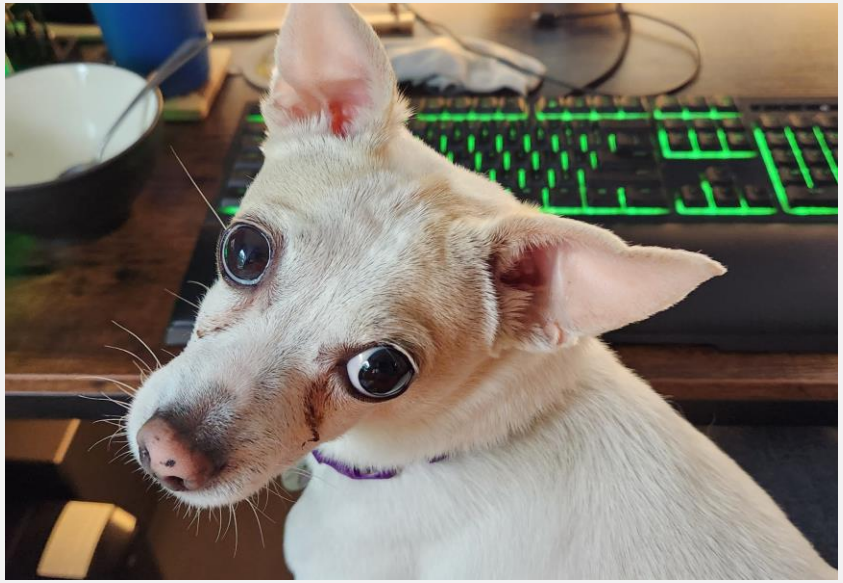
DebtheDBA.wordpress.com

dgmelkin@gmail.com

- 20+ years as a DBA
- Regular speaker at User Groups, SQL Saturdays, etc.
- Data Platform Women in Tech (WIT) Virtual UG, Co-leader
- WITspiration, co-founder
- Speaker Idol Winner 2019
- #Redgate100 (2022)
- Microsoft MVP. Data Platform

