


SQLskills UG Presentation

Optimizing Stored Procedures Plan Stability Testing



Kimberly L. Tripp
Kimberly@SQLskills.com
@KimberlyLTripp

- **Team of world-renowned SQL Server experts:**
 - Paul S. Randal (@PaulRandal)
 - Erin Stellato (@ErinStellato)
 - Jonathan Kehayias (@SQLPoolBoy)
 - Tim Radney (@TRadney)
- **Instructor-led training: Immersion Events (US, UK, and Ireland)**
- **Online training:**  PLURALSIGHT <http://pluralsight.com/>
- **Consulting: health checks, hardware, performance, upgrades**
- **Remote DBA: system monitoring and troubleshooting**
- **Conferences: SQLBits, PASS Summit, TechEd, DEVteach**
- **Become a SQLskills Insider**
 - <https://www.sqlskills.com/Insider>





PLURALSIGHT

- Email paul@SQLskills.com with the subject line: **Pluralsight code**
Get a FREE (no catches / no credit card) 30-day trial of our **180+ hours** of SQLskills content on Pluralsight including 10+ hours on Optimizing Procedural Code!
- Check out courses like these:
 - [SQL Server: Why Physical Database Design Matters](#)
 - 4 hours on row structures, index structures, and why these things matter
 - [SQL Server: Optimizing Stored Procedure Performance](#) (Parts 1 & 2)
 - 11 hours on stored procedure performance tuning and troubleshooting
 - [SQL Server: Logging, Recovery, and the Transaction Log](#)
 - 7 hours on understanding and optimizing logging (by Paul Randal)
 - [SQL Server: Performance Troubleshooting Using Wait Statistics](#)
 - 4.5 hours on waits, latches, spinlocks (by Paul Randal)
- See our full online library:
<https://www.sqlskills.com/sql-server-training/online-training/>



Kimberly L. Tripp

Founder / President, SQLskills


 Kimberly@SQLskills.com

 [@KimberlyLTripp](https://twitter.com/KimberlyLTripp)

 www.sqlskills.com/blogs/Kimberly



Consultant / Trainer / Speaker / Writer


- Author / instructor for SQL Server Immersion Events: IEPTO1, IEPTO2, IEVLT, IETLB, IEQuery
- Author / presenter for Pluralsight 
- Instructor and exam author for SQL Server and Sharepoint MCMs
- Author / manager of SQL Server 2005 and 2008 Launch Content
- Author / speaker at Microsoft TechEd, SQLPASS, ITForum, TechDays, TechReady, DEVTeach, + + +
- Author of SQL Server Whitepapers on MSDN/TechNet: Partitioning, Snapshot Isolation, Manageability, SQLCLR for DBAs
- Author / presenter for MSDN and TechNet (25+)
- Instructor / SME for Microsoft University

Data Platform MVP



- 17 years: 2002-2019

When I'm not working with SQL

- Traveler/adventurer and avid diver/photographer: www.BlueWaterImages.com
- @KimberlyLTripp on Insta 

