Optimization and Recompilation

Kimberly L. Tripp
SQLskills.com
@KimberlyLTripp

pluralsight
hardcore dev and IT training

OPTION (RECOMPILE)

- **It's not what it's doing but the fact that the optimal plan varies**
- **If you execute 15 test cases and get 15 different plans with no single plan that's the most common, then the optimal plan wildly varies so use OPTION (RECOMPILE)**
- **Expected pros and cons**
  - Increased CPU in compilation costs
  - Decreased I/O in more efficient plan costs
- **Use OPTION (RECOMPILE) sparingly**
  - Use ONLY when the execution of bad plans takes seconds, minutes, or longer to execute
  - Use ONLY when the optimal plan wildly varies and no specific execution patterns emerge
  - Do not use it "to prevent" parameter sniffing

OPTIMIZE FOR …

- **If you execute 15 test cases and get 2 plans:**
  - If there are 2 primary plans that are fairly split (8 / 7 for example), then create sub-procedures and control the behavior that way
    - Especially if they either don't need to recompile OR only one needs to recompile
  - If there are only 2 plans, where one is 14 of the executions (and, is the norm) then you might use **OPTION (OPTIMIZE FOR …)** but it depends on how bad the performance is the for 15th plan
  - Can you control that through [other] parameters or programmatically?
    - If those executions are very atypical and infrequent, can you programmatically determine those executions and use EXECUTE WITH RECOMPILE instead?
  - If you don't know the data well or you don't want to hard-code a value, but there are only a few anomalies that you don't want to get into cache, use **OPTION (OPTIMIZE FOR UNKNOWN)** to get the "average" plan

Summary: Walkthrough Demo (1)

- **Imagine one stored procedure used to access a date range with parameters (@StartDate and @EndDate)**
  - Typical users: access a date range of 1-2 hours (no more than 1 day of info)
  - Super users: a limited number of managers can enter any date range
- **Problem: the typical user plan is horrible for the super users**
- **Problem: if an optimal plan for the super user gets into cache, then ALL the typical users suffer**
- **Possible solution: in the application, detect that they're a super user and use: EXECUTE <procedure> WITH RECOMPILE**
  - Pro: the typical user case has a plan in cache that's good
  - Pro: the super user plan won't influence what's in cache, nor will it kick the plan that's in cache, out
  - Con: you have to trust that the application will know to do this
  - Con: frequency / number of executions (WITH RECOMPILE) cannot be tracked

Summary: Walkthrough Demo (2)

- **Better solution: in the procedure, detect something more plan-specific and then conditionally use sub-procedures to execute the right code and therefore plan**
  - Pro: each of the sub-procedure's plans will be optimized with parameters intended directly for it
  - Pro: the super user plan won't influence the plan for typical users (and vice versa)
  - Pro: the application doesn't need to know anything has changed
  - Pro: you can see the number of executions of the typical user's single plan and you can see the number of executions of the super user's changing plan (unless the plan is evicted from the cache)
  - Con: you'll have a few more procedures to manage
- **Possible solution: stabilize the plan with indexes?**
  - Can you change the schema?
  - Is this procedure critical enough to warrant it?