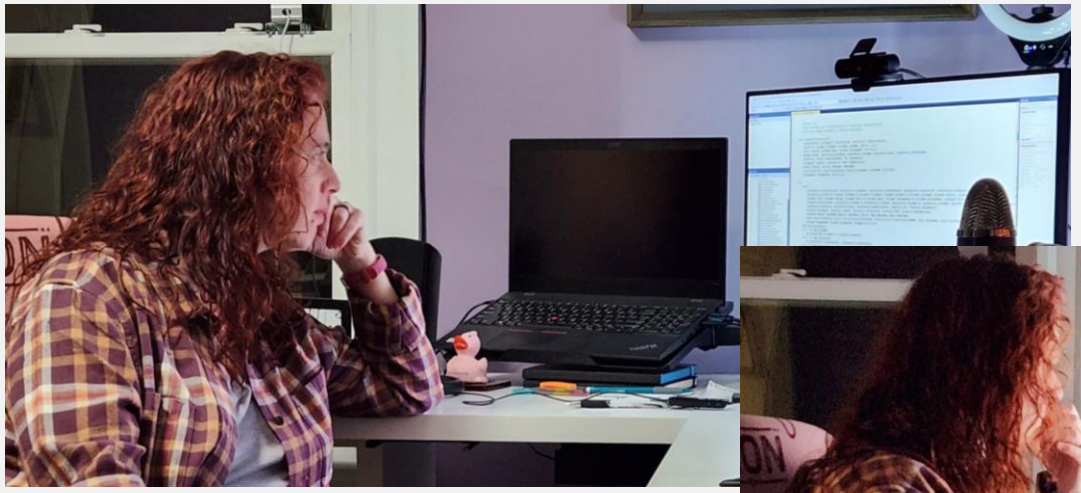
Story Time

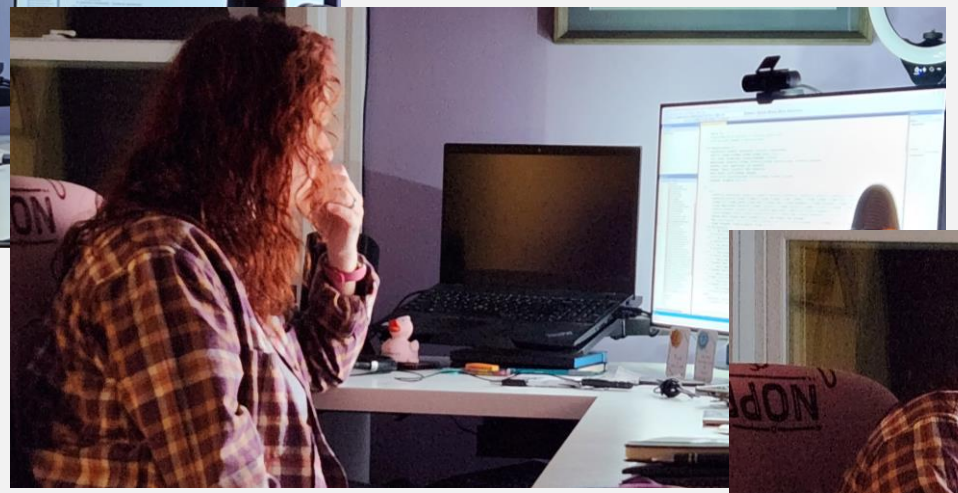
A DBA and a Developer walked into my office....

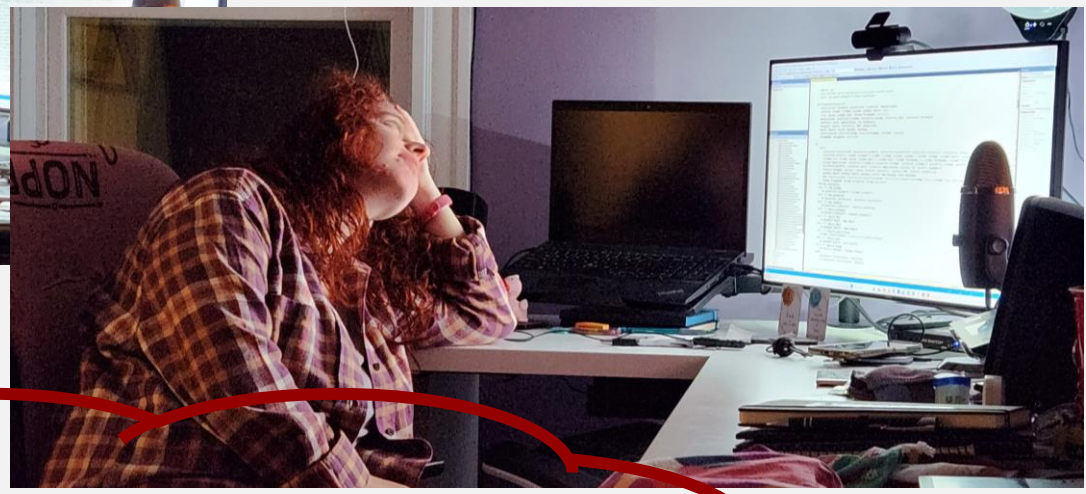


Congratulations!
You know that proc everyone hates?
You finally get to fix it!

Woo hoo!!!!







Oh wait, now what do I do?

Does this sound familiar?

Our goal is to be smarter about how to attack these legacy code.

Here’s the code we’re going through…

demo.sp_NightlyProcessingForReporting

History of the Procedure

The Programmers

- Originally created in “2006”
- Modified by multiple programmers over the years
- Updates the demo.NightlySalesStaging table with the sales history for specified dates
- Creates the data needed for about 9 tables used by reports based on all of the data in staging table

Here’s how we’re going to attack it…

Two Plans of Attack to Make Changes

- Technical ← **Why**
- Business ← **Why Not**

Technical Plan of Attack

Technical Plan of Attack

- Gather requirements
- “Reverse Engineer” the code
- Gather performance information
- Determine the fix
- Design test plans

Gather Requirements

- What problem are you trying to solve?
- What is the code supposed to do?
- Is the code currently doing what it's supposed to do?

