

SQLintersection

SQL318

Stored Procedures: Tips and Tricks for Optimal Performance

Kimberly L. Tripp

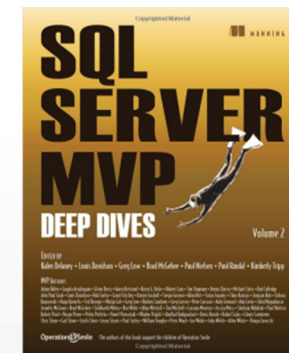
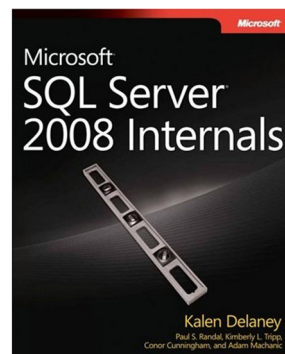
Kimberly@SQLskills.com



SQL
intersection

Author/Instructor: Kimberly L. Tripp

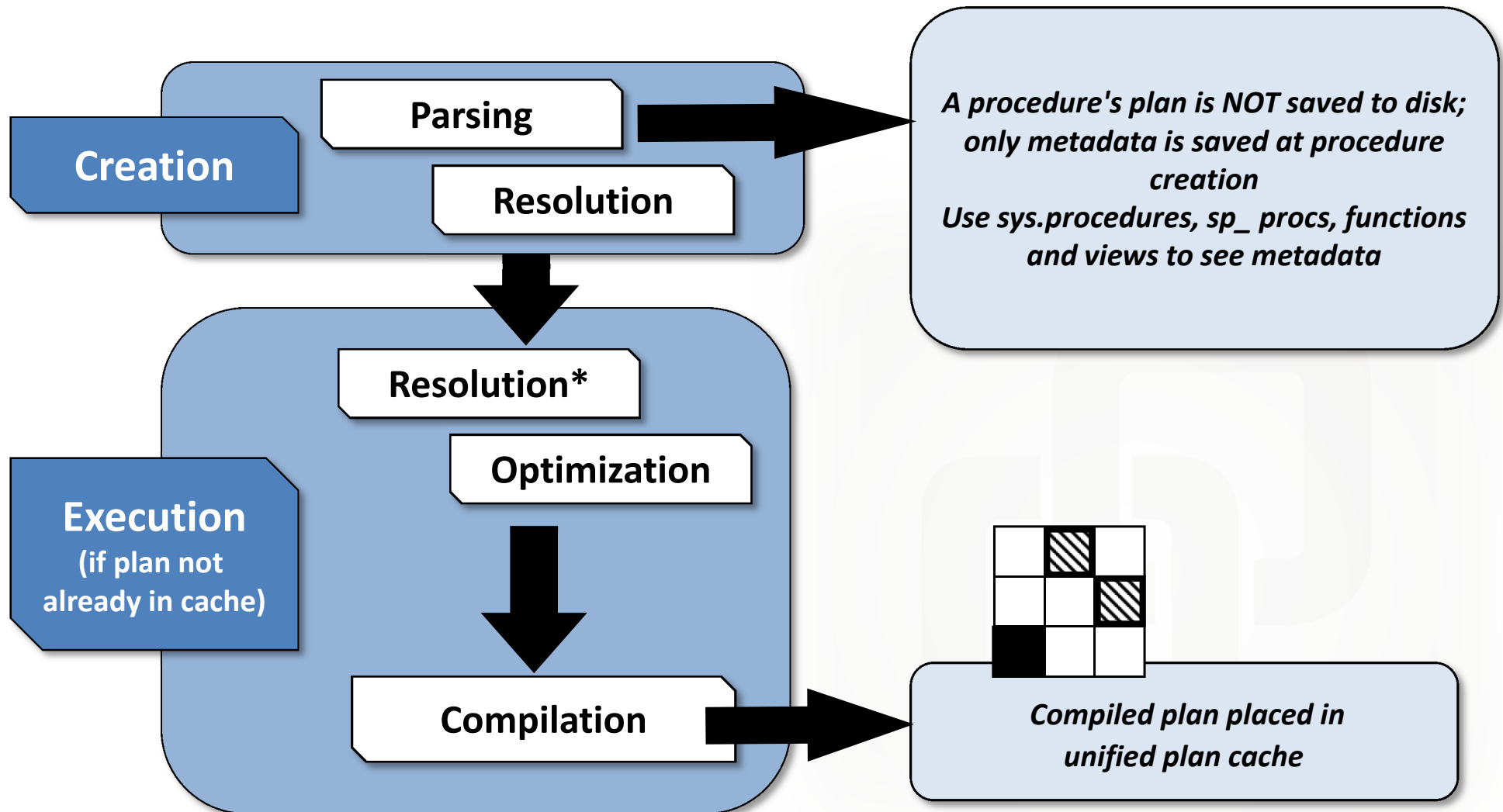
- Consultant/Trainer/Speaker/Writer
- President/Founder, SYSolutions, Inc.
 - e-mail: Kimberly@SQLskills.com
 - blog: <http://www.SQLskills.com/blogs/Kimberly>
 - Twitter: @KimberlyLTripp
- Author/Instructor for SQL Server Immersion Events
- Instructor for multiple rotations of both the SQL MCM & Sharepoint MCM
- Author/Manager of SQL Server 2005 & 2008 Launch Content
- Author/Speaker at Microsoft TechEd, SQL Connections, SQLPASS, ITForum, TechDays
- Author of several SQL Server Whitepapers on MSDN/TechNet: Partitioning, Snapshot Isolation, Manageability, SQLCLR for DBAs
- Author/Presenter for more than 25 online webcasts on MSDN and TechNet
- Author/Presenter for multiple online courses at Pluralsight 
- Co-author MSPress Title: SQL Server 2008 Internals, the SQL Server MVP Project (1 & 2), and SQL Server 2000 High Availability
- Owner and Manager of the SQLIntersection conference



