5 Ways to Examine Cardinality Metrics

  By [monica](https://sqlespresso.com/author/monica/) [May 2, 2018](https://sqlespresso.com/2018/05/02/5-ways-to-examine-cardinality-metrics/) May 7, 2018 [Performance Tuning](https://sqlespresso.com/category/performance-tuning/)

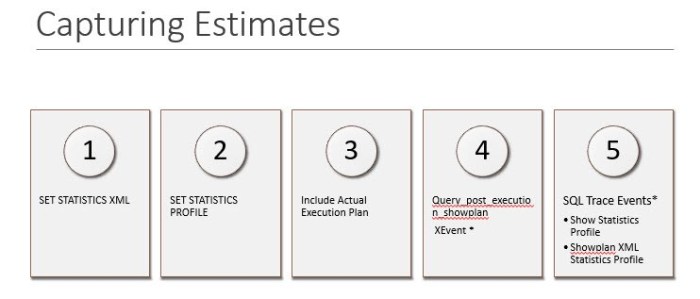
from: <https://sqlespresso.com/2018/05/02/5-ways-to-examine-cardinality-metrics/>

In SQL Server the query optimizer uses a [cardinality](https://en.wikipedia.org/wiki/Cardinality_(data_modeling)) estimator to answer data SELECTIVITY questions like the ones below. The optimizer uses this cardinality data, which relies heavy on statistics to get the answers and calculate a cost estimate. It takes the #of Rows to Satisfy a Predicate/Total # of Input Rows.

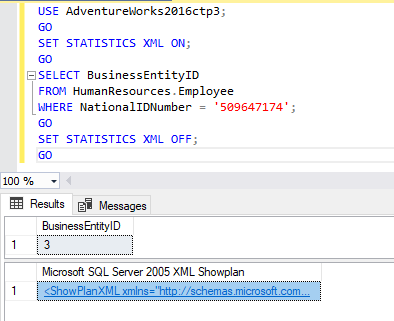
* How many rows will satisfy a single **filter predicate?** Multiple?
  + **WHERE**
* How many rows will satisfy a **join predicate** between two tables?
  + **JOIN**
* How many distinct values do we expect from a specific column?
  + **DISTINCT**
  + **GROUP BY**

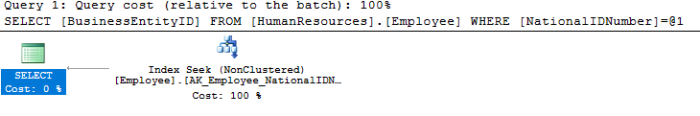
From this estimate the optimizer is able to find an efficient execution plan that fulfills a query request. You will note that query optimizer, including SQL Server does not generate **all** possible plans, it creates several then choose a good enough plan to return results requested. This behavior gives us the most efficient processing strategy (generating plans is computationally expensive, hence the limited number) for executing queries across a wide variety of workloads.

Here I will go over five ways we can see the estimations it creates. Some of which I am sure you are already aware of and hopefully use daily. Note that items four and five on the picture below are starred. This denotes that they are very resource intensive to run. I highly suggest if you use those options you only keep them running for a very short interval to capture what you want and then turn them off.

[](https://sqlespresso.com/2018/05/02/5-ways-to-examine-cardinality-metrics/ce/)

First, we will look at **SET STATISTICS XML**. It’s set in a simple T-SQL Statement and remains on until set to off. The results return an XML link to the actual execution plan.

[](https://sqlespresso.com/2018/05/02/5-ways-to-examine-cardinality-metrics/ce6/)

[](https://sqlespresso.com/2018/05/02/5-ways-to-examine-cardinality-metrics/ce7/)

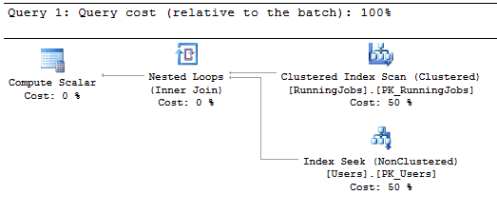
A second option is to use **statistics profiling**. This was introduced in SQL Server 2014 and is easily set by using **SET STATISTICS PROFILE ON or** enable query profiling globally using **DBCC TRACEON (7412, -1)**. This trace flag is only available in SQL Server 2016 SP1 and above.  Selecting from the dynamic management view (DMV) [*Sys.dm\_exec\_query\_profiles*](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-exec-query-profiles-transact-sql?view=sql-server-2017) you can do real time query execution progress monitoring while the query is running.  This option will return estimated and actual rows by operator.