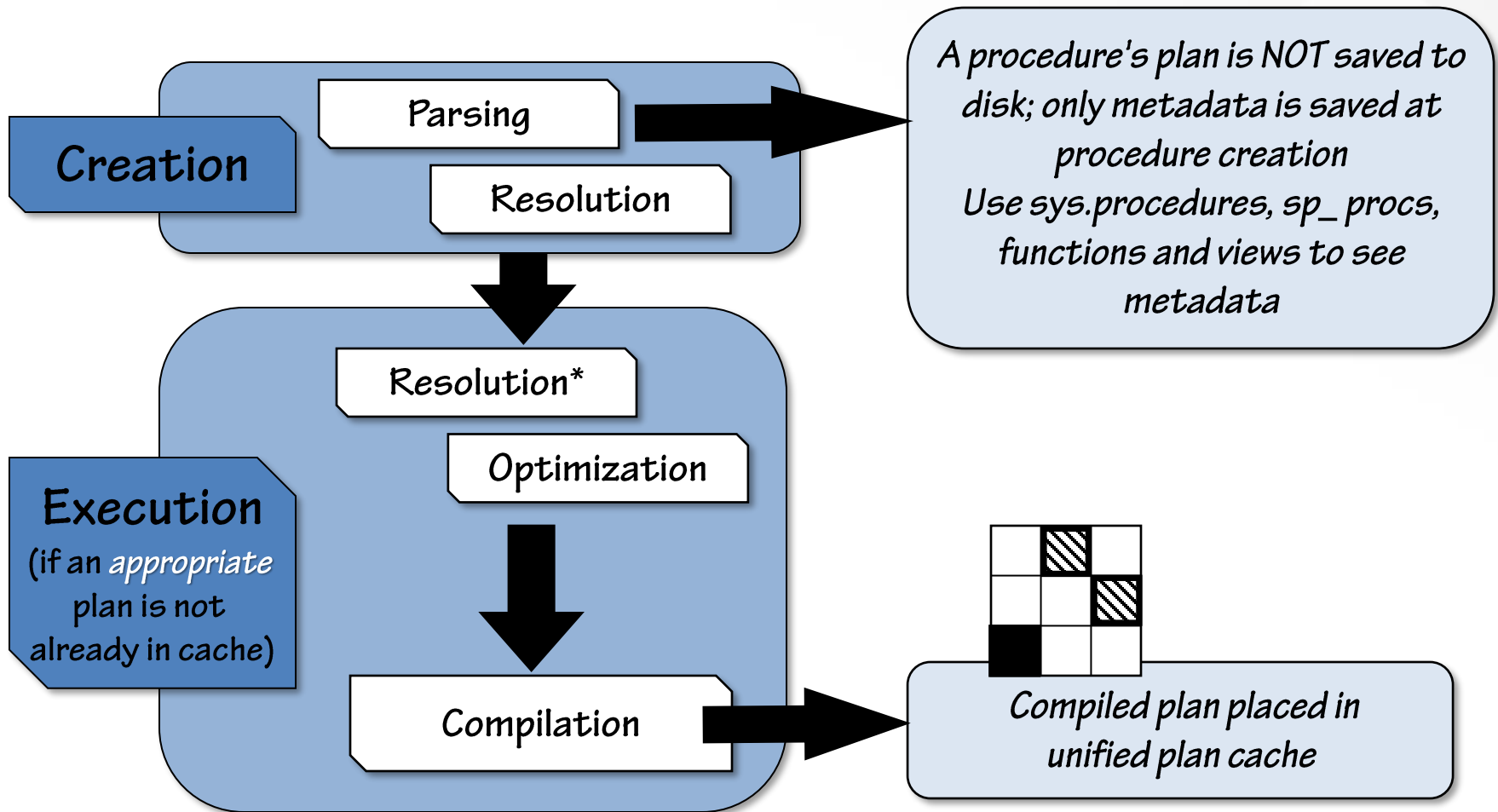


Overview

- Processing Stored Procedures
- Stored Procedure Optimization
- Plan Invalidation
- Procedure Caching
- Plan Quality Problems
- When to Recompile
- Key Problem with Recompilation
- The Hybrid Approach

*Make sure to review
all of the HIDDEN slides!*

Processing Stored Procedures



Session Settings and Client Connectivity

This comes from my Pluralsight course:
**SQL Server: Optimizing Stored
 Procedure Performance – Part 2**
 Section: Session Settings

SET Options	Required for Perf Features	Default Server Value	SSMS	SQLCMD	SQL Server Agent	Default OLE DB and ODBC Value	Default DB- Library Value	.NET / Your App
ANSI_DEFAULTS ²	NO	OFF	OFF	OFF	OFF	OFF	OFF	?
ANSI_NULL_DFLT_ON ²	NO	OFF	ON	ON	ON	ON	OFF	?
ANSI_NULLS ^{1,3}	YES = ON	OFF	ON	ON	ON	ON	OFF	?
ANSI_PADDING ^{2,3}	YES = ON	ON	ON	ON	ON	ON	OFF	?
ANSI_WARNINGS ²	YES = ON	OFF	ON	ON	ON	ON	OFF	?
ARITHABORT ²	YES = ON	ON	ON	OFF	OFF	OFF	OFF	?
CONCAT_NULL_YIELDS_NULL ^{2,3}	YES = ON	OFF	ON	ON	ON	ON	OFF	?
NUMERIC_ROUNDABORT ²	YES = OFF	OFF	OFF	OFF	OFF	OFF	OFF	?
QUOTED_IDENTIFIER ¹	YES = ON	OFF	ON	OFF	OFF	ON	OFF	?

