SQL Server 200x Optimizing Stored Procedure Performance

Kimberly L. Tripp
SQLskills.com
Email: Kimberly@SQLskills.com
Blog: http://www.SQLskills.com/Blogs/Kimberly
http://www.SQLskills.com

SQL skills

Speaker – Kimberly L. Tripp

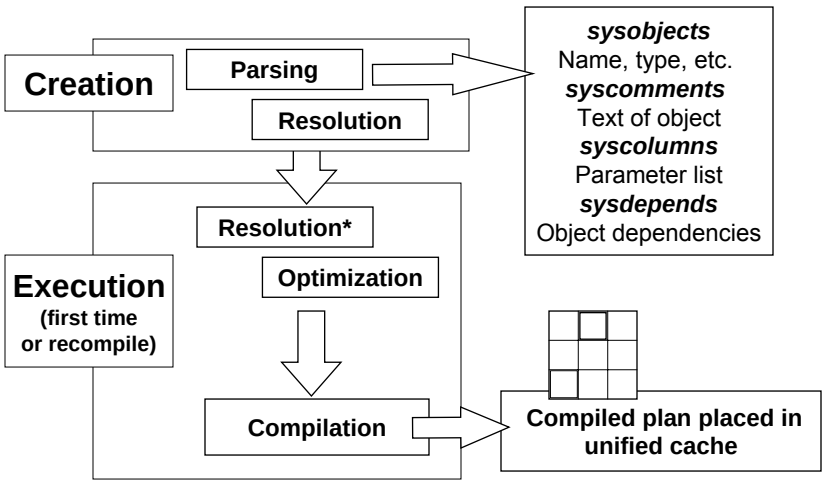
- Independent Consultant/Trainer/Speaker/Writer
- Founder, ***SY*Solutions, Inc.** www.SQLskills.com
  - Email: Kimberly@SQLskills.com
  - *Become a subscriber on SQLskills.com and learn about new resources which can improve your productivity and server performance!*
- Microsoft Regional Director http://msdn.microsoft.com/isv/rd/
- SQL Server MVP http://mvp.support.microsoft.com/
- Author for some SQL Server 2005 Whitepapers on MSDN (links from home page on www.SQLskills.com)
- SQL Server 2005 Launch Content Manager – Data Platform Track Sessions, Demos and Cross-training
- Writer/Editor for SQL Magazine www.sqlmag.com

SQL skills.com
immerse yourself in sql server

Overview

- Initial Processing - Review
  - Resolution
  - Compilation/Optimization
  - Execution/Recompilation
- Recompilation Issues
  - When do you want to Recompile?
  - Options for Recompilation?
  - What to Recompile?
- **For extra info:** Stored Procedure Best Practices
  - Naming Conventions
  - Writing Solid Code
  - Excessive Recompilations – How? Detecting?

❖Processing of Stored Procedures



SQLskills.com
immerse yourself in sql server

Compilation/Optimization

- Based on parameters supplied
- Future executions will reuse the plan
- Complete optimization of all code passed (*more on this coming up…statement-based recompilation and/or modular code!*)
- Poor coding practices can cause excessive locking/blocking
- Excessive recompilations can cause poor performance

Execution/Recompilation

- Upon Execution if a plan is not already in cache then a new plan is compiled and placed into cache
- What can cause a plan to become invalidated and/or fall out of cache:
  - Server restart
  - Plan is aged out due to low use
  - DBCC FREEPROCCACHE (sometime desired to force it)
- Base Data within the tables - changes:
  - Same algorithm as AutoStats, see Q195565 INF: How SQL Server 7.0 and SQL Server 2000 Autostats Work

Recompilation Issues

RECOMPILATION = OPTIMIZATION
OPTIMIZATION = RECOMPILATION

- When do you want to recompile?
- What options do you have Recompilation?
- 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?

When to recompile?

- When the plan for a given statement within a procedure is not consistent in execution plan–due to parameter changes
- Cost of recompilation might be significantly less than the execution cost of a bad plan!
- Why?
  - Faster Execution with a better plan
  - Saving plans for reuse is NOT always beneficial
  - Some plans should NEVER be saved
- Do you want to do this for every procedure?
  - No, but start with the highest priority/expensive procedures first!

Options for Recompilation

- CREATE … WITH RECOMPILE
- EXECUTE … WITH RECOMPILE
- sp_recompile objname
- Statement Recompilation
  - The old way
    - Dynamic String Execution
    - Modularized Code
  - The new way
    - OPTION(RECOMPILE)
    - OPTIMIZE FOR
      (@variable_name = literal_constant, ...)

