Reducing Plan Cache Pollution (1 of 3)

- Server setting: Optimize for adhoc workloads
  - On first execution, only the query_hash will go into cache. For the prior query this is only 380 bytes (compared to the 24K of the plan)
  - 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!
  - A lot of limitations to SQL Server detecting this as safe (see Appendix A of the "Plan Caching in SQL Server 2008" whitepaper) but if the plans are actually stable…
  - Consider database setting: forced parameterization
    - Not as highly recommended but if you're finding A LOT of single-use statements that are executed frequently but **with only one query_plan_hash** then this might be great!

Reducing Plan Cache Pollution (2 of 3)

Two primary scenarios to consider

Analyze the plan cache for the number of query plans per query hash (as well as the number of executions)

```
SELECT qs.query_hash
    , COUNT(DISTINCT qs.query_plan_hash) AS [Distinct Plan Count]
    , SUM(qs.EXECUTION_COUNT) AS [Execution Total]
FROM sys.dm_exec_query_stats AS qs
   CROSS APPLY sys.dm_exec_sql_text(sql_handle) AS st
   CROSS APPLY sys.dm_exec_query_plan(plan_handle) AS qp
WHERE st.text LIKE '%member%'
GROUP BY qs.query_hash
ORDER BY [Distinct Plan Count] DESC
```

Scenario 2
Generally, more likely…

Scenario 1

query_hash	# Plans	# Executions
0x04BB791B589774AD	1	6456456
0x1706E9EC3049A95B	**6**	276543
0x5BD9FF487079B335	1	124345
0x6604520C5200ABC0	1	78905
0x77BA5A89C7EBE605	1	14342
0xA078B4BC8768A9A6	1	4567
0xB81E270A58A79D16	1	6

Mostly stable plans (only 1 plan per hash)

Scenario 2

query_hash	# Plans	# Executions
0x04BB791B589774AD	**34**	6456456
0x1706E9EC3049A95B	**6**	276543
0x5BD9FF487079B335	**8**	124345
0x6604520C5200ABC0	**3**	78905
0x77BA5A89C7EBE605	**2**	14342
0xA078B4BC8768A9A6	**9**	4567
0xB81E270A58A79D16	**24**	6

Mostly UNstable plans (multiple plans per hash)

Reducing Plan Cache Pollution (3 of 3)

- Scenario 2: Default parameterization mode: SIMPLE
  - Use "templatized" plan guides to take the few statements that are safe and make them forced (reduced compilation/CPU)

```
EXEC sp_create_plan_guide
    Name_of_plan_guide
    templatized_version_of_safe_query,
    N'TEMPLATE',
    NULL,
    @Parameters,
    N'OPTION(PARAMETERIZATION FORCED)';
```

- Scenario 1: Consider changing parameterization to FORCED
  - Use "templatized" plan guides to take the few statements that are NOT safe and make SIMPLE (recompiled)

```
EXEC sp_create_plan_guide
    Name_of_plan_guide
    templatized_version_of_unsafe_query,
    N'TEMPLATE',
    NULL,
    @Parameters,
    N'OPTION(PARAMETERIZATION SIMPLE)';
```