Gather Requirements – Q1 Answered

- What problem are you trying to solve?
  - *I've been told the performance is bad*
  - *It has some very involved logic that's hard to understand*

Gather Requirements – Q2 Answered

- What is the code supposed to do?
  - *Load tables used by reports after a nightly staging process*
  - *Note hidden inside*

```
173  /*************************************************************************
174  * Populate Reporting Tables
175  *
176  * 2006-10: Jess
177  * Each SSRS report will read from a vw_rpt_[name] view that will be a
178  * SELECT * from the respective rpt_[name] table
179  *
180  * IMPORTANT - All reports should only ever query demo.NightlySalesStaging
181  *************************************************************************/
```

Gather Requirements – Q3 Answered

- Is the code currently doing what it's supposed to do?
  - *Code does update report tables*
  - *Not all query use only the demo·NightlySalesStaging table*

- *Any other technical questions that I've forgotten to ask?*

Reverse Engineer the code

- What does each step do?
- Initial thoughts
- Review the code multiple times

Reverse Engineer - Handwritten Notes

Full proc comments :

- Proc starts with sp_*

Initial comments reading through the proc:

1. Takes information from sales history tables and inserts new data into the NightlySalesStaging table joining on the SalesHistoryID
   a. Updates and Inserts only
   b. Potential Performance issues:
      i. Uses MERGE and CTE

2. Drops table demo.rpt_SoldModels_by_Age
   a. DROPS TABLE!!!!
   b. Potential Performance issues:
      i. Effect on recompiles for the database since the stats are gone and have to be recreated

3. Creates demo.rpt_Sold_Models_by_Age from a SELECT from demo.NightlySalesStaging
   a. Additional Research points:
      i. How big is this table?
      ii. What are the indexes on demo.NightlySalesStaging
         1) 5 indexes, 3 of them are variations of each other
      iii. There's a GROUP BY that includes converts of DateReceived for formatting purposes.
   b. Potential Performance Issues:
      i. SELECT INTO to create a table. Any potential locking issues
      ii. What does this do for the performance of the SSRS report - the table is dropped and r

7. Same pattern for demo.rpt_salessummarypermonth
   a. NOTE - different developer (Sebastian), naming convention
   b. NOLOCKS!!!! :(
   c. Uses a temp table to hold data from sales person and sales history
      i. Still uses SELECT INTO *smh*
   d. Joins the temp table toa view to do a SELECT INTO the report table.
      i. Temp table doesn't have any indexes
      ii. View doesn't have a schema name so SQL Prompt is no help
         1) Confirmed it's in the dbo schema.

...Or Annotate Script

C:\GitRepo\ConqueringtheMonsterProc\Scripts\demo.sp_NightlyProcessingForReporting.sql - TortoiseGitMerge