CREATE … WITH RECOMPILE

- When procedure returns widely varying results
- When the plan is not consistent
- For SMALL procedures
- For backward compatibility
- Why?
  - 2000: Only complete procedure recompiles
  - 2005: Statement level recompilation
- Always target the smallest amount possible to recompile!

EXECUTE WITH RECOMPILE

- Excellent for Testing
- Verify plans for a variety of test cases
  - EXEC dbo.GetMemberInfo 'Tripp' WITH RECOMPILE
  - EXEC dbo.GetMemberInfo 'T%' WITH RECOMPILE
  - EXEC dbo.GetMemberInfo '%T%' WITH RECOMPILE
- Do the execution plans match?
- Are they consistent?
- Yes ⇨ then create the procedure normally
- No ⇨ Determine what should be recompiled

Statement-level Recompilation

- The old way: Modularizing your code
  - Doesn't hurt!
  - May allow better reuse of code "snippets"
- The new way: "inline recompilation"
  - OPTION(RECOMPILE)
    - Excellent when parameters cause the execution plan to widely vary
    - Bad because EVERY execution will recompile
  - OPTIMIZE FOR (@variable_name = literal_constant, ...)
    - Excellent when large majority of executions generate the same optimization time
    - You don't care that the minority may run slower with a less than optimal plan?

Modular Code – Still works!

Solution?
Do not use a lot of conditional SQL Statement Blocks
Call separate stored procedures instead!

IF (expression operator expression)
SQL Statement Block1
ELSE
SQL Statement Block2

Scenario 1 – upon first execution…
Parameters are passed such that the ELSE condition executes – BOTH Block1 and Block2 are optimized with the input parameters

Scenario 2 – upon first execution…
Parameters are passed such that the IF condition executes – ONLY Block1 is optimized. Block2 will be optimized when a parameter which forces the ELSE condition is passed.

See ModularProcedures.sql

sp_recompile

- Can be used to periodically and directly force recompilation of a procedure (or trigger)
- Can be used on tables and views to indirectly force the recompilation of all procedures and triggers that reference the specified table or view
- Does not actually recompile the procedures ⇨ Instead it invalidates plans for next execution
- SQL Server invalidates plans as data changes
- Never really negative – especially if you run it at night as part of batch processing after index rebuilds or statistics updates with FULLSCAN

Extra Info – Stored Procedure Best Practices

- Naming Conventions
  - Owner Qualify
  - Do not use sp_
- Modifying Procedures
- Write Solid Code
  - Writing Better Queries/Better Search Arguments
  - Changing Session Settings
  - Interleaving DML/DDL
  - Temp Table Usage
  - Modular Code
- Detecting Excessive Recompilations

Naming Conventions

- Owner Qualify to Eliminate Ambiguity
  - On execution
    EXEC dbo.procname
  - On creation
    CREATE PROC dbo.procname
    AS
    SELECT columnlist FROM dbo.tablename
    EXEC dbo.procname
- Minimize Blocking – initial cache lookup by owner will fail. It will not cause a recompile but excessive lookups can cause significant blocking and cache misses.
- Do not use **sp**_ in stored procedure names – causes cache misses on lookup as well because SQL Server looks in master first!

See KB Article Q263889

Modifying Procedures

- DROP and RECREATE
  - Loses the dependency chain stored in sysdepends
  - Loses the permissions already granted
  - Invalidates all plans
- ALTER PROC
  - Loses the dependency chain stored in sysdepends
  - **Retains the permissions**
  - Invalidates all plans
- To retain the dependency chain you must also ALTER all procedures that depend on the procedure being altered.

Changing SESSION Settings

- Certain Session Settings can be set within a stored procedure – some can be desired:
  - SET NOCOUNT ON
  - SET QUOTED_IDENTIFIER OFF (not recommended except for backward compatibility and upgrades)
- Some Session Settings will cause EVERY execution to force a recompile:
  - ANSI_DEFAULTS
  - ANSI_NULLS (***tip:*** do not use WHERE col = null, use col IS NULL)
  - ANSI_PADDING
  - ANSI_WARNINGS
  - CONCAT_NULL_YIELDS_NULL (***tip:*** use the ISNULL function to concatenate strings)
- Recommendation: DO NOT Change these session settings in the client or the server!

