



# SQL Server™ 2005: SQL Query Tuning

---

# Table of Contents

---

SQL Server 2005: SQL Query Tuning .....	3
Lab Setup.....	4
Exercise 1 Troubleshooting Deadlocks with SQL Profiler.....	5
Exercise 2 Isolating a Slow Running Query Using SQL Profiler.....	9
Exercise 3 Examining an Execution Plan .....	11
Exercise 4 Using the Database Tuning Advisor .....	12

# SQL Server 2005: SQL Query Tuning

---

## Objectives

**NOTE:** This lab focuses on the concepts in this module and as a result may not comply with Microsoft® security recommendations.

**NOTE:** The SQL Server 2005 labs are based on beta builds of the product. The intent of these labs is to provide you with a general feel of some of the planned features for the next release of SQL Server. As with all software development projects, the final version may differ from beta builds in both features and user interface. For the latest details on SQL Server 2005, please visit <http://www.microsoft.com/sql/2005/>.

After completing this lab, you will be able to:

- Use the SQL Profiler to troubleshoot deadlocks
- Generate a query plan for a poorly performing query and save the plan as an XML document
- Use the Database Tuning Advisor

## Scenario

You are the database administrator for the **AdventureWorks** database.

Your users are experiencing frequent deadlocks and you are concerned that the deadlocks might be a cause of poor system performance. You have isolated a query that is often involved in deadlocks. You will capture a trace of the events leading up to the deadlock and the details of the deadlock itself, using SQL Profiler.

After tracking down the cause of the deadlocks, you realize that it is not a major cause of system performance degradation, so you decide to investigate key queries. You will analyze the queries by inspecting the query plans being used for them, and you will employ the Index Tuning Advisor to suggest optimum indexes.

## Prerequisites

- Experience with basic administration tasks in SQL Server 2000
- A familiarity with the Transact-SQL language
- Completed the SQL Server Management Studio hands-on lab

## Estimated Time to Complete This Lab

45 Minutes

## Lab Setup

Tasks	Detailed Steps
1. Log in.	1. Log in using the <b>Administrator</b> user account. The password is <b>Pass@word1</b> .

# Exercise 1

## Troubleshooting Deadlocks with SQL Profiler

### Scenario

In this exercise, you will use the SQL Profiler tool to capture deadlock information. Deadlocks occur when two processes are mutually blocking each other because each is holding an exclusive lock that prevents the other from completing its transaction.

Deadlocks can also occur when two processes are trying to access rows in the same table if the rows are accessed in a different sequence by the two processes. Deadlocks can involve more than two processes. For example, process A could block process B, which is blocking process C, which is blocking process A.

SQL Server detects when a deadlock occurs by periodically looking for “cycles” in the chain of blocked and blocker processes. When a deadlock is detected, SQL Server will choose one of the processes as the “victim,” abort that session and rollback the process’s current transaction. This will release all the locks that process was holding and usually allow the any blocked processes to continue execution.

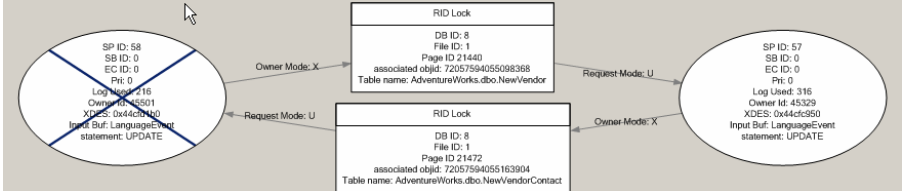
The likelihood of deadlocks can be reduced by careful application design and good indexing. The best prevention is to keep your transactions as short as possible, and never to allow transactions to span batches. In addition, wherever possible you should try to access tables in the same order. For example, if you have several different procedures that all need to update the same three tables, they all should update the tables in the same order. Even with careful planning, deadlocks cannot be prevented completely. When they do occur, you can use the SQL Profiler to capture the events leading up to the deadlock, and to determine the actual objects and TSQL statements involved.