C:\GitRepo\ConqueringtheMonsterProc\Scripts\demo.sp_NightlyProcessingForReporting - Annotated.sql - C:\GitRepo\ConqueringtheMonsterProc\Scripts\demo.sp_NightlyProcessingForReporting.sql

```
            i. Effect on recompiles for the database since the stats are gone and have to be recreated

    */
172     DROP TABLE demo.rpt_Sold_Models_by_Age;

    /* DGM Pass 1:
        3. Creates demo.rpt_Sold_Models_by_Age from a SELECT from demo.NightlySalesStaging
            a. Additional Research points:
                i. How big is this table?
                ii. What are the indexes on demo.NightlySalesStaging
                    1) 5 indexes, 3 of them are variations of each other
                iii. There's a GROUP BY that includes converts of DateReceived for formatting purposes. Why formatting here?
            b. Potential Performance Issues:
                i. SELECT INTO to create a table. Any potential locking issues
                ii. What does this do for the performance of the SSRS report - the table is dropped and recreated? What if someone made the report view an indexed view
173
    */
174     SELECT MakeName, ModelName,
175         CAST(CAST(YEAR(DateReceived) AS VARCHAR(10)) + '-' + CAST(MONTH(DateReceived) AS VARCHAR(10)) + '-01' AS DATE) AS MonthReceived,
176         SUM(TrueCost) AS TotalTrueCost,
177         SUM(MSRP) AS TotalMSRP,
178         SUM(InvoicePrice) AS TotalInvoicePrice,
179         SUM(SellPrice) AS TotalSoldPrice,
180         COUNT(1) AS MyCount
181     INTO demo.rpt_Sold_Models_by_Age
182     FROM demo.NightlySalesStaging
183     GROUP BY MakeName, ModelName, CAST(CAST(YEAR(DateReceived) AS VARCHAR(10)) + '-' + CAST(MONTH(DateReceived) AS VARCHAR(10)) + '-01' AS DATE)
184     ORDER BY MakeName, ModelName, CAST(CAST(YEAR(DateReceived) AS VARCHAR(10)) + '-' + CAST(MONTH(DateReceived) AS VARCHAR(10)) + '-01' AS DATE);
185
186     PRINT '-----'
187     PRINT '--- rpt_Sold_Models_by_Age complete'
188
189     -----
190     -- Profit By Model
191     -- 2008-10: Jess

    /* DGM Pass 1:
        4. Repeats similar process for demo.rpt_Profit_By_Model where it drops the table then recreates it using SELECT INTO
            a. The use of count(1) is a little different. I'm used to seeing Count(some not null column) but thinking through logic, this should work. Confirm there's no errors. Possibly learn something new
            b. Potential Performance Issues: see above
            c. Full stored proc observation:
                i. Performance issues of the views that read these.
                ii. No error handling
                iii. For these patterns of dropping the table and recreating them, what does this mean for the data types on the new table that's created. Does that create problems for the SSRS view?


    */
192     DROP TABLE demo.rpt_Profit_By_Model;
193
194     SELECT MakeName, ModelName,
195         SUM(SellPrice) - SUM(TrueCost) AS TotalProfit,
196         (SUM(SellPrice) - SUM(TrueCost)) / COUNT(1) AS AvgProfit,
197         COUNT(1) AS SoldCount
198     INTO demo.rpt_Profit_By_Model
199     FROM demo.NightlySalesStaging
200     GROUP BY MakeName, ModelName
```

SQL Prompt – Summarize Script

SQL Prompt - Summarize Script

Script name: demo.sp_NightlyProcessingForReporting - Annotated.sql

redgate

Summary

The actions that this script performs in the order in which they occur.

```
CREATE PROCEDURE demo.sp_NightlyProcessingForReporting
WITH SalesHistorySource_CTE
PRINT '-----'
PRINT '-- load complete'
DROP TABLE demo.rpt_Sold_Models_by_Age
SELECT FROM demo.NightlySalesStaging
PRINT '-----'
PRINT '-- rpt_Sold_Models_by_Age complete'
DROP TABLE demo.rpt_Profit_By_Model
SELECT FROM demo.NightlySalesStaging
PRINT '-----'
PRINT '-- rpt_Profit_By_Model complete'
DROP TABLE demo.rpt_Avg_Profit_Classification_Type
SELECT FROM demo.nightlysalesstaging
OPEN RSEXE
FETCH NEXT FROM rsExe
WHILE @@FETCH_STATUS = 0
CLOSE RSEXE
PRINT '-----'
```

Select a keyword to highlight the keyword or the corresponding block of code in the script editor.
You can resize this dialog box if required.

Help | Refresh | Close

Tools to help identify potential issues

- Code Analysis
  - ScriptDOM
  - SQLPrompt Code Analysis
  - Visual Studio – Analyze
  - Linters (SQLFluff, etc.)
- Look for Dependencies
  - sys.sql_dependencies
  - sp_helpExpandView (Not just for views!)

SQL Prompt - Code Analysis

List of Code Analysis Issues

Path: demo.sp_NightlyProcessingForReporting.sql

Discovered issues: 53

Refresh Export

Group by: Issue

Code	Issue	Path	Line
PE015	No FETCH FIRST/LAST/PRIOR, but cursor is not declared as forward only (1)		
BP015	Scope of cursor (LOCAL/GLOBAL) is not specified (1)		
PE018	Cursor is not declared as readonly (1)		
PE010	Interleaving DDL and DML in stored procedure/trigge		
EI024	Stored procedure name starts with sp_ (1)		
ST010	Use alias for all table sources (17)		
PE006	TABLE HINT is used (4)		
PE002	Schema name for table or view is not specified (1)		
MI003	Unqualified column name (2)		
PE003	Creation of table by SELECT INTO statement (11)		
ST006	Old-style TOP clause is used (3)		
PE020	INSERT INTO table with ORDER BY (2)		
BP004	INSERT without column list (3)		
BP012	CASE without ELSE (2)		
BP005	Asterisk in select list (2)		
PE019	Consider using EXISTS instead of IN (1)		

