**Adaptive Query Processing in SQL Server**

**By** [**Anoop Kumar**](https://www.databasejournal.com/feedback/https:/www.databasejournal.com/features/mssql/adaptive-query-processing-in-sql-server.html)

**FROM:** [**https://www.databasejournal.com/features/mssql/adaptive-query-processing-in-sql-server.html**](https://www.databasejournal.com/features/mssql/adaptive-query-processing-in-sql-server.html)

**Introduction**

Adaptive query processing is the latest, improved query processing feature introduced in the SQL Server database engine. This method is available in SQL Server (starting with SQL Server 2017 (14.x)) and Azure SQL Database.

Query performance is always a subject of research and innovation. Every so often, Microsoft includes new feature and functionality in SQL Server to improve query performance to help their users. In SQL Server 2016, Microsoft introduced Query Store. Query Store monitors query performance and provides methods to choose better query plans for execution. Query store is a feature to persist query execution plans and analyze the history of query plans to identify the queries that can cause performance bottlenecks. Any point in time these plans can be reviewed, and the user can use plan forcing to enforce the query processor to select and use a specific query plan for execution. You can read more about Query Store in [Monitor Query Performance Using Query Store in SQL Server](https://www.databasejournal.com/features/mssql/monitor-query-performance-using-query-store-in-sql-server.html).

In SQL Server 2017, Microsoft went one step further and worked on improving the query plan quality. This improved query plan quality is Adaptive Query Processing.

**Query Execution Plan**

Prior to SQL Server 2017, query processing was a uniform process with a certain set of steps. The SQL Server query optimizer first generates a set of feasible query plans for a query or batch of T-SQL code submitted by the database engine to query optimizer. Based on the cost of the query plan, optimizer selects the lowest-cost query plan.

**Related Articles**

* [9 New Features with SQL Server 2017](https://www.databasejournal.com/features/mssql/slideshows/9-new-features-with-sql-server-2017.html)
* [Understanding Migration from SQL Server to SQL Server 2017 using Data Migration Assistant](https://www.databasejournal.com/features/mssql/understanding-migration-from-sql-server-to-sql-server-2017-using-data-migration-assistant.html)

Once the query execution plan is ready and available to execute, SQL Server storage engines get the query plan and execute it based on the actual query plan and return the data set.

This end to end process (query plan generation, and submitting the plan for processing) once started, has to run to completion. Please see [Understanding a SQL Server Query Execution Plan](https://www.developer.com/db/understanding-a-sql-server-query-execution-plan.html) for more information.

In this process, query performance depends on the overall quality of the query plan, join order, operation order and selection of physical algorithm like nested loop join or hash join, etc. Also, cardinality estimates play a vital role to decide the quality of the query plan. Cardinality represents the estimated number of rows returned on execution of a query.

The accuracy of cardinality drives a better quality of query execution plan and more accurate allocation of memory, CPU to the query. This improvement leads to a better and improved execution of query, known as adaptive query processing. We will discuss exactly how SQL Server achieves this in rest of the section,.

**Adaptive Query Processing**

Prior to SQL Server 2017, the behavior of Query Optimizer was not optimal; it bound to select the lowest cost query plan for execution, despite poor initial assumptions in estimating inaccurate cardinality, which led to bad query performance.

In Adaptive Query Processing, SQL Server optimized the query optimizer to generate more accurate and quality query plans with more accurate cardinality. Adapting Query Processing gets a better-informed query plan. There are three new techniques for adapting to application workload characteristics.

**Adapting Query Processing Techniques**

1. **Batch Mode Memory Grant Feedback**. This technique helps in allocating the required memory to fit in all returning rows. The right allocation of memory helps in, 1) reducing excessive memory grants that avoid concurrency issues and, 2) fixing underestimated memory grants that avoid expensive spills to disk.

Query performance suffers when memory allocation sizes are not sized correctly. In Batch Mode Memory Grant Feedback, optimizer repeats the workload and recalculates the actual memory required for a query and then updates the grant value for the cached plan.  When an identical query statement is executed, the query uses the revised memory grant size and improves the query performance.

1. **Batch Mode Adaptive Join**. This technique provides the choice of algorithm selection and the query plan dynamically switches to a better join strategy during plan execution.  This defers either hash join or nested loop join selection until after first input has been scanned.