SQL Server 2005 will also allow you to save the deadlock graph in XML. Eventually, there will be tools in SQL Profiler to read in a Deadlock XML file and display the deadlock graphically, clearly indicating which resources were requested by each process involved, which locks were granted, and where blocking occurred prior to the deadlock detection.

Tasks	Detailed Steps
<p>1. Define a trace to capture deadlock information.</p> <p><b>NOTE:</b> Not all data items are available for every event. This is also true in SQL Server 2000 Profiler, but isn't obvious from the way data items are selected in that version. For example, performance data items, such as duration, reads, and writes, are not</p>	<ol style="list-style-type: none"> <li>1. From the Windows® task bar, select the <b>Start   All Programs   Microsoft SQL Server 2005   Performance Tools   SQL Server Profiler</b> menu item.</li> <li>2. Click the <b>New Trace</b> button (the first button on the SQL Profiler toolbar).</li> <li>3. When the <b>Connect to Server</b> dialog box appears, verify that <b>Server type</b> is set to <b>SQL Server</b>, <b>Server name</b> is set to <b>localhost</b>, and that <b>Windows Authentication</b> is selected as the authentication method. Click <b>Connect</b>.</li> <li>4. If a SQL Profiler message box appears to retrieve a trace definition file from the server, click <b>OK</b>.</li> </ol> <p>The <b>Trace Properties</b> dialog box opens. The first tab, <b>General</b>, is similar to the <b>General</b> tab in the SQL Server 2000 Profiler's trace definition dialog box.</p> <ol style="list-style-type: none"> <li>5. In the <b>Trace Name</b> text box, enter <b>Deadlock1</b>. In the <b>Use the template</b> drop-down, select <b>Blank</b>.</li> <li>6. Select the <b>Save to File</b> checkbox and change the directory to <b>C:\SQL Labs\User Projects</b>. Accept the default filename of <b>Deadlock1.trc</b>. Click <b>Save</b>. Leave the other options on the <b>Trace Properties</b> dialog box at the default values.</li> <li>7. Click the <b>Events Selection</b> tab.</li> </ol> <p>In SQL Server 2005, the process of defining the events to capture in the Profiler is different from what it was in SQL Server 2000. Events are selected by expanding an event category and then checking the boxes for the events you want to capture.</p> <ol style="list-style-type: none"> <li>8. Select the checkboxes for the following events, and ensure that all of their available data values are selected:             <ul style="list-style-type: none"> <li>▪ In the <b>Locks</b> category:</li> </ul> </li> </ol>

