(T25)處理DeadLock(死鎖)  
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc  
=======================================================================  
(T25)處理DeadLock(死鎖)  
=======================================================================  
0. Summary

1. Deadlock Example

2. Deadlock Priority : Same Deadlock Priority, different expensive to rollback

3. Deadlock Priority : different DEADLOCK\_PRIORITY

-----------

4. Deadlock Analysis And Prevention

4.1. Logging Dead locks

4.2. Clean up

4.3. Deadlock Analysis And Prevention

-----------

5. Deadlock Analysis And Prevention

6. Deadlock Error Handling

7. Asp.Net Handling Deadlocks

-----------

8. Asp.Net Handling Deadlocks

8.1. Set up SQL Authentication

8.2. Create Web Application

8.3. