SQL Server: Deadlock Analysis and Prevention

Module 3: Deadlock Detection

Jonathan M. Kehayias
Jonathan@SQLskills.com

pluralsight
hardcore developer training

Introduction

- **SQL Server is designed to handle deadlocks automatically**
  - Manual intervention is not required
- **The lock monitor thread is ultimately responsible for deadlock detection and management inside the Database Engine**
  - Deadlock detection also extends into the SQLCLR hosting implementation
- **In most cases deadlock victim selection is predictable**

- **In this module we'll cover:**
  - Deadlock detection
  - Deadlock victim selection
  - Deadlock priority

Deadlock Detection

- **The lock monitor thread is responsible for deadlock detection and initiates periodic searches to identify and resolve deadlocks**
- **The default between deadlock searches is 5 seconds**
- **Each time a deadlock is detected, the search interval decreases to as low as 100ms based on the frequency of the deadlocks occurring in the server**
  - When a deadlock search does not find a deadlock, the search interval increases again towards the 5 second default
  - When a deadlock is detected, the lock monitor thread assumes that the next lock waits are entering a deadlock cycle and will automatically trigger a deadlock search, allowing a true deadlock to be detected immediately
- **During a deadlock search, the lock monitor identifies blocked tasks and then finds the blocking resource owner by recursively searching the tasks to identify the cyclic blocking that forms a deadlock**

Deadlock Priority

- **Any user can set the DEADLOCK_PRIORITY session option to control deadlock resolution behavior**
  - It is not possible to stop a user setting DEADLOCK_PRIORITY, even with Resource Governor
- **Setting a higher DEADLOCK_PRIORITY for important transactions will ensure that those transactions are not selected as the deadlock victim if a deadlock occurs with a lower priority session**
  - Setting DEADLOCK_PRIORITY should usually not be used by developers to prevent deadlock involving SELECT statements
  - The exception is where deadlocks cannot be prevented in other ways, and it is critical that the SELECT succeeds

Deadlock Victim Selection

- **When a deadlock is detected, the lock monitor ends it by choosing one of the threads as the deadlock victim**
  - The deadlock victim is killed, rolling back its transaction
  - The client receives a 1205 error
- **The deadlock victim is selected based on the following criteria:**
  - The DEADLOCK_PRIORITY of the two sessions is compared and the lowest priority session is selected as the victim
  - If both of the sessions have the same DEADLOCK_PRIORITY value, the transaction that is the least expensive to rollback, based on the log records that have been generated, is selected as the victim (default)

Deadlock Resolution in SQLCLR

- **Deadlock detection extends into the hosted SQLCLR stack for synchronization resources accessed inside managed procedures**
  - Synchronization resources include monitors, reader/writer locks and thread joins
  - Deadlock resolution in SQLCLR occurs by throwing an exception in the procedure that was selected as the victim
  - Resources are not automatically released by SQLCLR, so they must be explicitly released by the exception handling code in the procedure

Summary

- **The database engine automatically eliminates deadlocks by selecting the process with the least amount of log generation or of the lowest priority as the victim and kills the process**
- **Manual intervention is not necessary to eliminate deadlocks when they occur inside of SQL Server**

- **The next module will look at:**
  - Collecting deadlock information