Tasks	Detailed Steps
<p>available for starting events. These performance data items are only available for events such as SP:Completed (stored procedure completion event) and SP:StmtCompleted (statement completion within a stored procedure). In addition, some events have data for EventSubClass, BinaryData and IntegerData, and others do not.</p> <p><b>NOTE:</b> You do not need to apply any additional filters for this exercise, but you may want to explore which data items allow filters and which do not. Some data items allow a range filter. You can use this filter to specify exact values that must match, or a range of values specified by greater than or less than operators. Other data items allow a pattern-matching filter, which you can use with "like" or "not like" operators. If you right-click on any data item or event name, the shortcut menu includes the option, <b>Organize Columns</b>. This option allows you to rearrange the order in which the data items appear in your output.</p>	<ul style="list-style-type: none"> <li>○ Lock: Deadlock Chain</li> <li>○ Deadlock graph</li> <li>▪ In the <b>Stored Procedures</b> category: <ul style="list-style-type: none"> <li>○ RPC:Starting</li> <li>○ SP:StmtStarting</li> <li>○ SP:StmtCompleted</li> <li>○ RPC:Completed</li> </ul> </li> </ul> <p>In SQL Profiler for 2005, filters are set in the <b>Events Selection</b> screen.</p> <ol style="list-style-type: none"> <li>9. Scroll to the right until you find the <b>DatabaseID</b> column. You may need to increase the width of the columns to see the full names.</li> <li>10. Right-click the DatabaseID column heading and select <b>Edit Column Filter</b>. Expand the <b>Not equal to</b> node, and type in the number 4. Click <b>OK</b>. This will filter out all activity in the msdb database.</li> <li>11. Right-click the DatabaseName column heading and select <b>Edit Column Filter</b>. Expand the <b>Not like</b> node, and type in ReportServer. Click <b>OK</b>. This will filter out all activity in the ReportServer database.</li> <li>12. Click the <b>Events Extraction Settings</b> tab. Select the <b>Save Deadlock XML Events Separately</b> checkbox. Verify that the current directory is <b>C:\SQL Labs\User Projects</b>, enter the <b>File name</b> "XMLdeadlock", and click <b>Save</b>. Click <b>Each Deadlock XML batch in a distinct file</b>, and click <b>Run</b>. The trace begins.</li> </ol>
<ol style="list-style-type: none"> <li>2. Generate a multi-table deadlock.</li> </ol>	<ol style="list-style-type: none"> <li>1. From the Windows task bar, select <b>Start   All Programs   Microsoft SQL Server 2005   SQL Server Management Studio</b>.</li> <li>2. When the <b>Connect to Server</b> dialog box appears, enter <b>localhost</b> as the <b>Server name</b>, and verify that <b>Windows Authentication</b> is selected as the authentication method. Click <b>Connect</b>.</li> <li>3. Select the <b>File   Open   Project/Solution</b> menu item. Navigate to <b>C:\SQL Labs\Lab Projects\Tuning Lab\Exercise 1</b>. Select <b>Exercise 1.ssmssln</b> and click <b>Open</b>.</li> <li>4. In the <b>Solution Explorer</b> window, double-click <b>Copy tables.sql</b> under the <b>Queries</b> folder, to open this SQL script.</li> <li>5. Press <b>F5</b> or click the <b>Execute</b> toolbar button.</li> </ol> <p>This script, which can take a few minutes depending upon the speed of your machine,</p>

Tasks	Detailed Steps
