SQL Server: Deadlock Analysis and Prevention

Module 7: Handling Deadlocks

Jonathan M. Kehayias
Jonathan@SQLskills.com

Introduction

- **Deadlocks do not have to result in application tier errors when they occur in SQL Server**
- **Proper handling of the resulting 1205 error from SQL Server can reduce the end-user impact of deadlocks if design changes cannot be implemented to prevent deadlocks completely**
    - An example of this is deadlock management during index crawls in Microsoft Office SharePoint Server

- **In this module we'll cover:**
    - Handling deadlocks in Transact-SQL
    - Handling deadlocks in ADO.NET

Catching Deadlock Errors

- **TRY/CATCH blocks in Transact-SQL can handle 1205 errors from deadlocks when they occur**
  - The ERRORNUMBER() function will return the error number being raised
- **ADO.NET can handle deadlocks when they occur by catching the SqlException that is raised by the 1205 error returned by SQL Server when a deadlock occurs**
  - The Number property of SqlException will return the error number raised

Retrying After a Deadlock

- **Custom retry logic can be implemented to reattempt the operation that was selected as the deadlock victim**
  - Typically the lock scenario that resulted in the deadlock occurring only lasts a short duration, generally milliseconds, and will not exist when the transaction is resubmitted
  - The retry logic must be coded so that an infinite loop does not occur if the deadlocking persists in the engine
- **Logging of the deadlock can occur to allow for diagnosis and potential prevention in the future**

Summary

- It may not be possible to prevent deadlocks from occurring within the current database or application design
- With proper application design and defensive coding deadlocks will not result in negative end user experiences when deadlocks occur
- Retrying the victim operation of a deadlock will generally result in a successful execution due to different locks being held