Interleaving DDL and DML in stored proc/trigger

TABLE HINT is used

Creation of table by SELECT INTO statement

Consider using EXISTS instead of IN

Code Analysis – Visual Studio

```
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(465,3): Warning:  : SR0001 : Microsoft.Rules.Data : The shape of the result set produced by a SELECT * statement will change if the underlying table or view structu
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(510,8): Warning:  : SR0001 : Microsoft.Rules.Data : The shape of the result set produced by a SELECT * statement will change if the underlying table or view structu
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(72,4): Warning:  : SR0007 : Microsoft.Rules.Data : Nullable columns can cause final results to be evaluated as NULL for the predicate.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(73,8): Warning:  : SR0007 : Microsoft.Rules.Data : Nullable columns can cause final results to be evaluated as NULL for the predicate.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(245,9): Warning:  : SR0007 : Microsoft.Rules.Data : Nullable columns can cause final results to be evaluated as NULL for the predicate.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(301,9): Warning:  : SR0007 : Microsoft.Rules.Data : Nullable columns can cause final results to be evaluated as NULL for the predicate.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(326,7): Warning:  : SR0007 : Microsoft.Rules.Data : Nullable columns can cause final results to be evaluated as NULL for the predicate.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(475,8): Warning:  : SR0007 : Microsoft.Rules.Data : Nullable columns can cause final results to be evaluated as NULL for the predicate.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(439,13): Warning:  : SR0007 : Microsoft.Rules.Data : Nullable columns can cause final results to be evaluated as NULL for the predicate.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(439,29): Warning:  : SR0007 : Microsoft.Rules.Data : Nullable columns can cause final results to be evaluated as NULL for the predicate.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(446,13): Warning:  : SR0007 : Microsoft.Rules.Data : Nullable columns can cause final results to be evaluated as NULL for the predicate.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(446,29): Warning:  : SR0007 : Microsoft.Rules.Data : Nullable columns can cause final results to be evaluated as NULL for the predicate.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(453,13): Warning:  : SR0007 : Microsoft.Rules.Data : Nullable columns can cause final results to be evaluated as NULL for the predicate.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(453,29): Warning:  : SR0007 : Microsoft.Rules.Data : Nullable columns can cause final results to be evaluated as NULL for the predicate.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(460,9): Warning:  : SR0007 : Microsoft.Rules.Data : Nullable columns can cause final results to be evaluated as NULL for the predicate.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(518,9): Warning:  : SR0007 : Microsoft.Rules.Data : Nullable columns can cause final results to be evaluated as NULL for the predicate.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(243,4): Warning:  : SR0014 : Microsoft.Rules.Data : Data loss might occur when casting from Money to Int.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(326,34): Warning:  : SR0014 : Microsoft.Rules.Data : Data loss might occur when casting from Date to DateTime.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(326,49): Warning:  : SR0014 : Microsoft.Rules.Data : Data loss might occur when casting from Date to DateTime.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(390,60): Warning:  : SR0014 : Microsoft.Rules.Data : Data loss might occur when casting from Money to Decimal(3, 2).
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(475,35): Warning:  : SR0014 : Microsoft.Rules.Data : Data loss might occur when casting from Date to DateTime.
demo\Stored Procedures\sp_NightlyProcessingForReporting.sql(475,50): Warning:  : SR0014 : Microsoft.Rules.Data : Data loss might occur when casting from Date to DateTime.
```


```
: Microsoft.Rules.Data : Data loss might occur when casting from Money to Int.
4 : Microsoft.Rules.Data : Data loss might occur when casting from Date to DateTime.
4 : Microsoft.Rules.Data : Data loss might occur when casting from Date to DateTime.
4 : Microsoft.Rules.Data : Data loss might occur when casting from Money to Decimal(3, 2).
```


```
demo\Stored Procedures\sp_SearchAllSoldInventory.sql(2,25): Warning:  : SR0016 : Microsoft.Rules.Data : Stored procedure(sp_SearchAllSoldInventory) includes sp_ prefix in its name.
```

