Performance Analysis Comparison of Current Production DIAG\_AGGREGATE Script Using Common Table Expressions VS Script Writing to Temp Tables

# Goal

Record and correlate results that will provide performance analysis of the current production script vs a new collection script that eliminates common table expressions. The new script uses temp tables to minimize overload of the tempdb. Also tested is a Merge function vs an Upsert (combined Insert and Update) in simulated real world conditions. The final data will be used to support any conclusions regarding the benefits of one function over the other.

# Methodology

Creation of scripts were designed to test and stress the server using a static data source to insure accurate comparative results. Multiple runs were done and averages were taken.

Each time a script was run, it was run under a new session. SQL Profiler was used to collect data. A separate script was run after each session to query and record the objects created in TempDB related to that session to further record the impact on the server.

The scripts have 3 basic elements;

1. Data load of the sub-source tables
2. Collation of the sub-tables into one source table
3. Integration of the data from the source table into the destination table (either inserts or updates)

The scripts were broken into parts to separately or jointly test each element. Further, varying timelines were used and data was deleted and restored in the source and destination tables to simulate data uploads, inserts and updates.

**Scripts Used (SQL scripts provided. Excel sheet has results listed)**

1. CONFIG – 200 days, generate source results only
   1. Source tables truncated to generate new source, no destination merge
2. CONFIG - all time, full merge only
   1. Uses existing source tables, destination table truncated for full merge
3. CONFIG – 200 days, full upsert only
   1. Uses existing source tables, destination table truncated for full merge
4. CONFIG - all time, generate source and do full merge 10k rows deleted
   1. Truncate source tables, 10k rows deleted to force inserts and updates
5. DIAG - all time, generate source results only
   1. Truncate source table, no update to destination table
6. DIAG – 200 days, full merge only
   1. Uses existing source table, truncate destination to force full insert
7. DIAG – 200 days, full upsert only
   1. Uses existing source table, truncate destination to force full insert
8. DIAG - all time, generate source and do full merge
   1. Truncate source, truncate destination table to force full insert
9. CONFIG - 200 days, generate source and do full merge, run 10 times and get averages
   1. Truncate source, truncate destination table to force full insert
10. DIAG - 200 days, generate source and do full merge, run 10 times and get averages
    1. Truncate source, truncate destination table to force full insert
11. CONFIG - 200 days 10K row upsert only run 10 times and get averages
    1. Uses existing source table, 10k row delete to force inserts and updates
12. CONFIG - 200 days 10K row merge only run 10 times and get averages
    1. Uses existing source table, 10k row delete to force inserts and updates
13. DIAG - 200 days 10K row upsert only run 10 times and get averages
    1. Uses existing source table, 10k row delete to force inserts and updates
14. DIAG - 200 days 10K row merge only run 10 times and get averages
    1. Uses existing source table, 10k row delete to force inserts and updates
15. DIAG Current Production - 200 days 10K row merge Generate Source run 5 times and get averages
    1. Generates new source, 10k rows deleted to force inserts and updates
16. DIAG - 200 days 10K row merge Generate Source run 2 times and get averages
    1. Truncate source tables, 10k rows deleted to force inserts and updates

**SQL Profiler Counters Used**

* **CPU – Use as a comparison value (higher vs lower), actual number value is not important**
* **Reads – Count of read operations against the logical disk**
* **Writes – physical writes performed by current procedure or batch**
* **Duration – the length of time it took in milliseconds to complete procedure or batch**
* **StartTime – listed as Run DT, the starting time of the recorded event**

**Temp DB**

**A script was run to record two counters in Temp DB after each run.**

* user\_objects\_alloc\_page\_count - Number of pages reserved or allocated for user objects by this session
* **internal\_objects\_alloc\_page\_count -** Number of pages reserved or allocated for internal objects by this session

# Comparison and Analysis

**DIAG Current Production vs New Script Comparison run for 200 days (compiled averages). 10k rows deletion.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Insert** | **Update** | **CPU** | **Reads** | **Writes** | **Duration** | **User Obj Pg Cnt** | **Int Obj Pg Cnt** |
| **Current(15)** | 9485 | 11309 | 31213266 | 13143244550 | 13455584 | 8:58:50 | 513 | 19235003 |
| **New(16)** | 11250 | 11258 | 5792362 | 673638873 | 1425477 | 0:31:14 | 257 | 2701824 |

10k rows were deleted to force inserts and updates. Besides being far faster, it’s very obvious that the newer script using temp tables puts far less pressure on the server. There are also big gains in the amount of objects created and therefore memory being consumed in tempDB.

**New DIAG Script Comparison Merge vs Upsert(Insert/Update) run for 200 days (compiled averages). 10k rows deletion.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Insert** | **Update** | **CPU** | **Reads** | **Writes** | **Duration** | **User Obj Pg Cnt** | **Int Obj Pg Cnt** |
| **Upsert(13)** | 9485 | 9470 | 9207 | 303463 | 10204 | 0:00:12 | 281 | 40736 |
| **Merge(14)** | 9485 | 9470 | 9399 | 361774 | 10435 | 0:00:12 | 257 | 40882 |

10k rows were deleted to force inserts and updates. These scripts were run using an existing source so all counters are just for the creation of the destination table. It’s so close that there is no real advantage of one over the other. This corroborates what has been written in several articles.

**New CONFIG Script Comparison Merge vs Upsert(Insert/Update) run for 200 days (compiled averages). 10k rows deletion.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Insert** | **Update** | **CPU** | **Reads** | **Writes** | **Duration** | **User Obj Pg Cnt** | **Int Obj Pg Cnt** |
| **Upsert(11)** | 9476 | 9463 | 7222 | 325856 | 5253 | 0:00:09 | 281 | 35904 |
| **Merge(12)** | 9476 | 9463 | 7379 | 384455 | 6718 | 0:00:09 | 257 | 36056 |

10k rows were deleted to force inserts and updates. These scripts were run using an existing source so all counters are just for the creation of the destination table. Once again its so close that there is no real advantage of one over the other.

**New CONFIG Script Comparison Merge vs Upsert(Insert/Update) run for 200 days (compiled averages).**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Insert** | **Update** | **CPU** | **Reads** | **Writes** | **Duration** | **User Obj Pg Cnt** | **Int Obj Pg Cnt** |
| **Merge(2)** | 299722 |  | 14703 | 2621090 | 9190 | 0:00:16 | 281 | 35560 |
| **Upsert(3)** | 299722 |  | 16500 | 2640399 | 8855 | 0:00:17 | 257 | 68552 |

These scripts were run using an existing source so all counters are just for the creation of the destination table. Destination table was truncated to force an insert only. Once again its close, there is only a slight advantage for merge.