Session Overview

- **Processing stored procedures**
- **Plan invalidation**
- **Plan caching and reuse**
- **Multipurpose procedures defined**
- **Define, discuss, and demo parameter sniffing**
- **Define, discuss, and demo when parameter sniffing goes horribly wrong (parameter sniffing problems = PSP)**

Processing Stored Procedures



Stored Procedure Plan Reuse

- **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 | some configuration changes | some database-level changes
 - DBCC FREEPROCCACHE | DBCC FLUSHPROCINDB | sp_recompile
 - Aged out through non-use
 - Schema of base object changes
 - Statistics of base objects change
 - See next slide for details specific to version
- **When a plan exists in cache, subsequent executions use that plan**
- **Plans are not regenerated for subsequent executions (even when statistics are added – use sp_recompile after adding statistics)**

Plan Invalidation – A Painful History

- **As of SQL Server 2012, 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 always ran but does NOT cause plan invalidation
- **Recommendations:**
 - In earlier versions, use `sp_recompile` after updating stats against a table
 - In SQL 2012, only update stats (programmatically) if / when data has changed...
 - Also, use `sp_recompile` after adding indexes
- **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!
- **To do this, I'll explain this in the context of a “multipurpose procedure”**

The Multipurpose Procedure

Customer Lookup



Customer Number:

#####

Customer Last Name:

Last name or partial Tri%

Customer First Name:

First name or partial Kim%

Customer Middle Initial:

L

Customer Email Address:

Email address or partial email@hotmail%

Region:

##

List Regions

Customer Type:

##

List Types

Enter any combination of criteria for searching. Searches for a customer Number, a full name without wildcards, or an email should be fast. Wildcard matches may be slow and inefficient; grab a cup of coffee!

SEARCH

Demo

Multipurpose Procedures

#WorkGreatInDev

#DontScaleInProd



SQL
intersection

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. a [potentially] more optimal plan for a combination of parameters by using EXECUTE WITH RECOMPILE (sadly, not always the best plan)**
- **You have to look at the plans; know the pitfalls of COMPILED value vs. RUNTIME value**

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

Query	Query 1 / 5 / 10	Query 2	Query 3	Query 4*		Query 8	Query 11		Query 13	Query 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	

Is There No Hope?

■ What do we do?

- Execute everything WITH RECOMPILE?
 - To be honest, this doesn't even get us the best plan...
- Try sp_recompile to kick the plan out of cache and HOPE that the next execution is the “more common” plan for our workload?
- Update statistics?
 - 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...

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 mean 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!!

Demo

Multipurpose Procedures: The QUICK & EASY Fix!

#WorkGreatInDev

#ScalesWellInProd



SQL
intersection

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 procedures 😊)
- For longer term scalability, I have yet another (and, often BETTER) way to solve this

Other Options for Recompilation

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

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)
 - Simple, easy syntax, preferred and recommended option for recompilation
 - Dynamic strings – executed with EXEC (@ExecStr)
 - More complicated to write / troubleshoot / tune, need to be tested thoroughly to ensure SQLinjection potential is eliminated (only possible with certain patterns – such as the one we’re looking at)
 - If you have VERY specific and known patterns you can use the others

The Checkered Past of OPTION (RECOMPILE)

- **FYI – VERY RARE BUG w/FIX: Incorrect result when you execute a query that uses WITH RECOMPILE option in SQL Server 2012 or SQL Server 2014. KB2965069 Fixed in:**
 - [Cumulative Update 4 for SQL Server 2014](#)
 - [Cumulative Update 2 for SQL Server 2012 SP2](#)
 - [Cumulative Update 11 for SQL Server 2012 SP1](#)
- **SQL Server 2008 CU4 and SQL Server 2008 R2 CU1 fixed similar problem: KB article 968693 FIX**
 - SQL Server 2008 R2 SP1 fixed:
 - Problems where special performance-related features that should have been accessible with OPTION (RECOMPILE) were not
- **If I ever run into problems with OPTION (RECOMPILE), I always consider using a dynamic string instead – this always works!**