<p><b>Note:</b> You have executed a BEGIN TRAN and no COMMIT TRAN or ROLLBACK TRAN. You would normally want to avoid this, because transactions held open across batches have a greater chance of being involved in a deadlock. This exercise violates that recommendation intentionally to cause a deadlock.</p> <p><b>NOTE:</b> The transaction that</p>	<p>to run, makes copies of several tables from the AdventureWorks database. When you modify data in the following tasks, you will be working with the new copies and will not be affecting the original data. After inspecting the statements, you can close this script.</p> <p><b>6.</b> Open the SQL script <b>First part multi table deadlock.sql</b>.</p> <p>This script allows you to create a deadlock. The scenario is that you need to change the Active status for two of your vendors to “false”. You also need to update the contact data for these vendors, to identify two contacts as owners.</p> <p>The script in <b>First part multi table deadlock.sql</b> updates the NewVendor data for Vendor ID 1, and then attempts to update the NewVendorContact data for Vendor ID 1. However, you will not execute the entire script at once.</p> <p><b>7.</b> Select a portion of the script, beginning at the first line and ending before the comment line that reads <b>Only execute up to this point in the first batch</b>. Press <b>F5</b> or click the <b>Execute</b> button.</p> <p><b>8.</b> Open the SQL script <b>Second part multi table deadlock.sql</b>.</p> <p>The script in <b>Second part multi table deadlock.sql</b> updates the NewVendorContact data for Vendor ID 15, and then attempts to update the NewVendor data for Vendor ID 15. Again, you will not execute the entire script all at once.</p> <p><b>9.</b> Select the portion of the script from the first line down to before the comment line that reads <b>Only execute up to this point in the first batch</b>. Press <b>F5</b> or click the <b>Execute</b> button.</p> <p><b>10.</b> Use the down-arrows in the upper right corner of the code window to select <b>First part multi table deadlock.sql</b>. Select only the second UPDATE statement in the transaction, after the comment <b>Next, start execution here</b>, and click the <b>Execute</b> button.</p> <p>The update statement will not complete executing, because it will be blocked by the first update in the transaction in the <b>Second part multi table deadlock.sql</b> script. This is not a deadlock yet, because there is only one blocking process and one blocked process.</p> <p><b>11.</b> Go back to the query window for <b>Second part multi table deadlock.sql</b>. Select only the second UPDATE statement in the transaction, beginning after the comment <b>Next, start execution here</b>, and click the <b>Execute</b> button.</p> <p>This second update will be blocked by the first update in the transaction in the <b>First part multi table deadlock.sql</b> script. Because both processes are now blocking each other, this is a true deadlock. SQL Server will detect the deadlock and choose one process as the “victim.” You should see this message in the results window for one of the deadlock scripts (the Process ID may be different on your computer):</p> <div data-bbox="548 1493 1429 1623" style="background-color: #f0f0f0; padding: 5px;"> <pre>Msg 1205, Level 13, State 45, Line 1 Transaction (Process ID 57) was deadlocked on lock resources with another process and has been chosen as the deadlock victim. Rerun the transaction.</pre> </div> <p>The transaction is rolled back when the deadlock error message is generated, and all the locks are released. The other “winning” transaction is then no longer blocked and can complete its second update.</p> <p><b>12.</b> Run the last line (ROLLBACK TRAN) in the transaction that wasn’t the deadlock victim, to rollback the transaction.</p>

