**Checklist for Troubleshooting Slow-Running Queries**

When a query or update takes longer than expected, ask yourself the following questions, which address the reasons for slow-running queries that are listed in the previous section:

1. Is the performance problem related to a component other than queries? For example, is the problem slow network performance? Are there any other components that might be causing or contributing to performance degradation?  
   The Windows System Monitor can be used to monitor the performance of SQL Server and non-SQL Server related components. For more information, see [Monitoring Resource Usage (System Monitor)](http://msdn.microsoft.com/en-us/library/ms191246(v=SQL.90).aspx).
2. If the performance issue is related to queries, which query or set of queries is involved?  
   First use SQL Server Profiler to help identify the slow query or queries. For more information, see [Using SQL Server Profiler](http://msdn.microsoft.com/en-us/library/ms187929(v=SQL.90).aspx).  
   After you have identified the slow-running query or queries, you can further analyze query performance by producing a Showplan, which can be a text, XML, or graphical representation of the query execution plan that the query optimizer generates. You can produce a Showplan using Transact-SQL SET options, SQL Server Management Studio, or SQL Server Profiler.  
   For information about using Transact-SQL SET options to display text and XML execution plans, see [Displaying Execution Plans by Using the Showplan SET Options (Transact-SQL)](http://msdn.microsoft.com/en-us/library/ms180765(v=SQL.90).aspx).  
   For information about using SQL Server Management Studio to display graphical execution plans, see [Displaying Graphical Execution Plans (SQL Server Management Studio)](http://msdn.microsoft.com/en-us/library/ms178071(v=SQL.90).aspx).  
   For information about using SQL Server Profiler to display text and XML execution plans, see [Displaying Execution Plans by Using SQL Server Profiler Event Classes](http://msdn.microsoft.com/en-us/library/ms190233(v=SQL.90).aspx).  
   The information gathered by these tools allows you to determine how a query is executed by the SQL Server query optimizer and which indexes are being used. Using this information, you can determine if performance improvements can be made by rewriting the query, changing the indexes on the tables, or perhaps modifying the database design. For more information, see [Analyzing a Query](http://msdn.microsoft.com/en-us/library/ms191227(v=SQL.90).aspx).
3. Was the query optimized with useful statistics?   
   Statistics on the distribution of values in a column are automatically created on indexed columns by SQL Server. They can also be created on nonindexed columns either manually, using SQL Server Management Studio or the CREATE STATISTICS statement, or automatically, if the **AUTO\_CREATE\_STATISTICS** database option is set to **TRUE**. These statistics can be used by the query processor to determine the optimal strategy for evaluating a query. Maintaining additional statistics on nonindexed columns involved in join operations can improve query performance. For more information, see [Index Statistics](http://msdn.microsoft.com/en-us/library/ms190397(v=SQL.90).aspx).  
   Monitor the query using SQL Server Profiler or the graphical execution plan in SQL Server Management Studio to determine if the query has enough statistics. For more information, see[Errors and Warnings Event Category (Database Engine)](http://msdn.microsoft.com/en-us/library/ms177448(v=SQL.90).aspx).
4. Are the query statistics up to date? Are the statistics automatically updated?   
   SQL Server automatically creates and updates query statistics on indexed columns (as long as automatic query statistic updating is not disabled). Additionally, statistics can be updated on nonindexed columns either manually, using SQL Server Management Studio or the UPDATE STATISTICS statement, or automatically, if the **AUTO\_UPDATE\_STATISTICS** database option is set to **TRUE**. Up-to-date statistics are not dependent upon date or time data. If no UPDATE operations have taken place, the query statistics are still up-to-date.  
   If statistics are not set to update automatically, set them to do so. For more information, see [Index Statistics](http://msdn.microsoft.com/en-us/library/ms190397(v=SQL.90).aspx).
5. Are suitable indexes available? Would adding one or more indexes improve query performance? For more information, see [General Index Design Guidelines](http://msdn.microsoft.com/en-us/library/ms191195(v=SQL.90).aspx), [Finding Missing Indexes](http://msdn.microsoft.com/en-us/library/ms345417(v=SQL.90).aspx), and [Database Engine Tuning Advisor Reference](http://msdn.microsoft.com/en-us/library/ms173494(v=SQL.90).aspx). Database Engine Tuning Advisor can also recommend the creation of necessary statistics.
6. Are there any data or index hot spots? Consider using disk striping. Disk striping can be implemented by using RAID (redundant array of independent disks) level 0, where data is distributed across multiple disk drives. For more information, see [Using Files and Filegroups](http://msdn.microsoft.com/en-us/library/ms187087(v=SQL.90).aspx) and [RAID](http://msdn.microsoft.com/en-us/library/ms184252(v=SQL.90).aspx).
7. Is the query optimizer provided with the best opportunity to optimize a complex query? For more information, see [Query Tuning Recommendations](http://msdn.microsoft.com/en-us/library/ms188722(v=SQL.90).aspx).
8. If you have a large volume of data, do you need to partition it? Data manageability is the main benefit of partitioning, but if your tables and indexes on them are partitioned similarly, partitioning can also improve query performance. For more information, see [Understanding Partitioning](http://msdn.microsoft.com/en-us/library/ms188232(v=SQL.90).aspx) and [Tuning the Physical Database Design](http://msdn.microsoft.com/en-us/library/ms191531(v=SQL.90).aspx).