Tracing SQL Server Activity

Module Overview

- Tracing SQL Server Workload Activity
- Using Traces

Lesson 1: Tracing SQL Server Workload Activity

- SQL Server Profiler
- Trace Events
- Trace Columns and Filters
- Trace Templates
- Viewing Trace Files
- Demonstration: Using SQL Server Profiler
- SQL Trace
- Demonstration: Using SQL Trace

SQL Server Profiler

- Tool used to trace activity against SQL Server
- Based on the SQL Trace programming interface
- Used in many scenarios such as debugging, performance monitoring, and deadlock monitoring
- Can replay functionality for stress testing

Trace Events

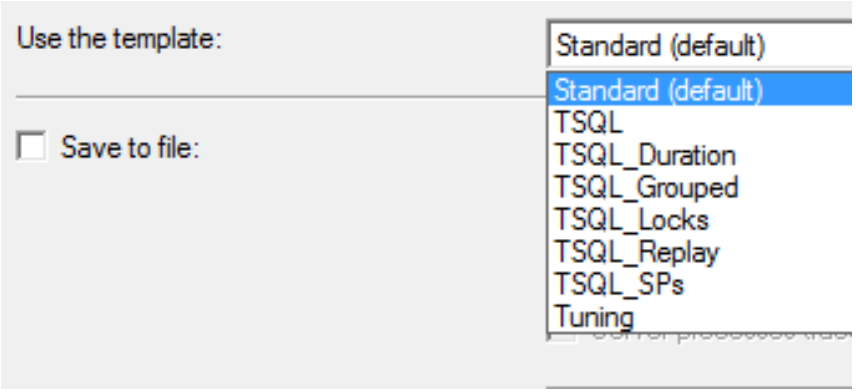
Event	Description
SQL:BatchCompleted	Fires when a batch of Transact-SQL statements is completed
SQL:StmtCompleted	Fires when an individual Transact-SQL statement is completed
RPC:Completed	Fires when a stored procedure completes execution
Audit Login/Audit Logout	Fires when a login or logout event occurs
Deadlock Graph	Fires when deadlocks occur

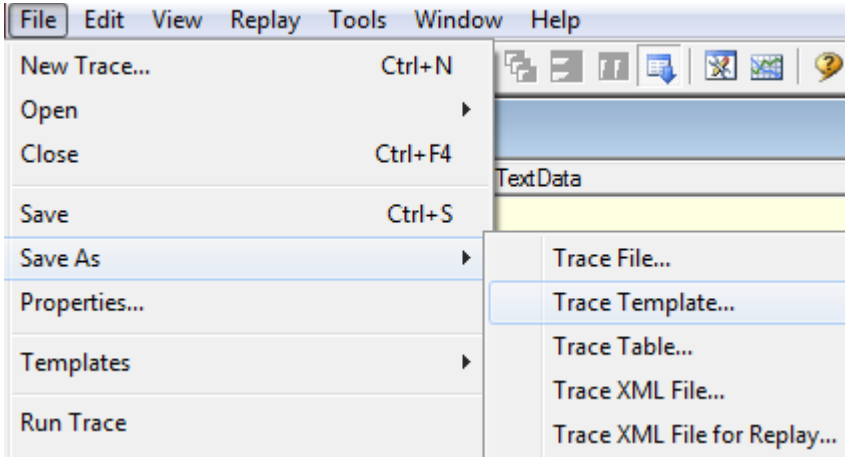
Trace Columns and Filters

- Columns contain data values that you can capture
  - Not all columns contain data for all events
- Select specific columns to include in the trace
- Filter the trace based on column values
  - Filter is only applied when the filtered column is supported by the selected event

Trace Templates

- Predefined sets of event classes and trace columns:
  - Use SQL Server predefined templates
  - Create your own templates
  - Modify predefined templates and save as your own





Viewing Trace Files

- Analyze traces by:
  - Opening in SQL Server Profiler
  - Importing into a SQL Server table

```
CREATE TABLE dbo.tracetable
(
  TextData nvarchar(max) NULL,
  BinaryData varbinary(max) NULL,
  ...
);
INSERT INTO dbo.tracetable
SELECT * FROM
fn_trace_gettable('L:\Traces\adworks.trc',default);
```

Demonstration: Using SQL Server Profiler

In this demonstration, you will see how to:

- Use SQL Server Profiler to create a trace
- Run a trace and view the results

SQL Trace

SQL Trace	SQL Server Profiler
Traces: Are defined by stored procedures Run directly within the database engine Write events to files or SMO Used for: Long-term monitoring Performance-critical traces Large traces	Traces: Are defined by using a graphical tool Utilize SQL Trace Write to files or database tables Used for: Debugging on test systems Short-term analysis Small traces

Demonstration: Using SQL Trace

In this demonstration, you will see how to:

- Export a trace definition
- Configure and run a trace

Lesson 2: Using Traces

- Replaying Traces
- The Database Engine Tuning Advisor
- Demonstration: Using the Database Engine Tuning Advisor
- Combining Traces with Performance Monitor Logs
- Demonstration: Correlating a Trace with Performance Data
- Troubleshooting Concurrency Issues
- Demonstration: Troubleshooting Deadlocks

The Database Engine Tuning Advisor

- Processes workloads captured by SQL Server Profiler
- Suggests index, statistics, and partition changes to improve performance

Demonstration: Using the Database Engine Tuning Advisor

In this demonstration, you will see how to:

- Configure a tuning session
- Generate recommendations
- Validate recommendations

Troubleshooting Concurrency Issues

- Blocking
  - Common occurrence as SQL Server uses locks to ensure data consistency
  - Excessive blocking can affect application performance
  - Monitor locks using the **TSQL_Locks** trace template

- Deadlocks
  - Automatically detected by SQL Server:
    - Rolls back one transaction
    - Returns error message 1205
  - Capture Deadlock Graph trace event
    - View details of statements that resulted in deadlock

Demonstration: Troubleshooting Deadlocks

In this demonstration, you will see how to:

- Capture a trace based on the TSQL_Locks template