

Session Highlights from SQL Saturday #55

Visit [SQL Saturday on the web](#) for information on upcoming events

[Ramona Maxwell](#) ©2010

[Tara Kizer](#) of [Qualcomm](#) presented **Performance Tuning with Traces**

Ms. Kizer emphasized first and repeatedly the importance of never running tuning traces on your production database server – else it soon won't be classified as 'production' anymore. She offered both entry level and advanced information on using SQL Server Management Studio's [SSMS's] trace tools including SQL Profiler, Performance Monitor and command-line options. While she deemed a server-side trace least expensive she pointed out the advantages of using a GUI in Profiler. She recommends getting a baseline for your system using Performance Monitor before starting to search out problem areas and noted that [sql-server-performance.com](#) is a great resource for how to begin.

A few of her tips for traces included outputting the trace data to a table where you can index columns that are good indicators of performance issues such as CPU, duration, writes and reads as well as query specific baselines (i.e. SELECT x WHERE reads >5000) to pinpoint problems. While a trace can be managed by SQL Server Agent she included the reminder that it had to be specifically stopped with a stored procedure or it would run endlessly. Basic coding information can be found on [MSDN](#). She also recommended a utility called ClearTrace available at [ScaleSql.com](#). To avoid gaps in trace information she schedules traces in a slightly overlapping pattern so that as one is concluding and storing its findings a new one is already starting. She lays claim to more than 29,000 posts on [sqlteam.com](#) and if her tutorial today is any indicator, it's well worth looking up the ones you need.

[Dean Richards](#) of [Confio](#) presented **Query Tuning – Getting it Right the First Time**

Mr. Richards provided an impressive demonstration of the advantages of hand-tuning queries over blindly accepting SSMS's recommendations as to which indexes will improve the execution plan. He offered guidelines for identifying which queries to tune with possible hints being user feedback, excessive CPU or I/O activity, table or index scans and data from SQL Profiler traces [as described in Ms. Kizer's presentation above]. Within dynamic management views [total_elapsed_time](#), ordered DESC, is a KPI he favors. A simple explanation of how to collect the I/O metrics can be found on [Technet](#).

Once the slow query has been identified he suggests figuring out what the query is spending the excess time on. Are there locks? Excessive disk reads? Where are the wait times? If a stored procedure is running within a T-SQL batch of code using [statement_start_offset](#) and [statement_end_offset](#) can show the cost of the procedure. He offered tips for reading the execution plan such as starting from the bottom reading right-to-left and noting that thickened lines connecting objects in the execution plan graphic indicate a lot of data traversing between them. He offered the intriguing insight that SSMS's 'actual' execution plan in truth cannot be called that until after the query has run through and its particular statements are in the procedure cache. Therefore he suggests using [sys.dm_exec_query_plan](#) to obtain an instantaneous real plan overview.

He suggested that time to execute the query was not the only useful metric for judging how well a query runs since other system processes, such as a virus scan, could slow overall system performance and make the query's performance seem worse than it really is. To gauge how much heavy lifting the query engine is doing he suggests monitoring logical disk reads, and it was by that metric that his tuning methods really excelled. In one example of querying payroll data he took a query from in excess 65,000+ logical reads to 46! Each of his examples used a multifaceted approach from looking at how the query is written to modifying indexes while carefully examining the impact of each adjustment on the execution plan. He suggests carefully examining the business logic in order to look for fields to query that have the highest degree of selectivity and that are index-able, thereby avoiding table scans. If you are able to attend one of his presentations in person you will see excellent step-by-step examples of this.

Finally he commented that a top resource for him has been a book by Dan Tow titled [SQL Tuning](#) that teaches a method of SQL diagramming and he led us through several examples of the usefulness of this approach, which uses mathematical calculations to form a visual representation of the cost of various query methods.

[Bill Sheldon](#), contributing editor at [SQL Server Magazine](#), presented **Looking at LINQ**

Mr. Sheldon highlighted the surprising information that LINQ to SQL is not a forward-moving technology in the current Microsoft .NET master plan and while current uses of LINQ to SQL based on .NET 3.5 will likely be supported for some time to come, new [SQL data connections](#) using [ADO.NET 4.0](#) and forward should be built using [Entity Framework](#) [EFW] instead. He stressed that [LINQ](#) is still a key tool within Visual Studio and that SQL is the only language that won't be playing as much in this particular yard. According to Mr. Sheldon the change to using EFW is thought to provide better data abstraction and be platform independent, although it will be more complex. He warned against using EFW in .NET 3.5, where it was not as stable as it is currently.

Please respect that all product, company and website names in this article are copyrighted and trademarked by their respective owners. Their use is for reference only. Thank you, Ramona Maxwell ~ [Return to SQLSolver's main page.](#)