Troubleshooting SQL Server 2005/2008 Query Performance/Scalability Issues

This flowchart is meant to address at a high level the most common issues that can affect performance and scalability in SQL Server 2005/2008 implementations.   This is an iterative process and should be performed only in a test environment.

Work left to right, top to bottom	1. READ	2. CAPTURE	3. EVALUATE	4. RESOLVE

OPTIMAL EXECUTION PLANS

Search: www.microsoft.com/TechNet for:
Troubleshooting Performance Problems in SQL Server 2005

Search: support.microsoft.com for ID 224587
Troubleshooting Application Performance

Search: support.microsoft.com for ID 243589
Troubleshooting Slow-running Queries

Search: SQL Server Books On Line
Database Tuning Advisor

Tool: SQL Management Studio
Statistics
Use the SSMS Database Options feature to determine if Auto Create and Update Statistics are enabled

Tool: SQL Profiler
Slow-Running Queries
Use the SQL Profiler TSQL_Duration Template to determine the long-running queries with the longest duration and consuming the most IO

Tool: SQL Profiler
High CPU Usage
Use the SQL Server Standard template to determine queries consuming the most CPU

Auto Create and Update Statistics Enabled?
Have Auto Create and Update Statistics been disabled, indicating that the statistics might be out of date?
— YES → **Update Statistics** — Run the sp_updatestats procedure to update all the statistics in the database
— NO →

Evaluate in SSMS — Copy each of the slow-running queries into a query window and evaluate the execution plan

Table/Index Scans? — Are there tables or index scans that could consume large amounts of IO?
— YES → **Perform Index and Query Analysis** — Remove any join or index hints that may no longer be valid / Evaluate the validity of the search arguments / Ensure there are no columns inside expressions or functions in a predicate that would disallow an index seek / Use the INCLUDE clause in an index to minimize key lookups / Evaluate the query in the Database Tuning Advisor
— NO →

Query Too Complex? — Are there too many joins, temp tables, sub-queries and/or control of low language to allow for optimal execution plans?
— YES → **Rewrite Query** — Try forcing a different plan with a join or index hint / Try breaking the query into smaller parts
— NO →

Faster in SSMS? — Does the query run faster in the SSMS query window than in from the application?
— YES → **Examine App for Result Handling** — Ensure that the application is returning all of the results as soon as possible to the client → **Modify Application Code** — Ensure that only the results that are needed are being requested, and that all results are consumed as soon as possible
— NO →

Cannot Get an Optimal Plan? — Statistics are up-to-date, all of the above steps have been followed, and it seems that the optimizer has not selected an optimal plan?
— YES → **More In-Depth Analysis** — Search on support.mcirosoft.com and msdn.microsoft.com for more in-depth query tuning resources

Evaluate in SSMS — Copy each of the problematic queries into a SSMS query window and evaluate the query

Hash Joins, Sorts, and/or Filters? — Are there hash joins, sorts, and/or filters displayed in the execution plan?
— YES → **Perform Index and Query Analysis** — Determine indexes that will best support this query, using the SSMS query plan or the Database Tuning Advisor.

BLOCKING

Search: support.microsoft.com for ID 271509
How to Monitor SQL Server Blocking

Search: support.microsoft.com for ID 169960
Analyzing and Avoiding Deadlocks in SQL Server

Search: support.microsoft.com for ID 832524
How to Resolve a Deadlock

Tool: SQL Server Profiler
Blocking
Use the sp_blocker_pss80 procedure outlined in article 271509 to determine if there is blocking

Tool: SQL Server Profiler
Deadlock Tracing
Use the trace flags outline in article 832524 or use the Profiler Deadlock Events in a Profiler trace to log deadlock information

Blocking? — Is the blocking script showing that there are blocks occuring?
— YES → **Open Transactions?** — Are there currently open transactions (can be captured using DBCC OpenTran())
— YES → **Examine Transaction Handling Logic** — Ensure that the transactions are all committed and rolled back correctly within the application code, or within the stored procedures

Deadlocks? — Are there deadlocks occuring in the system?
— YES → **Evaluate SQL Server Deadlock Trace Information** — The trace and/or error logs will identify which spids are involved in the deadlocks and the command will be in the input buffer → **Examine Transaction Handling Logic** — Ensure that resources are being acquired in the same order and held for the shortest possible time / Ensure that user input is not being requested in the middle of a transaction

EXECUTION PLAN REUSE

Search: www.microsoft.com/technet for:
Batch Compilation, Recompilation, and Plan Caching

Tool: SQL Server Profiler
Recompilations
Use the Profiler trace outlined in the article to identify recompilation issues

Ad hoc or Dynamic Queries? — Are the queries built dynamically or on an ad hoc basis?
— YES → **Rewrite Queries** — Try using prepared statements, stored procedures, or sp_executesql to reduce compilations
— NO →

Recompliations? — Are there excessive recompilations?
— YES → **Rewrite Queries** — Try using prepared statements, stored procedures, or sp_executesql to reduce compilations / Try changing the database PARAMETERIZATION property between forced and simple

ADVANCED TOOLS

Search: www.microsoft.com/downloads for:
SQL Server 2005 Performance Dashboard

Search: SQL Server Books On Line for:
SQL Server 2008 Management Data Warehouse

Search: www.codeplex.com for:
DMVStats 1.01

Search: www.microsoft.com/downloads for:
RML Utilities for SQL Server