The results are saved in \bin\Debug\<db project>.StaticCodeAnalysis.Results.xml

Dependencies - sp_helpExpandView

- Community tool by Andy Yun

```sql
/* run sp_helpExpandView for the main proc. Includes information for the related stored procedure */
EXEC sp_helpExpandView @ViewName = '[demo].[sp_NightlyProcessingForReporting]', @OutputFormat = 'horizontal'
```

sults Messages

BaseObject_FullName	Lvl_1	Obj_1	Typ_1	Lvl_2	Obj_2	Typ_2	Lvl_3	Obj_3	Typ_3	Lvl_4	Obj_4	Typ_4
demo.sp_NightlyProcessingForReporting	1	NULL	NULL	2			3			4		
demo.sp_NightlyProcessingForReporting	1	dbo.Customer	U	2			3			4		
demo.sp_NightlyProcessingForReporting	1	dbo.Inventory	U	2			3			4		
demo.sp_NightlyProcessingForReporting	1	dbo.SalesHistory	U	2			3			4		
demo.sp_NightlyProcessingForReporting	1	dbo.SalesPerson	U	2			3			4		
demo.sp_NightlyProcessingForReporting	1	dbo.vw_salesperson_annualnumofsales	V	2	dbo.vw_SalesPerson_SalesPerMonth	V	3	dbo.BaseVw_SalesHistory	V	4	dbo.SalesHistory	U
demo.sp_NightlyProcessingForReporting	1	dbo.vw_salesperson_annualnumofsales	V	2	dbo.vw_SalesPerson_SalesPerMonth	V	3	dbo.BaseVw_SalesPerson	V	4	dbo.SalesPerson	U
demo.sp_NightlyProcessingForReporting	1	demo.NightlySalesStaging	U	2			3			4		
demo.sp_NightlyProcessingForReporting	1	demo.rpt_Avg_Profit_Classification_Type	U	2			3			4		
demo.sp_NightlyProcessingForReporting	1	demo.rpt_Avg_Vehicle_Age_Classification_Type	U	2			3			4		
demo.sp_NightlyProcessingForReporting	1	demo.sp_searchallsoldinventory	P	2	dbo.Inventory	U	3			4		
demo.sp_NightlyProcessingForReporting	1	demo.sp_searchallsoldinventory	P	2	dbo.SalesHistory	U	3			4		
demo.sp_NightlyProcessingForReporting	1	demo.sp_searchallsoldinventory	P	2	Vehicle.BaseModel	U	3			4		
demo.sp_NightlyProcessingForReporting	1	demo.sp_searchallsoldinventory	P	2	Vehicle.Color	U	3			4		
demo.sp_NightlyProcessingForReporting	1	demo.sp_searchallsoldinventory	P	2	Vehicle.Make	U	3			4		
demo.sp_NightlyProcessingForReporting	1	demo.sp_searchallsoldinventory	P	2	Vehicle.Model	U	3			4		
demo.sp_NightlyProcessingForReporting	1	demo.sp_searchallsoldinventory	P	2	Vehicle.Package	U	3			4		
demo.sp_NightlyProcessingForReporting	1	demo.udf_CalculateNetProfit	FN	2	demo.NightlySalesStaging	U	3			4		
demo.sp_NightlyProcessingForReporting	1	Vehicle.BaseModel	U	2			3			4		
demo.sp_NightlyProcessingForReporting	1	Vehicle.Classification	U	2			3			4		
demo.sp_NightlyProcessingForReporting	1	Vehicle.Color	U	2			3			4		
demo.sp_NightlyProcessingForReporting	1	Vehicle.Make	U	2			3			4		
demo.sp_NightlyProcessingForReporting	1	Vehicle.Model	U	2			3			4		
demo.sp_NightlyProcessingForReporting	1	Vehicle.Package	U	2			3			4		

Gather Performance Information

- sp_WhoIsActive
- Query Store
- I/O information
- Extended Events
  - Run time actions
  - Other activity

Gather Performance Information (cont'd)

- Table stats for objects used
  - Number of rows
    - MAX & AVG over clients data
    - Diff between client and dev data
  - Constraints\Indexes
  - Triggers
- 3rd Party Monitoring

Determine the Fix

- Do the fixes have to be done together or separately?
- How much of the code needs to be changed?
- Are there different options for each fix?
- Which require changes outside of the proc code?
  - Table\Index changes
  - Application code changes

Design Test Plans

- Types of Tests:
  - Performance impact
  - Unit testing
  - Bug fixes
  - Regression testing
- Are there environments to support testing?

