# Optimization and Recompilation

Kimberly L. Tripp
SQLskills.com
@KimberlyLTripp

pluralsight
hardcore dev and IT training

## Overview

- **Options for recompilation**
  - sp_recompile <*object_name*>
  - CREATE … WITH RECOMPILE
  - Special considerations for conditional logic
    - Modularization
  - EXECUTE … WITH RECOMPILE
  - Statement-level recompilation
    - OPTION (RECOMPILE)
    - OPTIMIZE FOR …
    - Server-wide: OPTIMIZE FOR UNKNOWN
    - The checkered past of OPTION (RECOMPILE)
    - Dynamic string execution
- **Multi-purpose procedures**

# Options for Recompilation

- sp_recompile <*object_name*>
- CREATE … WITH RECOMPILE
- EXECUTE … WITH RECOMPILE
- **Statement-level (sometimes called inline) recompilation**
  - SQL Server 2000 (and earlier) only had procedure-level compilation
    - Could simulate statement-level recompilation
  - From SQL Server 2005 onward, statement-level compilation / recompilation is the norm but you can also influence how the compilation is done / redone:
    - SQL Server 2005+: OPTION (RECOMPILE)
    - SQL Server 2005+: OPTION (OPTIMIZE FOR (@variable_name = constant, ...) )
    - SQL Server 2008+: OPTION (OPTIMIZE FOR UNKNOWN)

# sp_recompile <*object_name*>

- **sp_recompile <*procedure_name*>**
  - Evicts that procedure's statements and plans from cache
  - Always an option when evaluating for parameter sniffing
    - If the current plan is performing poorly and improves after invalidating / evicting that procedure's plans in cache, then we know our plan was not tuned for our parameters
- **sp_recompile <*table_name*>**
  - Invalidates all plans that reference the object specified from cache
  - Requires a SCH_M (schema modification lock) on the object being recompiled
    - NOTE: This can create horribly problematic blocking chains as there are absolutely no other locks that are compatible with a SCH_M lock
      - ALL queries/modifications/anything against the base table must complete before the SCH_M can be acquired.
      - While the SCH_M is waiting, so is every new request (waiting / blocked)

# CREATE … WITH RECOMPILE

- **SQL Server 2005 introduced statement-level recompilation, so no real need to request / force procedure-level recompilation**
  - Possibly for small procedures
  - When procedure returns widely-varying results
  - For backward compatibility
    - 2000: only complete procedure recompiles (KB article 263889 Compile Locks)
    - 2005 onward: statement-level recompilation
- **Always target the smallest amount possible to recompile!**
- **NOTE: If a procedure is created WITH RECOMPILE, the non-dynamic / ad hoc statement(s) won't show up in [sys].[dm_exec_query_stats] OR [sys].[dm_exec_procedure_stats]**
  - Result: very limited troubleshooting capabilities so **avoid using**

## Conditional Logic

- **During optimization SQL Server looks for any statements that can be optimized**
  - Variables are unknown (their state isn't set until execution)
  - Literals and parameters can be sniffed
- **Just like variables, the branching of a conditional statement is unknown**
  - It doesn't matter what WILL be executed only that the statement (during optimization) could be optimized with the literals and parameters supplied
- **If you think that you can branch just for optimization, don't**
  - It probably won't work like you think it should
- **Be careful of block statements / conditional logic**
  - SQL Server optimizes the process of optimization and this may lead to an less-than-optimal plan (no, really!)

## Modularization

- **Instead of having large blocks, consider breaking the stored procedure into smaller chunks**
  - Can handle the block of statements on a more granular basis
  - SQL Server will never step into a sub-procedure unless it executes
  - Sub-procedures can be optimized / compiled
    - CREATE sub-procedure WITH RECOMPILE
    - EXECUTE sub-procedure WITH RECOMPILE
    - Statement-level optimization options (this is the BEST OPTION; *more coming up*)

```
CREATE PROCEDURE [Original_procedurename]
   (<parameter list>)
AS
IF <conditional_logic>
   EXEC [subprocedure1] parameters;
ELSE
   EXEC [subprocedure2] parameters;
GO
```