Tasks	Detailed Steps
<p>was not the deadlock victim is still pending, although it has completed both its updates. It has been neither committed nor rolled back.</p>	
<p><b>3.</b> Examine the captured deadlock information.</p> <p><b>TIP:</b> Move the cursor over the ovals in the actual Deadlock Graph to see the query text.</p>	<ol style="list-style-type: none"> <li><b>1.</b> Return to your running trace in the SQL Profiler and click the <b>Stop Selected Trace</b> toolbar button, which has a red square icon. Near the end of the trace you will see two rows that begin with <b>Lock:Deadlock Chain</b>.</li> <li><b>2.</b> Select the <b>Deadlock Graph</b> event below the Lock:Deadlock Chain events. When you select this event in Profiler, Deadlock Graph information appears in the lower window of the Profiler, similar to that shown in Figure 1.</li> </ol>  <p><b>Figure 1: Sample Deadlock Graph</b></p> <p>This visual representation of the processes involved in deadlock includes the type and mode of lock each process held, the type of lock each process was waiting on, and some information about the actual resource on which the lock was placed.</p>



## Exercise 2

# Isolating a Slow Running Query Using SQL Profiler

### Scenario

In this exercise, you will use the SQL Profiler to isolate a poorly performing query within a query batch.

Tasks	Detailed Steps
<p>1. Define a trace to capture query performance information.</p>	<ol style="list-style-type: none"> <li>1. If SQL Profiler is not already loaded, then from the Windows task bar, select the <b>Start   All Programs   Microsoft SQL Server 2005   Profiler</b> menu.</li> <li>2. Click the <b>New Trace</b> button (the first button on the SQL Profiler toolbar).</li> <li>3. When the <b>Connect to Server</b> dialog box opens, verify that <b>Server type</b> is set to <b>SQL Server</b>, <b>Server name</b> is set to <b>localhost</b>, and that <b>Windows Authentication</b> is selected as the authentication method. Click <b>Connect</b>.</li> <li>4. In the Trace Name text box, enter <b>TuneBatch</b> as the name of your trace. In the <b>Use the template</b> drop-down, select <b>TSQL</b>. Check the <b>Save to File</b> box and confirm that the directory is <b>C:\SQL Labs\User Projects</b>. Accept the default file name of <b>TuneBatch.trc</b>. Click <b>Save</b>, and leave the other options on the <b>General</b> tab of the <b>Trace Properties</b> dialog box at the default values.</li> <li>5. Click the <b>Events Selection</b> tab. In addition to the events that are selected by default in this template, select the following events, and include all of their available data values. You may need to select <b>Show All Events</b> to see these events. <ul style="list-style-type: none"> <li>▪ In the <b>Performance</b> category: <ul style="list-style-type: none"> <li>○ Showplan XML</li> <li>○ Showplan All</li> </ul> </li> <li>▪ In the <b>Stored Procedures</b> category: <ul style="list-style-type: none"> <li>○ SP:StmtCompleted</li> <li>○ In the <b>TSQL</b> category: <ul style="list-style-type: none"> <li>○ SQL:BatchCompleted</li> </ul> </li> </ul> </li> </ul> </li> <li>6. Right-click the <b>DatabaseName</b> column heading and select <b>Edit Column Filter</b>. You may need to select <b>Show All Columns</b> to see these columns. Expand the option <b>Like</b> and type in "AdventureWorks". Click <b>OK</b>. This filters out all activity except for the selected events in the AdventureWorks database.</li> <li>7. Click the <b>Events Extraction Settings</b> tab. Select the <b>Save XML Showplan events separately</b> checkbox. Verify that the target directory is <b>C:\SQL Labs\User Projects</b>, enter the file name <b>XMLQueryPlans</b>, and click <b>Save</b>. Click <b>Each XML Showplan batch in a distinct file</b>, and click <b>Run</b>. The trace begins.</li> </ol>
<p>2. Isolate a poorly performing query in a batch.</p>	<ol style="list-style-type: none"> <li>1. Load <b>Microsoft SQL Server Management Studio</b>, if you do not already have it loaded.</li> <li>2. Click <b>File   Open   Project/Solution</b>. Navigate to <b>C:\SQL Labs\Lab Projects\Tuning Lab\Exercise 2</b>. Select <b>Exercise 2.ssmssl</b> and click <b>Open</b>.</li> <li>3. In the <b>Solution Explorer</b> window, double-click <b>BatchToTune.sql</b> under the <b>Queries</b> folder to open this SQL script.</li> <li>4. Press <b>F5</b> or click the <b>Execute</b> toolbar button.</li> </ol>

---

Tasks	Detailed Steps
	<ol style="list-style-type: none"><li data-bbox="509 239 1438 331">5. Once the script execution completes, go back to the trace running in <b>SQL Profiler</b> and stop the trace by clicking the <b>Stop Selected Trace</b> button, the red square in the toolbar.</li><li data-bbox="509 338 1438 394">6. Scroll to the right to find the <b>Duration</b> column. Only some of the events have a duration listed.</li><li data-bbox="509 401 1438 464">7. Isolate the event with the greatest value in the Duration column. Select that row in the Profiler output, and you should see the query in the lower Profiler window.</li><li data-bbox="509 470 1438 562">8. Look in your <b>C:\SQL Labs\User Projects\</b> directory. It should contain several <b>.SQLPlan</b> files. Select the most recent file and open it in Management Studio by right-clicking on the file name, choosing <b>Open</b>.</li></ol> <p data-bbox="509 569 1438 693">Having query plans in XML allows third party vendors to develop software to interpret and display them. In addition, an XML query plan can be sent to a support provider, who can then view it in a graphical format and see the same output that appears in SQL Server Management Studio.</p>

# Exercise 3

## Examining an Execution Plan

### Scenario

In this exercise, you will use the **Include Actual Execution Plan** option to examine the execution plan for a poorly performing query.