- (1) The only state that's important is how it's set when the stored procedure is CREATED; the runtime setting is irrelevant.
- (2) If different than existing plan in cache, will be recompiled and added to the plan cache; performance may vary at execution.
- (3) In a future version of SQL Server, these options will always be ON and any applications that explicitly set the option to OFF will produce an error. Avoid using this feature in new development work and plan to modify applications that currently use this feature.

Troubleshooting Session Settings

This comes from my Pluralsight course:
SQL Server: Optimizing Stored Procedure Performance – Part 2
Section: Session Settings

- Currently connected users (and currently set session settings) can be viewed
 - At the user level:

```
SELECT *  
FROM [sys].[dm_exec_sessions]  
WHERE [session_id] = @@SPID
```
 - Across sessions:

```
SELECT *  
FROM [sys].[dm_exec_sessions]  
WHERE [is_user_process] = 1
```
 - This does not guarantee that the session settings are not being changed elsewhere in the application (or, by the user [e.g. ODBC DSN settings])

Stored Procedure Optimization

- Plan is generated when no plan already exists in cache
- Plans are never saved on disk and can “fall out” of cache
 - Forced out through:
 - Server restart
 - DBCC FREEPROCCACHE
 - DBCC FLUSHPROCINDB(DBID)
 - New in SQL Server 2016:
 - ALTER DATABASE SCOPED CONFIGURATION CLEAR PROCEDURE_CACHE;
 - sp_recompile
 - Aged out through non-use
 - Schema of base object changes
 - Statistics of base objects change
 - See next slide (hidden) for details specific to version
- When a plan exists in cache, all subsequent executions use that plan
- Plans are not regenerated for subsequent executions (even when statistics are added – use sp_recompile after adding statistics)

*Entire database's PLAN cache
not just procedures...*



Plan Invalidation – A Painful History

- In SQL Server 2012 and higher, statistics updates only cause plan invalidation IF the data has changed
(what defines data change? At least ONE row has changed...)
- In prior versions (SQL Server 2008R2, 2008, 2005)
 - If database option: auto_update_stats is ON
 - Updating statistics runs and always caused plan invalidation
 - If database option: auto_update_stats is OFF
 - Updating statistics runs but does NOT cause plan invalidation
- **Recommendations:**
 - Do not turn off auto_update_stats; it should be left ON
 - If off, and in earlier versions, use sp_recompile after updating stats
 - In SQL Server 2012 and higher, only update stats (programmatically) if a reasonable percentage of data has changed...
 - Also, use sp_recompile after adding indexes or adding new statistics
- **NOTE: Check out the resources slide for some blog posts on this issue**

Procedure Caching – Isn't that the Point?

- **Reusing plans can be good**
 - When different parameters don't warrant a plan change to execute optimally, then saving and reusing is excellent!
 - SQL Server saves time in compilation
- **Reusing plans can be VERY bad**
 - When different parameters wildly change the size of the result set and their optimal plans vary, then reusing the plan can be horribly bad
 - Don't worry, I'll show you how to see this...
 - Don't worry, I'll show you a few options to control / fix this!

