



*The World's Largest Community
of SQL Server Professionals*

Identifying SQL Server Performance Problems Using SQL Trace

Brad M. McGehee

Director of DBA Education

Red Gate Software

www.bradmcgehee.com/presentations

My Assumptions About You

- You are probably a DBA (production or developer) who has at least one year's experience using SQL Server.
- You have a basic understanding of how to use SQL Server Profiler.
- You have a basic understanding of how to write T-SQL code.
- You have little or no SQL Trace experience.

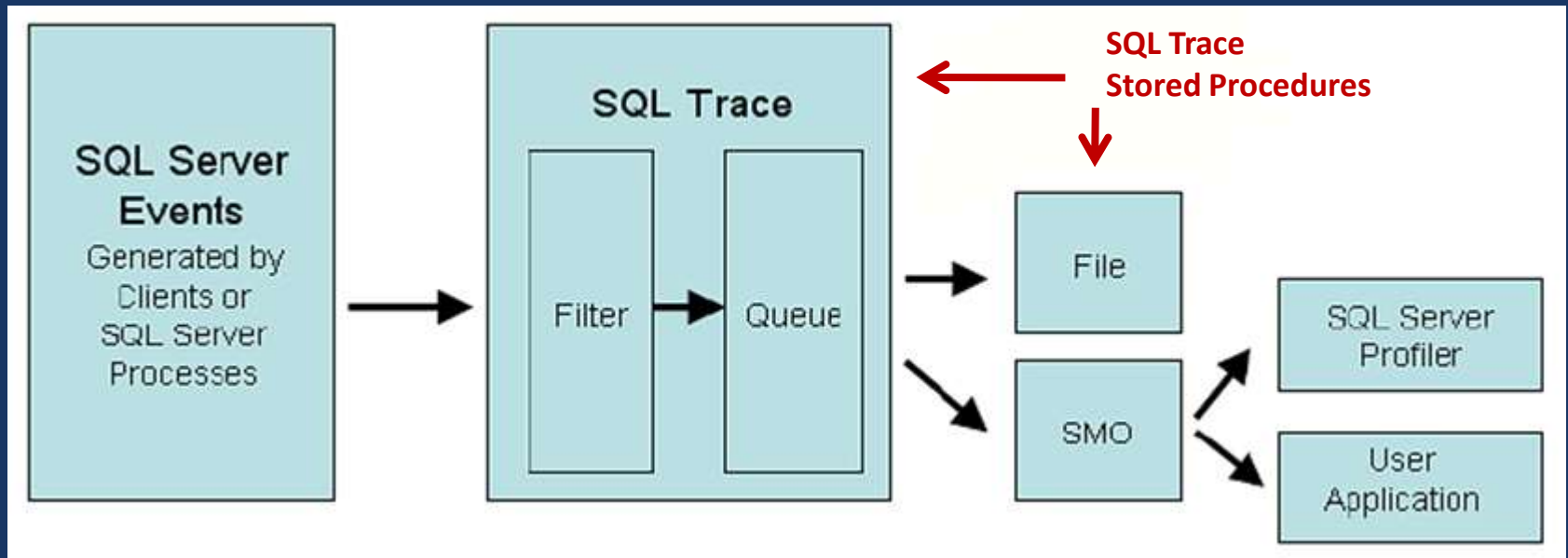
What We Are Going to Learn Today

- What is SQL Trace
- Pros and Cons of Using SQL Trace
- Overview of How SQL Trace Works
- Digging Into the SQL Trace Stored Procedures
- How to Manually Create Your Own SQL Trace Script
- How to Use SQL Server Profiler to Create SQL Trace Scripts

What is SQL Trace

- SQL Trace is an internal component of the database engine that allows predefined SQL Server events (and related data) to be collected and stored for later analysis.
- SQL Trace can be accessed directly using the SQL Server Profiler GUI, SMO, and system stored procedures.
- When *system stored procedures* are used to collect SQL Trace events, this is often referred to as a *server-side trace*, the focus of this session.

SQL Trace Architecture



Pros and Cons of Using SQL Trace

- Pros

- Uses less overhead than the Profiler GUI.
- Allows you to programmatically capture SQL Trace data using system stored procedures, no GUI required.
- Traces can be started and stopped with SQL Server Agent jobs.

- Cons

- A steeper learning curve is required to master SQL Trace. On the other hand, if you are a T-SQL expert already, then the learning curve is not that bad.
- Generally speaking, it initially takes a little more setup work to create a trace using system stored procedures than by using the Profiler GUI. On the other hand, once a trace script has been created, it can be used over and over.
- The Profiler GUI's Grouping feature is not available, but can be duplicated using T-SQL commands after events are collected and stored.
- Profiler trace templates cannot be used (directly).

Overview of How SQL Trace Works

- Create and Start the Trace
 - Create a new trace (and specify its output file location) using **sp_trace_create**
 - Select the events and data columns for the trace using **sp_trace_setevent**
 - Select and create (if any) filters to be used by the trace using **sp_trace_setfilter**
 - Start the trace using **sp_trace_setstatus**
- Stop and Close the Trace
 - Stop the trace using **sp_trace_setstatus** (can be stopped or started, like pausing)
 - Close the trace using **sp_trace_setstatus** (and the trace file is ready to view)
- View Trace Data (Multiple Ways to View & Analyze Data)
 - View and analyze data from within the Profiler GUI
 - View and analyze data using **fn_trace_gettable** function and T-SQL
 - Import the trace data into a table, then view and analyze it using T-SQL
 - Use third-party tool, such as RML Utilities, ClearTrace, and others

sp_trace_create

```
sp_trace_create [ @traceid = ] trace_id OUTPUT  
    , [ @options = ] option_value  
    , [ @tracefile = ] 'trace_file'  
    [ , [ @maxfilesize = ] max_file_size ]  
    [ , [ @stoptime = ] 'stop_time' ]  
    [ , [ @filecount = ] 'max_rollover_files' ]
```

