Why Use Stored Procedures?

Kimberly L. Tripp
SQLskills.com
@KimberlyLTripp

hardcore dev and IT training

Overview

- Different ways to execute SQL statements
- Some statements can be cached for reuse
- Reducing plan cache pollution
- Understanding sp_executesql
- Stored procedures / sp_executesql and the cache
- Parameter sniffing

Do Not Place Anything in This Space
(Add watermark during editing)
Note: Warning will not appear during Slide Show view.
pluralsight

Different Ways to Execute SQL Statements

- **Ad hoc statements**
  - Possibly, as auto-parameterized statements
- **Dynamic string execution (DSE)**
  - *EXECUTE (@string)*

These two behave EXACTLY the same way!

- **sp_executesql (forced statement caching)**
- **Prepared queries (forced statement caching through "parameter markers")**
  - Client-side caching from ODBC and OLEDB (parameter via question mark)
  - Exposed via *SQLPrepare / SQLExecute* and *ICommandPrepare*
- **Stored Procedures**

In this section, these behave the same way but some exceptions exist with certain statement types *inside* stored procedures (*more coming up on this*)

Do Not Place Anything in This Space
(Add watermark during editing)
Note: Warning will not appear during Slide Show view.
pluralsight

Some Statements Can Be Cached For Reuse (1)

- **Ad hoc statements and dynamic strings are evaluated at runtime**
- **Very simple ("safe") statements can be parameterized and cached**
  - It's generally a good thing that the plans are cached
    - Saves CPU/time
    - Reduced footprint in the cache
  - This *can* lead to a small amount of **prepared plan cache** bloat when the parameters are typed per execution:

```
SELECT … WHERE member_no = 12
 ↳ (@1 tinyint)SELECT … WHERE [member_no]=@1

SELECT … WHERE member_no = 278
 ↳ (@1 smallint)SELECT … WHERE [member_no]=@1

SELECT … WHERE member_no = 62578
 ↳ (@1 int)SELECT … WHERE [member_no]=@1
```

Do Not Place Anything in This Space
(Add watermark during editing)
Note: Warning will not appear during Slide Show view.
pluralsight

Some Statements Can Be Cached For Reuse (2)

- **Ad hoc statements and dynamic strings are evaluated at runtime**
- **And, unfortunately, most statements won't be safe**
  - Many query limitations
    - FROM clause cannot have more than one table
    - WHERE clause cannot have expressions joined by OR
    - WHERE clause cannot have an IN clause
    - Statement cannot contain a sub-query
    - VERY restrictive (see version-specific whitepapers on next slide for complete list)
  - Parameters do not change plan choice
- **Even when a statement is NOT safe, the un-parameterized statement (and the specific literal values) will be placed in the ad hoc plan cache**
  - Used for later "exact textual matching" cases
  - Eats up the cache quickly because most statements aren't safe and lots of statements are executing

Do Not Place Anything in This Space
(Add watermark during editing)
Note: Warning will not appear during Slide Show view.
pluralsight

Version-Specific Plan Caching Whitepapers

- Whitepaper: *Batch Compilation, Recompilation, and Plan Caching Issues in SQL Server 2005*
  - http://bit.ly/1pxrPwE
- Whitepaper: *Plan Caching in SQL Server 2008*
  - http://bit.ly/1lXJaZL
- Whitepaper: *Plan Caching and Recompilation in SQL Server 2012*
  - http://bit.ly/1gWKmKX

Do Not Place Anything in This Space
(Add watermark during editing)
Note: Warning will not appear during Slide Show view.
pluralsight

Reducing Plan Cache Pollution

- **Server setting: optimize for ad hoc workloads**
  - On first execution, only the query_hash will go into cache
  - On second execution (if), the plan will be placed in cache
- **Create a single and more consistent plan with covering indexes – might make the plan more stable!**
  - But will SQL Server detect it as safe? (lots of rules/restrictions, see whitepaper)
- **If the plan is stable then you can use sp_executesql or stored procedures to force the plan**
- **Have a SQL Agent job that periodically checks the single-use plan cache bloat and then clears the "SQL Plans" cache (if over 2GB, for example)**
  - Review my blog category on Plan Cache: http://bit.ly/1eqNP9H
  - And, specifically, this post: http://bit.ly/Rj0MIP

Do Not Place Anything in This Space
(Add watermark during editing)
Note: Warning will not appear during Slide Show view.
pluralsight

Understanding sp_executesql

- **Used to help build statements from applications**
- **Parameters are typed explicitly**
- **Forces a plan in cache for the parameterized string and subsequent executions will use this plan**
  - Can be EXCELLENT if the statement's plan is stable even with different parameter values
  - Can be horrible if the statement's most optimal plan would vary from execution to execution based on the different parameter values
- **Similar to dynamic string execution**
  - sp_executesql is a parameterized statement that works JUST like a stored procedure
  - Dynamic string execution (**`EXEC (@ExecStr)`**) is just a way of building an ad hoc statement that's not evaluated until runtime

Do Not Place Anything in This Space
(Add watermark during editing)
Note: Warning will not appear during Slide Show view.
pluralsight

Stored Procedures / sp_executesql and the Cache

- **Stored procedures and sp_executesql work the same way**
- **Literals and parameters can be optimized**
  - Literals inside the procedure CAN be “sniffed” and CAN leverage features like filtered indexes and filtered statistics
  - Parameters can go through “sniffing” and optimization for the specific value but they cannot use filtered objects for fear of subsequent execution failures
- **Variables are deemed unknown**
  - Variables are assigned at runtime through the execution of statements; their specific values are unknown until execution
  - SQL Server optimizes the statements BEFORE execution… how?
    - The values cannot be sniffed
    - The histogram cannot be used
    - The “average” is used, which comes from the density_vector portion of the statistics information

Do Not Place Anything in This Space
(Add watermark during editing)
Note: Warning will not appear during Slide Show view.
pluralsight

Parameter Sniffing

- **Literals and parameters can be "sniffed"**
- **Values that are known at the time of optimization can be fully evaluated against the histogram of data**
  - This allows more accurate estimates to be made
- **The initially "sniffed" parameters help to define an optimal plan for that execution**
- **Subsequent executions can suffer when ALL of the possible combinations of parameters don't benefit from the initial plan**
- **Enter the term *parameter sniffing problems* or PSP**
  - This is where parameter sniffing (which is normally good) becomes a parameter sniffing problem (which can be horribly bad)

Do Not Place Anything in This Space
(Add watermark during editing)
Note: Warning will not appear during Slide Show view.
pluralsight

Summary: Why Use Stored Procedures?

- **If the statement's optimal execution plan wildly varies**
  - An ad hoc statement will work well
  - A procedure may offer more benefits/possibilities
- **If the statement produces a single, stable plan, regardless of parameter values**
  - Use sp_executesql for forced statement caching and plan reuse
  - A procedure may offer more benefits/possibilities
- **If you want centralized logic, code reuse, and compiled / cached plans (when they're stable) and lots of other options (for when the plans are not stable), use <u>stored procedures</u>**
  - Written by database developers that should
    - Know the data / workload / requirements
    - Know how SQL Server works
  - Provide numerous options to help performance!

Do Not Place Anything in This Space
(Add watermark during editing)
Note: Warning will not appear during Slide Show view.
pluralsight