Including the **actual execution plan** is something we should all already being using (in testing–remember it’s expensive to capture actual plans) to see runtime information, actual resource usage metrics and runtime warnings.

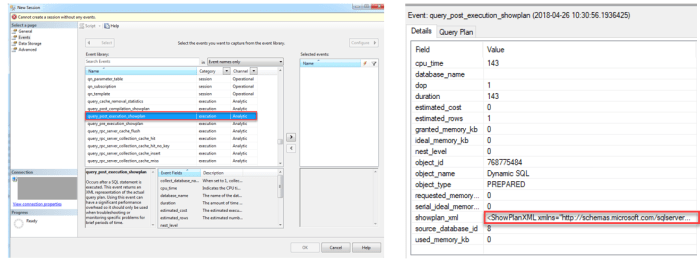
There are a variety of methods for displaying or triggering graphical execution plans

* **Under the Query**menu in SSMS, click **Include Actual Execution Plan**
* **Include Actual Execution Plan**toolbar button
* Use the **SET STATISTICS XML ON** T-SQL statement

Costs are displayed for each step, relative to the total cost of the query plan

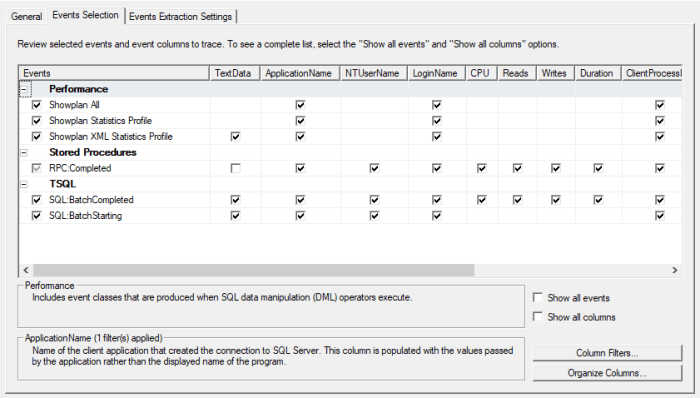
[](https://sqlespresso.com/2018/05/02/5-ways-to-examine-cardinality-metrics/ce2/)

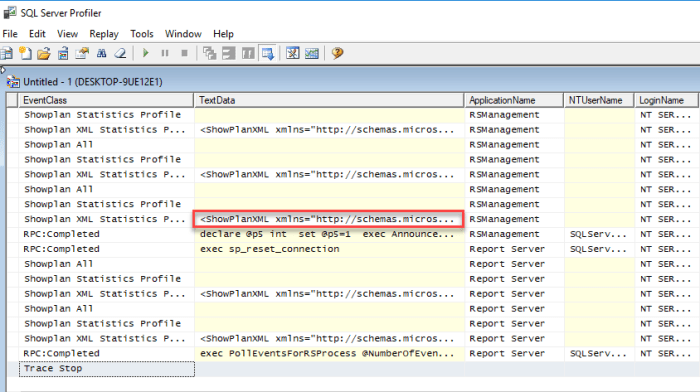
If you are a fan of Extended Events you may already be familiar with **Query\_post\_execution\_showplan**. It also returns an XML link to the actual execution plan along with estimate counts and costs.

[](https://sqlespresso.com/2018/05/02/5-ways-to-examine-cardinality-metrics/ce3-2/)

Lastly, for those die-hard “Profiler for Life” DBA’s there are traces you can add to capture this information as well. When creating the trace look under **Performance** and choose the follow.

* **Showplan ALL**
* **Showplan Statistics Profile**
* **Showplan XML Statistics Profile.**

[](https://sqlespresso.com/2018/05/02/5-ways-to-examine-cardinality-metrics/ce5/)

[](https://sqlespresso.com/2018/05/02/5-ways-to-examine-cardinality-metrics/ce8/)

It’s important to know these ways to get to execution plans and cardinality estimates. When diving into query performance issues these are key to discovering your bottle necks and plan defficiencies.

Now, I know, in this blog I refer to the cardinality estimator’s reliance on statistics for its calculation, but I really didn’t go into statistics much. Look to next week as I dive into those. Starting with DBCC SHOW\_STATISTICS and how over and under estimations due to bad statics can lead to inefficient plans.