

Deadlock Detection, Troubleshooting, and Prevention

References and Resources

Trevor Barkhouse

Introduction to Deadlocks

- A simple and humorous explanation from Michael J. Swart: “[Deadlocks explained ... in comic form!](#)” (used with permission)
- Wikipedia article on the general, computer science concept of deadlocks: “[Deadlock](#)”
- SQL Server Books Online article on deadlocks: “[Deadlocking](#)”

Deadlock Detection

- The deadlock victim receives a 1205 error from SQL Server: “Transaction (Process ID %d) was deadlocked on %.*ls resources with another process and has been chosen as the deadlock victim. Rerun the transaction.”
 - More information on the error can be found at “[MSSQLSERVER 1205](#)”
- Deadlock (1205) errors can be caught using exception handling and the operation can be attempted additional times
 - Since deadlocks are concurrency problems, they will often succeed on a subsequent attempt
 - Encountering deadlocks on multiple successive attempts of the same operation indicate a much more serious problem
 - The load on the system may exceed its capacity
 - There may be flaws in the design or implementation of the application that contribute to concurrency problems
 - The isolation level may be too restrictive (see “[Isolation Levels in the Database Engine](#)”)
 - The error can be caught in the [Data Access Layer \(DAL\)](#)
 - This type of functionality, “Fault retry” logic, should always be in place to handle the following types of transient issues (between a SQL Server client and server):
 - Deadlocks
 - Failover of a cluster resource group on a [Failover Cluster](#)
 - Failover (changing of the principal server) when using [Database Mirroring](#)
 - Momentary network interruption
 - The following blog links to a fault retry provider and sample on CodePlex: “[Fault Retry provider and sample](#)”
 - As of SQL Server 2005, 1205 errors can be caught within T-SQL code using the new [TRY...CATCH](#) construct
 - An example can be found in *Inside Microsoft SQL Server 2005: Query Tuning and Optimization* by Kalen Delaney, et al. (Microsoft Press: 2008)
 - Chapter 6, “Concurrency Problems,” pages 365 and 366
 - The example is also included in the presentation’s demonstrations: “Demo 09_Sample fault tolerance code in T-SQL.sql”

- Deadlock (1205) errors are not recorded anywhere (by default)
 - The application layer should catch them and log them somewhere
 - The error message can be changed, within SQL Server, so that they will be written to the [SQL Server error log](#)
 - This can be accomplished via the [\[sp_altermessage\]](#) system stored procedure
 - Doing so worked on SQL Server 2000 (and earlier) without any problems
 - See the “Configure error 1205 for logging,2000.sql” script for an example
 - The functionality (altering system messages with ID values below a certain number) was removed from SQL Server 2005 and 2008
 - In response to [numerous complaints from users](#), Microsoft finally [restored the functionality](#) in [Service Pack 3 for SQL Server 2005](#) (build 9.00.4035) and [Service Pack 1 for SQL Server 2008](#) (build 10.00.2531)
 - See the “Demo 04_Configure error 1205 for logging,2005+.sql” script for an example
 - Before the message modification functionality was put back into SQL Server, Patrick LeBlanc wrote an article on a work-around: “[Deadlock Notifications in SQL Server 2005](#)”
- Deadlocks can also be detected via the “Number of Deadlocks/sec” performance counter of the “[SQLServer:Locks](#)” performance object
 - Keep in mind that the performance object may have a different name if it is associated with a [named SQL Server instance](#)
- Starting with SQL Server 2005, deadlocks can also be detected via a WMI event
 - The following Books Online article has a sample: “[Sample: Creating a SQL Server Agent Alert by Using the WMI Provider for Server Events](#)”
 - Unfortunately, I have found this functionality to be too brittle to use (I have personally run into each of the following problems):
 - “[Dead Alert with WMI and deadlock_graph](#)”
 - “[FIX: Error message when you create a WMI event alert by using the sp_add_alert stored procedure in SQL Server 2005: ‘The @wmi_query could not be executed in the @wmi_namespace provided’](#)”
 - “[FIX: Error message when you create a WMI event alert by using the sp_add_alert stored procedure in a SQL Server 2005 cluster: ‘The @wmi_query could not be executed in the @wmi_namespace provided’](#)”
 - “[FIX: Error message when you try to create a WMI event alert by using the sp_add_alert stored procedure in SQL Server 2008: ‘The @wmi_query could not be executed in the @wmi_namespace provided’](#)”
 - “[FIX: Error message when you run the sp_add_alert stored procedure in SQL Server 2008 on a computer that is running Windows Server 2008 or Windows Vista: ‘SQLServerAgent Error: WMI error: 0x80041003’](#)”
- The database engine has a special thread, called LOCK_MONITOR, that checks for deadlocks (cycles in lock grants and requests or resource conflicts)
 - The mechanism is covered in *Inside Microsoft SQL Server 2005: The Storage Engine* by Kalen Delaney (Microsoft Press: 2007)
 - Chapter 8, “Locking and Concurrency,” pages 379 through 381
 - The lock monitor thread is also described in “[Detecting and Ending Deadlocks](#)”