Multipurpose Procedures

- Stored procedures with n parameters where any combination of these parameters can be supplied
- Often the code looks like this:

```
WHERE ...  
    (column1 = @parameter1 OR @parameter1 IS NULL)  
    AND (column2 = @parameter2 OR @parameter2 IS NULL)  
    ... AND (columnN = @parameterN OR @parameterN IS NULL)
```

- These are very difficult for the optimizer to “generalize” and what often happens is a generally bad plan
- The hybrid approach to fixing multipurpose procedures:
 - Concatenate parameter ONLY when the parameter is NOT NULL
 - Use `sp_executesql...` but with a twist!
 - Add `OPTION (RECOMPILE)` for cases where plan stability varies

Demo

Multipurpose Procedures

#WorkGreatInDev

#DontScaleInProd

Demo: Key Points

- What runs well in development might not scale unless someone's looking at the plans and understanding how they're generated
- Data size and data quality DO MATTER in development / [and, especially] test
- The parameters that generate the plan define the performance characteristics for subsequent executions
 - Depending on the execution characteristics of your workload, some plans in cache aren't nearly as bad as others (and won't be noticeable)
 - Some plans can cause tremendous performance problems LATER but with these data sizes...
- You can see the current plan in cache vs. the optimal plan for a combination of parameters by using EXECUTE WITH RECOMPILE (sadly, not always perfect)
- You have to look at the plans and know the pitfalls of COMPILED value vs. RUNTIME value

If no one's there, does it make a sound?

Test	Test 1 / 5 / 10	Test 2	Test 3	Test 4*		Test 8	Test 11		Test 13	Test 14	
1	50	78	293	114		79	126		926	369	
2	90035	78	90278	91981	Spilled	79	126	Spilled	926	369	Spilled
3	90035	90063	293	91981	Spilled	91946	91993	Spilled	926	369	
4	3152	90063	37517	507		91946	3300		926	38376	Spilled
5	50	78	293	114		79	126		926	369	
6	90035	90063	293	91981	Spilled	91946	91993	Spilled	926	369	
7	90035	90063	3392	91981	Spilled	91946	91993	Spilled	926	3540	
8	90035	345	90278	91981	Spilled	359	406	Spilled	926	649	Spilled
9	90035	90063	293	91981	Spilled	91946	91993	Spilled	926	369	
10	50	78	293	114		79	126		926	369	
11	3152	4635	37517	507		4739	271		926	2257	Spilled
12	90035	90063	293	91981	Spilled	91946	91993	Spilled	926	369	
13	90035	25512	17702	91981	Spilled	26050	26097	Spilled	926	5834	
14	90035	25512	1178	91981	Spilled	26050	26097	Spilled	926	646	
				Two worktables			Two worktables			Two worktables	

Play around with script:

MultipurposeProcedureExecutions_wDifferentParameters.sql

Is There No Hope?

■ What do we do?

- Execute everything WITH RECOMPILE?
 - NO: not in sys.dm_exec_procedure_stats or sys.dm_exec_query_stats
- Try sp_recompile to kick the plan out of cache and HOPE that the next execution is the “more common” plan for our workload?
 - NO: only fixed for the “first execution after the recompile”
- Update statistics?
 - NO! Sadly, this is a common “solution” and often has desirable results because of the side effect that the plan is invalidated (but, at the price of having updated the statistics)
- Add OPTION (RECOMPILE) to the code?
 - Come on, I KNOW you’ve heard this...
 - Fantastic as an interim solution and a great quick-fix in production

Plan Quality Problems

RECOMPILATION = OPTIMIZATION

OPTIMIZATION = RECOMPILATION

- Questions to consider?
 - When do you want to recompile?
 - What options do you have for recompilation – and, at what granularity?
 - How do you know you need to recompile?
 - Do you want to recompile the entire procedure or only part of it?
 - Can you test it? (and, what are we looking for?)

When to Recompile?

- When the plan for a statement within a procedure is not optimal as parameters change
- Cost of recompilation might be significantly less than the execution cost of a bad plan!
- Why?
 - Most of your time is in execution
 - Better plan means MUCH faster execution
 - Some plans just don't work for a wide variety of execution cases
 - Some plans should NEVER be saved
- Do you want to do this for every procedure?
 - **No, but start with the highest priority / expensive procedures that aren't performing well first – and test!!**

*If you don't know which procedures are the most expensive – check out the DMV:
`dm_exec_procedure_stats`*

Demo

Multipurpose Procedures – The QUICK & EASY Fix!