This SP is used once to create the trace, along with specifying some initial settings.

sp_trace_setevent

```
sp_trace_setevent [ @traceid = ] trace_id  
    , [ @eventid = ] event_id  
    , [ @columnid = ] column_id  
    , [ @on = ] on
```

This SP has to be executed for every combination of event and data column that you want captured. For example, if you want to capture 15 data columns for 5 events, then you would have to execute this stored procedure 75 times.

sp_trace_setfilter

```
sp_trace_setfilter [ @traceid = ] trace_id  
    , [ @columnid = ] column_id  
    , [ @logical_operator = ] logical_operator  
    , [ @comparison_operator = ]  
comparison_operator  
    , [ @value = ] value
```

This SP only needs to be used if you want to use a filter. This SP has to be repeated for every filter you want to create.

sp_trace_setstatus

```
sp_trace_setstatus [ @traceid = ] trace_id ,  
  [ @status = ] status
```

This SP is used to:

- Start a trace
- Stop a trace
- Close a trace

Quick Review

1. Create a new trace (and specify its file location) using **sp_trace_create**
2. Select the events and data columns for the trace using **sp_trace_setevent**
3. Select and create (if any) filters to be used by the trace using **sp_trace_setfilter**
4. Start the trace using **sp_trace_setstatus**.
 - The trace begins and is stored on disk
5. Stop the trace using **sp_trace_setstatus**
6. Close the trace using **sp_trace_setstatus**
 - The trace is now on disk in a file and ready to view and analyze

fn_trace_gettable

- Data collected by a trace in a trace file can be displayed using the fn_trace_gettable function using standard SELECT statements.

fn_trace_gettable (filename, number_of_files)

Where:

- Filename is the path and filename of the file you want to view.
- Number_of_files is the number of files, if more than one is available, included as part of the trace.

Demo: Putting All the Pieces Together

- Let's assume that we want to capture these five events in order to capture the queries running on a server.
 - RPC:Completed
 - SP:StmtCompleted
 - SQL:BatchStarting
 - SQL:BatchCompleted
 - Showplan XML

Demo Continued

- We also want to collect the following 15 data columns for the previous 5 events.
 - Duration
 - ObjectName
 - TextData
 - CPU
 - Reads
 - Writes
 - IntegerData
 - DatabaseName
 - ApplicationName
 - StartTime
 - EndTime
 - SPID
 - LoginName
 - EventSequence
 - BinaryData

Isn't There an Easier Way to Do This?

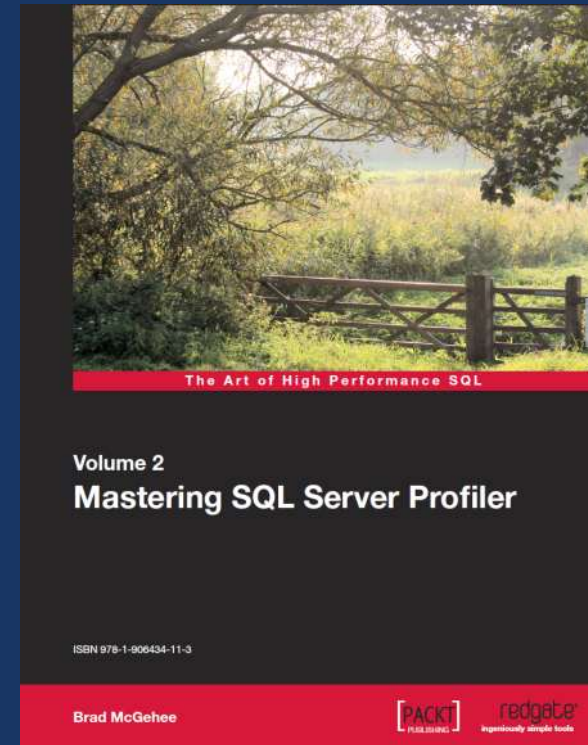
- Writing your own T-SQL code gives you lots of flexibility, but it is a lot of work.
- Instead, you can use Profiler to create a SQL Trace script for you. It produces code that creates and starts the trace, but you will have to write the code to stop the trace.
- Demo

Take Aways From This Session

- Traces can be done using either the Profiler GUI or through server-side traces.
- Server-side traces have less overhead than Profiler GUI traces, and are created programmatically.
- The easiest way to create a server-side trace is to have the Profiler GUI create the code for you. At this point, you can refine the code anyway you like, and even schedule it to run using a SQL Server job.
- Often, using the `fn_trace_gettable` function is the easiest way to query trace results.
- If you haven't tried a server-side trace before, give it a try.

E-books, Websites, Slides & More

- Free E-books on SQL Server:
 - www.sqlservercentral.com/Books
- Check these websites out:
 - www.SQLServerCentral.com
 - www.Simple-Talk.com
- Blogs:
 - www.bradmcgehee.com
 - www.twitter.com/bradmcgehee
- Contact me at:
 - bradmcgehee@hotmail.com





*The World's Largest Community
of SQL Server Professionals*

Thanks for Attending

Visit www.sqlservercentral.com for free SQL Server eBooks, articles, videos, blogs, news, and more.

Please Don't Forget to Turn in Your Evaluations