Deadlock Troubleshooting

- Overview:
 - [“Deadlock article on SqlServerCentral.com and Deadlock videos on JumpStartTv.com”](#)
 - [“Deadlock Troubleshooting, Part 1”](#)
 - [“Basic SQL Server Deadlock Debugging”](#)
 - [“Resolving Deadlocks in SQL Server 2000”](#)
- Interpreting the output of the trace flags:
 - [“Deadlock Troubleshooting, Part 1”](#)
 - [“Deadlock Troubleshooting, Part 2”](#)
 - [“Deadlock Troubleshooting, Part 3”](#)
 - [“Detecting and Ending Deadlocks”](#)
 - [“SQL Server technical bulletin - How to resolve a deadlock”](#)
- Capturing and using deadlock graphs for troubleshooting deadlocks:
 - [“Analyzing Deadlocks with SQL Server Profiler”](#)
 - [“Basic SQL Server Deadlock Debugging”](#)
 - [“Capturing SQL Server Deadlock Information in XML Format”](#)
 - [“SQL Server Profiler Graphical Deadlock Chain”](#)
 - [“Did you know? -- Deadlock Graph Event Isn't Generated When Filtering on DatabaseID”](#)
- Retrieving information on historical deadlocks from the built-in [“system_health” Extended Events](#) session:
 - [“Retrieving Deadlock Graphs with SQL Server 2008 Extended Events”](#)
 - [“Getting historical deadlock info using extended events”](#)
 - There was a bug with the generated XML data:
 - [“Invalid XML in Extended Events xml_deadlock_report output”](#)
 - [“FIX: Error message when you use the system_health Extended Event session to capture a deadlock graph in SQL Server 2008: ‘Msg 9436: XML parsing: line 54, character 12, end tag does not match start tag’”](#)
 - [“Changes to the Deadlock Monitor for the Extended Events xml_deadlock_report and Multi-Victim Deadlocks”](#)
- Information on specific types of deadlocks:
 - [“The Anatomy of a Deadlock”](#)
 - [“Anatomy of a Deadlock - Part Deux”](#)
 - [“The Curious Case of the Dubious Deadlock and the Not So Logical Lock”](#)
 - [“Today’s Annoyingly-Unwieldy Term: ‘Intra-Query Parallel Thread Deadlocks’”](#)
- Reproducing certain types of deadlocks:
 - [“Reproducing deadlocks involving only one table”](#)
 - [“Some heap tables may be more prone to deadlocks than identical tables with clustered indexes”](#)
 - [“Reproducing one more intermittent deadlock on only one table”](#)

Deadlock Prevention

- Avoiding logic that is prone to deadlocks:
 - [“Minimizing Deadlocks”](#)
 - [“Deadlocked”](#)
 - [“SQL Server 2005 Waiting and Blocking Issues – Deadlocks”](#)
 - [“When acquiring locks in the same order is not possible or not feasible.”](#)
 - [“SQL Server Deadlock Fix: Force join order, or automatically retry?”](#)
 - [“Minimizing Deadlocks”](#)
- Optimizing database queries and routines:
 - [“When Index Covering Prevents Deadlocks”](#)
 - [“Basic SQL Server Deadlock Debugging”](#)
 - [“Deadlock Troubleshooting, Part 1”](#)
 - [“Deadlock Troubleshooting, Part 2”](#)
 - [“Deadlock Troubleshooting, Part 3”](#)
 - [“Does SQL Server’s Database Tuning Advisor modify data?”](#)
- Adjusting isolation levels and using locking hints:
 - [“Minimizing Deadlocks”](#)
 - Problems with insert-or-update functionality (pre-MERGE statement):
 - [“The SELECT/UPDATE problem, or, why UPDLOCK?”](#)
 - [“Conditional INSERT/UPDATE Race Condition”](#)
 - [“Conversion Deadlock”](#)
 - [“Insert or Update pattern for Sql Server”](#)
 - [“Best way to Update row if exists, Insert if not”](#)
 - [“Processing Data Queues in SQL Server with READPAST and UPDLOCK”](#)
 - [“PRB: Deadlock May Be Encountered When Using UPDLOCK Hint”](#)
 - New to SQL Server 2005, the [SNAPSHOT isolation level](#) (see [“Using Row Versioning-based Isolation Levels”](#) for more information)
 - [“Snapshot Isolation and Deadlocking”](#)
 - [“Table Hints \(Transact-SQL\)”](#)
- Using bound connections for [Multiple Active Result Sets \(MARS\)](#):
 - [“Minimizing Deadlocks”](#)
 - [“Using Bound Sessions”](#)

Miscellaneous Resources

- *Inside Microsoft SQL Server 2005: The Storage Engine* by Kalen Delaney (Microsoft Press: 2007)
 - Chapter 8, “Locking and Concurrency,” pages 377 through 381
- *Inside Microsoft SQL Server 2005: Query Tuning and Optimization* by Kalen Delaney, et al. (Microsoft Press: 2008)
 - Chapter 6, “Concurrency Problems,” pages 354 through 368
- [“SQL Server Deadlock Priority Configuration”](#)
- [“SET DEADLOCK_PRIORITY \(Transact-SQL\)”](#)