When Is Recompilation Required

- **Specific types of code cannot support a single plan or perform well with one that's cached**
 - Filters
 - SQL Server cannot save a single plan based on filtered indexes
 - SQL Server won't use a filtered index unless recompilation is performed:
 - Using `OPTION (RECOMPILE)`: this is the most safe as it forces the statement to be recompiled and is not affected by the database setting for parameterization
 - Using a dynamically constructed string (unless forced parameterization is on)
 - Partitioned Views
 - SQL Server CAN do partition elimination correctly (even if the first execution is only against one partition)
 - You will get the correct data
 - However, the first execution will set the “estimates” for the plan – which could vary [significantly] across the sets

Multipurpose Procedures

- **Stored procedures with n parameters where any combination of these parameters can be supplied**

- **Often the code looks like this:**

```
WHERE ...  
(column1 = @variable1 OR @variable1 IS NULL)  
  AND (column2 = @variable2 OR @variable2 IS NULL)  
  ... AND (columnN = @variableN OR @variableN IS NULL)
```

- **These are very difficult for the optimizer to “generalize” and what often happens is a generally bad plan**
- **How do you fix it?**
 - Concatenate ONLY when the variable is NOT NULL
 - Use sp_executesql... but with a twist!
 - Add OPTION (RECOMPILE) for cases where plan stability varies

Demo

Making Multipurpose Procedures SCALE

#WorksGreatInDev

#ScalesToInfinityAndBeyondInProd



SQL
intersection

Demo: Key Points (1 of 2)

- **Multipurpose procedure at full scale: never let it finish... once, killed at 13 hours with roughly ~50K executions (only half way)**
 - Waits: MASSIVE CXPACKET waits due to horrible plans causing massive index scans over and over and over
- **Solution 1: Use OPTION (RECOMPILE)**
 - Total time is 51 seconds, still showing parallelism WITH 100% CPU
- **Solution 2: Build the statement and execute with sp_executesql adding OPTION (RECOMPILE) for unstable parameter combinations**
 - Total time is about 34 seconds with even smaller signal wait times showing even less CPU pressure
 - 50% less time with less CPU = more scalable!

Wait Statistics of Solutions 1 and 2

■ Solution 1

WaitType	Wait_S	WaitCount	AvgWait_S	AvgRes_S	AvgSig_S
CXPACKET	360.8	8051	0.0448	0.0434	0.0014
SOS_SCHEDULER_YIELD	206.73	49169	0.0042	0	0.0042

■ Solution 2

WaitType	Wait_S	WaitCount	AvgWait_S	AvgRes_S	AvgSig_S
CXPACKET	312.5	8286	0.0377	0.0364	0.0013
SOS_SCHEDULER_YIELD	126.78	43471	0.0029	0	0.0029

Wait Statistics of Solutions 2 and 3

- **Solution 3 (if you find you need to use dynamic string execution the performance characteristics should be the same as solution 2)**

WaitType	Wait_S	WaitCount	AvgWait_S	AvgRes_S	AvgSig_S
CXPACKET	298.25	8410	0.0355	0.0342	0.0013
SOS_SCHEDULER_YIELD	116.11	41710	0.0028	0.0000	0.0028

- **Solution 2 is recommended for the simplicity and because of the reduced possibility of SQLinjection**

WaitType	Wait_S	WaitCount	AvgWait_S	AvgRes_S	AvgSig_S
CXPACKET	312.5	8286	0.0377	0.0364	0.0013
SOS_SCHEDULER_YIELD	126.78	43471	0.0029	0	0.0029

Demo: Key Points (2 of 2)

- **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 (but temporary) fix
- **Writing multipurpose procedures with the hybrid strategy gives multiple benefits**
 - 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

Summary

Different parameter combinations can be handled EASILY AND EFFECTIVELY with a relatively easy change to procedure coding practices

- **Each sale take less time (get better performance)**
 - Sell more widgets
- **Save CPU and scale more**
 - Sell more widgets
- **Save power, save money**
- **Save the world!**

Resources



- **Pluralsight courses:**

- SQL Server: Optimizing Ad Hoc Statement Performance
 - <http://bit.ly/1nmYjHq>
- SQL Server: Optimizing Stored Procedure Performance
 - <http://bit.ly/1xwH7F6>

- **Whitepaper: Plan Caching and Recompilation in SQL Server 2012**

- <http://bit.ly/1gWKmKX>

- **Blog posts:**

- What caused that plan to go horribly wrong – should you update statistics? <http://www.sqlskills.com/blogs/kimberly/what-caused-that-plan-to-go-horribly-wrong-should-you-update-statistics/>
- Statistics and Recompilations: <http://erinstallato.com/2012/01/statistics-recompilations/> and Part II: <http://erinstallato.com/2012/02/statistics-recompilations-part-ii/>

Questions?



Don't forget to complete an online evaluation on EventBoard!

SQL318

Stored Procedures:

Tips and Tricks for Optimal Performance

Your evaluation helps organizers build better conferences
and helps speakers improve their sessions.



SQL
intersection

Thank you!