**Monitoring Deadlocks**

“A deadlock occurs when two or more tasks permanently block each other by each task having a lock on a resource which the other tasks are trying to lock.”

A *deadlock* is a special blocking scenario in which two processes get blocked by each other. Each process, while holding its own resources, attempts to access a resource that is locked by the other process. This will lead to a blocking scenario known as a *deadly embrace*,

In other words, Deadlock is a situation when two or more processes holds a lock on a different resource try to get a lock on the other process resource which is already locked. This will eventually create an endless loop. SQL Server knows how to deal with those kind of situations with the “Deadlock detection” mechanism.

The mechanism continuously (every five seconds) checks the lock monitor thread and searches for deadlocks.

If a deadlock occurs the mechanism choses a process “victim” and rolls back his transaction. The victim is the process that its rollback will need the less resources.

Note – If we want we can predefine the process importance level and then the victim will be chosen by our definition.

Rollback the victim process will enable the other process to finish its transaction and that will end the loop.

It will be wise to create a “try and catch” mechanism in the processes transactions that will re-execute the victim process after a few random seconds.

Here is a script as an example of a simple and classical deadlock, which contains two transactions for each query window.

First Query

|  |  |
| --- | --- |
|  | -- --Transaction A--  USE AdventureworksCTP3  GO  BEGIN TRANSACTION  --Statement 1--  UPDATE Sales.SalesOrderDetail  SET OrderQty = OrderQty \* 2  WHERE SalesOrderID = 43659 and SalesOrderDetailID = 1  --Hold for 10 seconds--  WAITFOR DELAY '00:00:10'  --Statement 2--  SELECT \* FROM HumanResources.Department  WHERE DepartmentID = 1  COMMIT  GO |

 Second Query

|  |  |
| --- | --- |
|  |  |

 USE AdventureworksCTP3

GO

BEGIN TRANSACTION

--Statement 1--

UPDATE HumanResources.Department

SET Name = Name + ' added text'

WHERE DepartmentID = 1

--Hold for 10 seconds--

WAITFOR DELAY '00:00:10'

--Statement 2--

SELECT \* FROM Sales.SalesOrderDetail

WHERE SalesOrderID = 43659 and SalesOrderDetailID = 1

COMMIT

GO

---Enable following traces to capture deadlcok

dbcc traceon(1204,1)

dbcc traceon(1222,1)

The scripts runs on AdventureworksCTP3database.

Script flow:

1. Open two queries with two transactions (each query resemble each process).
2. Each process updates different table.
3. Each process waits ten seconds (in order to execute the other query).
4. Each process tries to select the other process table.
5. You will be able to see in the end that one process finished its transaction and the other one will print a massage that a deadlock occurred.

Now that we understand what is a deadlock, these are the main tools to monitor and identify deadlocks:

**Windows performance monitor.**

This tool allows you to see if a deadlock occurs on your server.

In order to start we need to open “windows performance monitor” and to add the “deadlocksec” counter in the “sql server: locks category”. All is left to do is run again the two script queries that creates the deadlock or wait for a deadlock to occur on the server. We will be able to see the changes on the graph.

**SQL Server trace flags:**

You can monitor and analyze deadlocks with two sql server trace flags.

Trace flag 1204 and trace flag 1222. They are similar and will write the deadlock description to the sql server error logs. The difference between them is that trace flag 1204 reports for each process separately and trace flag 1222 reports firstly on the processes and then on the resources. It is possible to turn both flags on by typing the command:

DBCC TRACEON (trace number) for each trace flag.

According to the explanation:

1. turn on both trace flag
2. execute both scripts
3. open SQL Server Error logs to see which processes were participate in the deadlock, who was chosen to be the victim and most important, what is the transactions code for future analyze.

**SQL Server profiler:**

The sql server profiler lets you monitor almost everything that occurs on your server including deadlocks. In my opinion this is the easiest and most efficient tool to monitor and analyze deadlocks. With this tool we are able to see in a shape of a graph:

1. Which processes participate in the deadlock.
2. Which resources they already locked.
3. Which resource they are trying to lock.
4. Who was chosen to be the victim.
5. Transactions code.

All we got to do is:

1. Open the sql server profiler.
2. On the “general” tab at “use the template” row choose “TSQL\_Locks”.
3. On the “Event Selection” tab mark only the “deadlock Graph” under locks section.
4. Run the trace and wait for the deadlock to occur.

SQL Server profile example: