How to Capture the Actual Execution Plan

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:28 AM PST

Contents

How to capture the actual execution plan of a query

The approach described in this article uses the Lightweight query execution statistics profiling infrastructure v3. This feature was introduced with SQL Server 2019 (15.x) and Azure SQL Database and allows collecting row count information for all query executions.

Lightweight profiling v3 comes with a new sys.dm_exec_query_plan_stats function, which returns the equivalent of the last known actual execution plan for most queries (the "last query plan statistics"). It complements the older sys.dm_exec_query_statistics_xml, which returns the query execution plan for in-flight requests with intermediate, transient execution statistics.

While the "lightweight profiling" option is enabled by default on Azure SQL Database, the "last query plan statistics" option needs to be enabled separately at the database level. Either option can be switched on or off by the database administrator. Therefore, before starting to troubleshoot any scenario that requires the actual execution plan, you should explicitly enable both options before taking further steps:

```
ALTER DATABASE SCOPED CONFIGURATION SET LIGHTWEIGHT_QUERY_PROFILING = ON;
ALTER DATABASE SCOPED CONFIGURATION SET LAST_QUERY_PLAN_STATS = ON;
```

The new `sys.dm_exec_query_plan_stats` function only returns reasonable results if a successful execution of a query has been logged on sys.dm_exec_cached_plans, and if the `LAST_QUERY_PLAN_STATS` option had been enabled before the query was executed. It won't return the actual plan if the query had been executed before enabling `LAST_QUERY_PLAN_STATS` or if the plan had already been evicted from the plan cache.

Currently executing query

The query below will provide you with details regarding the active SQL query text and its actual execution plan, the corresponding client application, and some additional details. If you run the query in SQL Server Management Studio (SSMS), you can select the actual query execution plan directly from the resultset to investigate its details within SSMS.

Notes regarding the resultset:

- The column "[actual_query_plan_current]" will show you the in-flight, transient execution details while the underlying query continues to run.

- The column "[actual_query_plan_previous]" will show you the actual details of an execution that has already completed earlier. If the plan is not found on `sys.dm_exec_cached_plans` , you will only see a placeholder.
- The query also returns the `query_hash` and the `plan_handle` values, which you can use with other system views and functions.

```
-- enable lightweight query profiling to get the actual execution plan
-- only captures plan statistics for queries which have started and completed after enabling these options
ALTER DATABASE SCOPED CONFIGURATION SET LIGHTWEIGHT_QUERY_PROFILING = ON;
ALTER DATABASE SCOPED CONFIGURATION SET LAST_QUERY_PLAN_STATS = ON;

SELECT
    [session_id] = R.session_id,
    [request_id] = R.request_id,
    [database_id] = S.database_id,
    [database_name] = D.name,
    [query_hash] = R.query_hash,
    [query_text] = SUBSTRING(T.text, R.statement_start_offset/2 + 1, (CASE WHEN R.statement_end_offset = -1 TH
    [actual_query_plan_current] = P1.query_plan,
    [actual_query_plan_previous] = P2.query_plan,
    [query_plan_handle] = P1.plan_handle,
    [open_transactions] = S.open_transaction_count,
    [login_name] = S.login_name,
    [program_name] = S.program_name,
    [host_name] = S.host_name,
    [start_time] = R.start_time,
    [status] = R.status
FROM sys.dm_exec_requests R
INNER JOIN sys.dm_exec_sessions S ON R.session_id = S.session_id
LEFT JOIN sys.databases D ON S.database_id = D.database_id
CROSS APPLY sys.dm_exec_sql_text(R.sql_handle) T
OUTER APPLY sys.dm_exec_query_statistics_xml(R.session_id) AS P1
OUTER APPLY sys.dm_exec_query_plan_stats(P1.plan_handle) AS P2
ORDER BY R.session_id, R.request_id;
```

Recently executed query

Capturing the actual execution plan from a past execution is not as straightforward because the plan might have already been evicted from sys.dm_exec_cached_plans. This depends on the performance tier and workload of the database: the plan cache on a "Basic" DTU database is small and plan eviction might occur within hours or even minutes, whereas databases with many vCores might hold the plan much longer.

The steps below first try to identify the `query_hash` value of the query that you want to analyze. If you know a significant section of the query, e.g. a table name or a specific join condition or column list, you can filter the SQL query text for that and then pick the `query_hash` that matches the query in question. Once you have the `query_hash` , the next step is to retrieve the `plan_handle` value from `sys.dm_exec_query_stats` , then use this `plan_handle` to get the actual execution plan.

```
-- Check QDS - filter for the SQL text or the query_hash, whichever you know about the query
SELECT --TOP 50
    P.query_id, P.plan_id, P.last_execution_time,
    RS.count_executions, [max_duration_ms] = RS.max_duration / 1000, [max_cpu_time_ms] = RS.max_cpu_time / 100
    Q.query_hash, T.query_sql_text, [estimated_query_plan] = CAST(P.query_plan as xml)
FROM
sys.query_store_plan AS P
INNER JOIN sys.query_store_query AS Q ON P.query_id = Q.query_id
INNER JOIN sys.query_store_query_text AS T ON T.query_text_id = Q.query_text_id
INNER JOIN sys.query_store_runtime_stats AS RS ON P.plan_id = RS.plan_id
WHERE T.query_sql_text LIKE '%select top 1000 %'
--WHERE query_hash = 0xB8D25868C9E032A0
ORDER BY P.last_execution_time DESC

-- Get the plan_handle, once you know the query_hash from QDS
SELECT QS.query_hash, QS.plan_handle,
    [max_duration_ms] = QS.max_elapsed_time / 1000,
    [max_cpu_time_ms] = QS.max_worker_time / 1000,
    [max_rowcount] = QS.max_rows, T.text,
    [estimated_query_plan] = P.query_plan
FROM sys.dm_exec_query_stats QS
OUTER APPLY sys.dm_exec_sql_text(QS.sql_handle) AS T
OUTER APPLY sys.dm_exec_query_plan(QS.plan_handle) AS P
WHERE QS.query_hash = 0xB8D25868C9E032A0;

-- Select the execution plans either through the SQL text or the plan_handle
SELECT
    CP.cacheobjtype, CP.objtype, CP.plan_handle, CP.parent_plan_handle, CP.size_in_bytes, CP.usecounts,
    [sql_text] = T.text,
    [estimated_query_plan] = P1.query_plan,
    [actual_query_plan] = P2.query_plan
FROM sys.dm_exec_cached_plans CP
OUTER APPLY sys.dm_exec_sql_text(CP.plan_handle) T
OUTER APPLY sys.dm_exec_query_plan(CP.plan_handle) AS P1
OUTER APPLY sys.dm_exec_query_plan_stats(CP.plan_handle) AS P2
--WHERE T.text LIKE '%select top 1000 %'
WHERE CP.plan_handle = 0x060005008C263E29505CF5194A0200000100000000000000000000000000000000000000000000000
ORDER BY usecounts DESC;

-- replace the values for SQL text, query_hash, and plan_handle with your own values
```

Resources

- Query Profiling Infrastructure
- Lightweight query execution statistics profiling infrastructure v3
- Using Extended Events to capture an Actual Execution Plan

How good have you found this content?