Compile the Results of the Technical Analysis

Line #	Issue	Bug	Perf Concern	How to fix	How to test
79	comment says Merge statement has insert, update and delete; no delete statement	M	Y	if delete statement is supposed to be there, add it in	Add a record to demo.NightlySalesStaging that doesn't exist in dbo.SalesHistory. Confirm it doesn't exist after proc is run
172	Drop Table and recreate through SELECT INTO	N	Y	change DROP TABLE to TRUNCATE TABLE change SELECT INTO into INSERT INTO statement confirm table definition is correct	confirm that stored procedure still works confirm performance isn't impacted if indexes are added
192	Drop Table and recreate through SELECT INTO	N	Y	change DROP TABLE to TRUNCATE TABLE change SELECT INTO into INSERT INTO statement confirm table definition is correct	confirm that stored procedure still works confirm performance isn't impacted if indexes are added
212	Drop Table and recreate through SELECT INTO (rpt_Avg_Profit_Classification_Type)	Y	Y	BUG: create table statements creates an integer. Populated by an scalar UDF that returns a money datatype. change DROP TABLE to TRUNCATE TABLE change SELECT INTO into INSERT INTO statement confirm table definition is correct	confirm that stored procedure still works confirm performance isn't impacted if indexes are added
268	Drop Table and recreate through SELECT INTO	N	Y	change DROP TABLE to TRUNCATE TABLE change SELECT INTO into INSERT INTO statement confirm table definition is correct	confirm that stored procedure still works confirm performance isn't impacted if indexes are added

Business Plan of Attack

Business Plan of Attack

- Prioritize the issues
- Estimate time
- Additional resource needs
- Development\Release schedules
- Prioritize with existing projects

Prioritize the issues

- Which needs the least effort?
- Which are the riskiest?
- Which has the biggest application impact?
- Which has the biggest business impact?

Estimate Time

- How long does each fix take?
- How does that fit in with existing development workflows?
  - Sprints
- What’s the estimated time to test each fix?
  - Unit testing
  - Regression testing

Determine Additional Resource Needs

- QA
  - Performance Impact testing
  - Regression testing
- Developers
  - Application changes
- Database developers
  - How many are available?

Development & Release Schedules

- Release planning
  - General release
  - Hot fix
  - Client one-off hot fix

Prioritize with Existing Projects

- How will the changes affect other project timelines?
- Should this project take priority over other projects?
- How to convince project managers?

Bringing it all together

Make your business case

Final Thoughts…

Perfect is the enemy of good

3 languages

Article Talk Tools

From Wikipedia, the free encyclopedia

Perfect is the enemy of good is an aphorism which means insistence on perfection often p 30–20 rule ex me to comple he effort.[1 effort results in diminishing returns, further activity becomes increasingly inefficient.

The first 80% takes 20% of the time.

The last 20% takes 80% of the time.

https://en.wikipedia.org/wiki/Perfect_is_the_enemy_of_good

Additional Resources

- Redgate: "How Do My Peers Do This?" The latest best practices for IT architects implementing a major business initiative

Additional Resources - Tools

- Code Analysis:
  - Mala Mahadevan: Stairway to ScriptDOM
  - Redgate: SQL Prompt SQL Code Analysis
  - SQLFluff: https://sqlfluff.com/
  - Microsoft: Overview of Extensibility For Database Code Analysis Rules

Additional Resources – Tools (cont'd)

- sp_helpExpandView (Andy Yun)
- sp_HumanEvents (Erik Darling)
- sp_QuickieStore (Erik Darling)

Thank you

If you have any other questions, feel free to reach out and ask!

Deborah Melkin

Email: dgmelkin@gmail.com

Blog: DebtheDBA.wordpress.com

GitHub: github.com/DebTheDBA