  See "SET Options that Affect Results" in the BOL

Interleaving DML/DDL Statements

- Objects that don't exist at procedure first execution cannot be optimized until statement execution
- Upon execution of a DDL statement the procedure gets recompiled to recompile the plans for the DML
- But wait – not all of the objects are created…so later executions of DDL force recompilation AGAIN…
- Don't interleave DDL and DML separate it…
- All DDL at the beginning of the proc, all DML later!

Data Manipulation

- Derived Tables
  - Nested Subquery in FROM clause
  - May optimize better than temp tables/variables
- Views
  - Another option – rewrite existing temp table code to use views instead (simple rewrite)
  - May optimize better than temp tables/variables
- Temp Tables
  - Should be considered
- Table Variables
  - Limitations might not affect you
  - Might be the most optimal

Temp Table Usage

- Temp Table can create excessive recompilations for procedures. Consider creating permanent tables (with indexes) and manipulating data there.
- Consider dropping and re-creating or rebuilding indexes as part of the procedure instead!
- Try not to create tables conditionally (IF create… ELSE create…)
- Use Profiler to see if there are significant recompiles
- Use KEEP PLAN on SELECT statements if data changes more than 6 times but the plan should not change.

Table Variable Usage

- Scope is limited to the local procedure\transaction
- Does not cause excessive recompiles due to local only access
  - No re-resolution on CREATE/ALTER
  - Temp Tables need re-resolution for nested procedures
- Only Key Indexes can be created
  - Definition of Table allows PRIMARY KEY/UNIQUE constraint indexes
  - Use TEMP TABLES if large volumes of data will be manipulated – create the right indexes for access
- Population
  - Does not support INSERT EXEC
  - Does not support SELECT INTO

SQLskills.com
immerse yourself in sql server

Temp Table vs. Table Variables

- Temp Table
  - PROs
    - •Can create useful nonclustered non-unique indexes to improve join performance
    - •Can access from other nested procedures
    - •Can populate with INSERT EXEC or SELECT INTO
  - CONs
    - •Potential for excessive recompiles due to resolution
- Table Variable Table
  - PROs
    - •Local only – no excessive recompiles
  - CONs
    - •Cannot create additional nonclustered indexes
    - •Not flexible on population

Detecting SP Recompilation

Event = SP:Recompile & Column = EventSubClass

1	Local Schema, bindings or permissions changed between compile and execute or executions Shouldn't happen often. If it does, isolate where/how changes occur and batch/sched. off hours
2	Statistics changed Thresholds for statistics of the different types of tables vary. Empty Tables (Permanent >= 500, Temp >= 6, Table Variables = No threshold) Tables with Data (Perm/Temp >= 500 + 20% cardinality, Table Variables = No threshold) If consistent plan then eliminate recompiles from changes in statistics by using (KEEPFIXED PLAN) optimizer hint in SELECT
3	Object not found at compile time, deferred check at run-time If the objects on which the procedure are based are permanent objects consider recreating
4	Set option changed in batch Best Coding practice: Consistency in client session settings. Consistency in development environment. Only use SET options when connection is started and when procedure is created.
5	Temp table schema, binding or permission changed Change coding practice for #temptable
6	Remote rowset schema, binding or permission changed. Gets stats from remote server, may recompile. If you're going to another server often – for a relatively small amount of static data you might consider periodically brining over a local copy?

SQLskills.com
immerse yourself in sql server

Profiling SP Performance

Event Notifications
& WMI Events
Trace=SP:Recompile

- Create New Trace (SQLProfilerTSQL_sps)
- Replace **SP:StmtStarting** w/**SP:StmtCompletion**
  - Better if you want to see a duration (starting events don't have a duration)
  - Add Duration as a Column Value
- If short term profiling for performance:
  - Add columns: Reads, Writes, Execution Plan
- Always use Filters
  - Database Name (only the db you want)
  - Exclude system IDs (checkbox on filter dialog)

Review

- Initial Processing - Review
  - Resolution
  - Compilation/Optimization
  - Execution/Recompilation
- Recompilation Issues
  - When do you want to Recompile?
  - Options for Recompilation?
  - What to Recompile?
- **For extra info:** Stored Procedure Best Practices
  - Naming Conventions
  - Writing Solid Code
  - Excessive Recompilations – How? Detecting?

SQLskills.com
immerse yourself in sql server

Resources

- Whitepaper: *Query Recompilation in SQL Server 2000* http://msdn.microsoft.com/library/default.asp?url=/nhp/Default.asp?contentid=28000409
- Webcasts: http://support.microsoft.com/default.aspx?PR=pwebcst&FR=0&SD=MSDN&LN=EN-US&CT=SD&SE=NONA