Tasks	Detailed Steps
<p>1. Examine the statistics and Execution Plan for a poorly performing query.</p>	<ol style="list-style-type: none"> <li>1. Load <b>Microsoft SQL Server Management Studio</b>, if you do not already have it loaded.</li> <li>2. Click <b>File   Open   Project/Solution</b>. Navigate to <b>C:\SQL Labs\Lab Projects\Tuning Lab\Exercise 3</b>. Select <b>Exercise 3.ssmssl</b> and click <b>Open</b>.</li> <li>3. In the <b>Solution Explorer</b> window, double-click <b>Slow query.sql</b> under the <b>Queries</b> folder, to open this SQL script.</li> <li>4. Right-click anywhere in the query window, and click <b>Query Options</b>.</li> <li>5. In the left-hand pane, select <b>Execution   Advanced</b> and check the box for <b>SET STATISTICS TIME</b>. Click <b>OK</b>.</li> <li>6. Right-click anywhere in the query window, and click <b>Include Actual Execution Plan</b> (you can also select this option from the <b>Query</b> menu in the toolbar).</li> <li>7. Switch to the <b>Profiler</b> and run the <b>TuneBatch</b> trace by clicking the <b>Start Selected Trace</b> button, the green arrowhead. If you receive a message “Trace saving data options have been reset. Do you want to proceed without saving the output?”, click <b>Yes</b>.</li> <li>8. Switch back to Management Studio and execute the query.</li> <li>9. Click the <b>Execution plan</b> tab at the top of the results window.</li> </ol> <p>Notice that almost all the join operations are performed using Hash Match joins. SQL Server uses a hash join when there are no useful indexes on the query. This generally indicates that tuning is in order. All of the table accesses are performed as table scans, because there are no indexes on these tables at all.</p> <ol style="list-style-type: none"> <li>10. Switch to the <b>Profiler</b> and select the row for the <b>SQL:BatchCompleted</b> event for the <b>SELECT</b> query that you just executed. Note the <b>Duration</b> value, which you will refer to in the next exercise.</li> </ol>

## Exercise 4

# Using the Database Tuning Advisor

### Scenario

In this exercise, you will let the Database Tuning Advisor (DTA) recommend indexes for the poorly performing query that you examined in Exercise 3. The Database Tuning Advisor is the SQL Server 2005 successor to the Index Tuning Wizard. For this exercise, you will only be tuning a single query, but the DTA is capable of tuning an entire workload of events captured in a SQL Profiler trace. The workload could contain many queries spanning several hours during real-world conditions.

Tasks	Detailed Steps
<p>1. Use the Database Tuning Advisor to recommend indexes for a single query.</p>	<ol style="list-style-type: none"> <li>1. Return to SQL Profiler and stop any running traces by clicking on the red <b>Stop Selected Trace</b> button.</li> <li>2. Select the <b>Tools   Database Tuning Advisor</b> menu item, and confirm the connection settings if a dialog box pops up. (Be patient. The dialog box may take a few moments to load.) Supply connection information and click <b>Connect</b> when prompted.</li> <li>3. On the <b>Workload</b> tab, type in a Session name of <b>TuneQuery</b>.</li> <li>4. For the Workload, select <b>File</b>. Click the <b>Browse for a workload file</b> button (the button with a binoculars icon, to the right of the Workload textbox). In the <b>Select Workload File</b> dialog box, change <b>Files of type</b> to <b>SQL Script (*.sql)</b>.</li> <li>5. Browse to <b>C:\SQL Labs\Lab Projects\Tuning Lab\Exercise3</b> and select the file <b>Slow Query.sql</b>. Click <b>Open</b>.</li> <li>6. In the <b>Databases and Tables</b> section in the lower part of the <b>Workload</b> tab, check the <b>AdventureWorks</b> database.</li> <li>7. Examine the <b>Tuning Options</b> tab, including the <b>Advanced Options</b>. Do not change anything in the <b>Advanced Options</b> dialog box.</li> <li>8. Click the <b>Start Analysis</b> button (the green arrow) to start the analysis. Wait until the analysis completes, which may take a minute or two.</li> <li>9. When the analysis is done, you'll see three more tabs: <b>Progress</b>, <b>Recommendations</b>, and <b>Reports</b>. Click the <b>Recommendations</b> tab and clear the box <b>Show existing objects</b> at the bottom of the tab to show only the changes that are recommended.</li> <li>10. Select the <b>Actions   Apply Recommendations</b> menu item. Click <b>OK</b> in the <b>Apply Recommendations</b> dialog box, after verifying that <b>Apply now</b> is selected.</li> <li>11. Close the Database Tuning Advisor.</li> <li>12. In the <b>SQL Profiler</b>, restart the <b>TuneBatch</b> trace.</li> <li>13. Return to the <b>Slow Query.sql</b> window in the <b>SQL Server Management Studio</b>, and execute the query.</li> <li>14. In the SQL Profiler, check the Duration of the SELECT statement. It should be lower than the value you received in the last exercise, before you let the DTA add the recommended indexes.</li> </ol>