**Describe intelligent query processing**

In SQL Server 2017 and 2019, and with Azure SQL, Microsoft has introduced many new features into compatibility levels 140 and 150. Many of these features correct what were formerly anti-patterns like using user defined scalar value functions and using table variables.

These features break down into a few families of features:

Intelligent query processing includes features that improve existing workload performance with minimal implementation effort.

To make workloads automatically eligible for intelligent query processing, change the applicable database compatibility level to 150. For example:

ALTER DATABASE [WideWorldImportersDW] SET COMPATIBILITY\_LEVEL = 150;

**Adaptive query processing**

Adaptive query processing includes many options that make query processing more dynamic, based on the execution context of a query. These options include several features that enhance the processing of queries.

* **Adaptive Joins –** the database engine defers choice of join between hash and nested loops based in the number of rows going into the join. Adaptive joins currently only work in batch execution mode.
* **Interleaved Execution –** Currently this feature supports multi-statement table-valued functions (MSTVF). Prior to SQL Server 2017, MSTVFs used a fixed row estimate of either one or 100 rows, depending on the version SQL Server. This estimate could lead to suboptimal query plans if the function returned many more rows. An actual row count is generated from the MSTVF before the rest of the plan is compiled with interleaved execution.
* **Memory Grant Feedback –** SQL Server generates a memory grant in the initial plan of the query, based on row count estimates from statistics. Severe data skew could lead to either over- or under-estimates of row counts, which can cause over-grants of memory that decrease concurrency, or under-grants, which can cause the query to spill data to tempdb. With Memory Grant Feedback, SQL Server detects these conditions and decreases or increases the amount of memory granted to the query to either avoid the spill or overallocation.

These features are all automatically enabled under compatibility mode 150 and require no other changes to enable.

**Table variable deferred compilation**

Like MSTVFs, table variables in SQL Server execution plans carry a fixed row count estimate of one row. Much like MSTVFs, this fixed estimate led to poor performance when the variable had a much larger row count than expected. With SQL Server 2019, table variables are now analyzed and have an actual row count. Deferred compilation is similar in nature to interleaved execution for MSTVFs, except that it's performed at the first compilation of the query rather than dynamically within the execution plan.

**Batch mode on row store**

Batch execution mode allows data to be processed in batches instead of row by row. Queries that incur significant CPU costs for calculations and aggregations will see the largest benefit from this processing model. By separating batch processing and columnstore indexes, more workloads can benefit from batch mode processing.

**Scalar user-defined function inlining**

In older versions of SQL Server, scalar functions performed poorly for several reasons. Scalar functions were executed iteratively, effectively processing one row at a time. They didn't have proper cost estimation in an execution plan, and they didn't allow parallelism in a query plan. With user-defined function inlining, these functions are transformed into scalar subqueries in place of the user-defined function operator in the execution plan. This transformation can lead to significant gains in performance for queries that involve scalar function calls.

**Approximate count distinct**

A common data warehouse query pattern is to execute a distinct count of orders or users. This query pattern can be expensive against a large table. Approximate count distinct introduces a much faster approach to gathering a distinct count by grouping rows. This function guarantees a 2% error rate with a 97% confidence interval.