The new operator in the family, Adaptive Join operator helps in defining a threshold. This threshold is used to compare with the row count of the build join input; if the row count of the build join input is less than the threshold, a nested loop join would be better than a hash join and query plan switches to nested loop join. Otherwise, without any switches query plan continues with a hash join.

This technique mainly helps the workload with frequent fluctuations between small and large join input scans.

1. **Interleaved Execution**. Interleaved execution is applicable to multi-statement table valued functions (MSTVFs) in SQL Server 2017 because MSTVFs have a fixed cardinality guess of “100” in SQL Server 2014 and SQL Server 2016, and an estimate of “1” for earlier versions, which leads to passing through inaccurate cardinality estimates to the query plan. In Interleaved execution technique, whenever SQL Server identifies MSTVFs during the optimization process, it pauses the optimization, executes the applicable subtree first, gets accurate cardinality estimates and then resumes optimization for rest of the operations. This technique helps in getting actual row counts from MSTVFs instead of fixed cardinality estimates to make plan optimizations downstream from MSTFs references; this results in improvement in workload performance. Also, Interleaved execution enables plans to adapt based on the revised cardinality estimates as MSTVFs changes the unidirectional boundary of single-query execution between the optimization and execution phase.

**Additional facts for Adaptive Query Processing Techniques**

There are certain facts associated with each Adaptive Query Processing technique while optimizer improves the query performance using these techniques.

- Advertisement -

**Batch Mode Memory Grant Feedback.** Feedback is not persisted once a plan is evicted from cache. You will not be able to get the history of the query plan in Query Store, as Memory Grant Feedback will change only the cached plan and changes are currently not captured in the Query Store. Also, in case of failover, feedback will be lost.

**Batch Mode Adaptive Join.** SQL Server can dynamically switch between the two types of joins and always choose between nested loop and hash join; merge join is not currently part of the adaptive join. Also, adaptive joins introduce a higher memory requirement than an indexed nested loop Join equivalent plan.

**Interleaved Execution.** There are some restrictions; 1) the MSTVFs must be read-only, 2) the MSTVFs cannot be used inside of a CROSS APPLY operation, 3) the MSTVFs are not eligible for interleaved execution if they do not use runtime constants.

**Configure Adapting Query Processing**

There are multiple options to configure Adaptive Query Processing. These options are as follows:

**Option-1**: SQL Server by default enables Adaptive Query Processing and executes the queries under compatibility level 140.

**Option-2**: T-SQL is available to enable and disable all three Adaptive Query Processing techniques:

T-SQL to enable Adaptive Query Processing techniques

1. ALTER DATABASE SCOPED CONFIGURATION SET DISABLE\_BATCH\_MODE\_MEMORY\_GRANT\_FEEDBACK = ON;
2. ALTER DATABASE SCOPED CONFIGURATION SET DISABLE\_BATCH\_MODE\_ADAPTIVE\_JOINS = ON;
3. ALTER DATABASE SCOPED CONFIGURATION SET DISABLE\_INTERLEAVED\_EXECUTION\_TVF = ON;

       T-SQL to disable Adaptive Query Processing techniques

1. ALTER DATABASE SCOPED CONFIGURATION SET DISABLE\_BATCH\_MODE\_MEMORY\_GRANT\_FEEDBACK = OFF;
2. ALTER DATABASE SCOPED CONFIGURATION SET DISABLE\_BATCH\_MODE\_ADAPTIVE\_JOINS = OFF;
3. ALTER DATABASE SCOPED CONFIGURATION SET DISABLE\_INTERLEAVED\_EXECUTION\_TVF = OFF;

**Option-3**:Enforce Adaptive Query Processing technique using query hint. Query hint takes precedence over a database scoped configuration and other settings.

1. OPTION (USE HINT ('DISABLE\_BATCH\_MODE\_MEMORY\_GRANT\_FEEDBACK'));
2. OPTION (USE HINT('DISABLE\_BATCH\_MODE\_ADAPTIVE\_JOINS'));
3. OPTION (USE HINT('DISABLE\_INTERLEAVED\_EXECUTION\_TVF'));

**Summary**

The adaptive query processing improves the quality of a query plan. This helps in selecting the right join, right order of operation and more accurate memory allocation to fit all rows. The three techniques of adaptive query processing make SQL Server 2017 significantly faster at processing the workload. Also, adaptive query processing provides significant improvements without refactoring T-SQL code.

Overall, adaptive query processing is a great addition in SQL Server and enables SQL Server to generate well informed quality query plans to manage application workload processing.