#WorkGreatInDev

#ScalesToAPointInProd

Demo: The QUICK & EASY Fix

Query	With OPTION (RECOMPILE)
1	3
2	5
3	6
4	413
5	3
6	6
7	926
8	19
9	6
10	3
11	31
12	6
13	926
14	363

- OK, I know – it looks AMAZING
- Yes, it's SUPER easy
- Yes, I know you want to change all of your code RIGHT now...
- Do NOT RUN OUT OF THE ROOM
- Do NOT IMMEDIATELY CONNECT TO PRODUCTION and start changing code (well, OK, this might be a good idea for a few of your procs 😊)
- WAIT... I still have yet another (and often BETTER) way to solve this

Demo: Key Problem...

- **You can get HUGE benefits just by switching to OPTION(RECOMPILE)**
 - Yes, but you can't do this everywhere – you'll use too much CPU
 - GREAT starting point and a fantastic interim fix (but *"needs more investigating"*)
- **For some procedures, where execution patterns exist and/or certain parameters are stable, a better solution would be to use a hybrid approach:**
 - Stable plans are cached
 - Saving time AND CPUs
 - Unstable plans are recompiled
 - Only requiring the added CPU where needed
 - Depending on your workload this might be a smaller percentage of the executions

Other Options for Recompilation

HIDDEN SLIDE: Check out the Pluralsight courses for in-depth information about these options!

- **CREATE ... WITH RECOMPILE**
- **EXECUTE ... WITH RECOMPILE**
- **sp_recompile objname**
- **Statement-level recompilation**
 - Dynamic string execution (statement is built dynamically and then treated as an ad hoc statement); this works in all versions incl. SQL Server 2000
 - Modularized code (break different options into separate procedures [not just conditional statements])
 - Added in SQL Server 2005+: **OPTION(RECOMPILE)**
 - Added in SQL Server 2005+: **OPTION (OPTIMIZE FOR (@variable_name = constant, ...))**
 - Added in SQL Server 2008+: **OPTION (OPTIMIZE FOR UNKNOWN)**

Quick Notes

- Do NOT use **CREATE WITH RECOMPILE**
- Use these for testing and troubleshooting
 - **EXECUTE with RECOMPILE**
 - **sp_recompile**
- Look toward more granular options
 - If you WANT to recompile for EVERY execution
 - **OPTION (RECOMPILE)**
 - Dynamic strings – executed with **EXEC (@ExecStr)**
 - If you have VERY specific and known patterns you can use the others... but I still have a better idea!

HIDDEN SLIDE: Check out the “Optimizing Stored Procedures – Part 1” Pluralsight course for in-depth information about these options!

Review

- Processing Stored Procedures
- Stored Procedure Optimization
- Plan Invalidation
- Procedure Caching
- Plan Quality Problems
- When to Recompile
- Key Problem with Recompile
- The Hybrid Approach

SQLskills SQL101

- **Our SQLskills SQL101 blog post series**
 - Blog posts to make sure you've filled in the gaps... *back to the basics*
 - Relatively short posts with key points on features we see often implemented incorrectly or misunderstood
 - The FIRST post – STORED PROCEDURES!
 - www.sqlskills.com/blogs/kimberly/sqlskills-sql101-stored-procedures
 - Aggregated posts from the entire SQLskills team:
 - www.sqlskills.com/help/SQL101
- **Improve your SQL skills with SQLskills 😊**

Summary: Statement Caching

HIDDEN SLIDE: Check out the PS courses on AdHoc Statements for in-depth information about **PARAMETERIZATION!**

- **Ad hoc statements**

- Simple parameterization – almost all statements will be compiled just for that execution; they will not be parameterized/saved (see rules for parameterization in whitepaper)
- Forced parameterization – most statements will be parameterized/saved (you'll see this in decreased CPU/compilations and potential for parameter-sniffing problems)

- ***sp_executesql* (or, prepared statements)**

- A fantastic way to reduce the CPU/cache overhead that ad hoc has, but should only be used when a plan is stable and consistent
- This is a better way to force a statement into cache as opposed to using forced parameterization database-wide

- **Stored procedures**

- Can be “sniffed” but there are exceptions to what SQL Server will store in their plans

Thank you!

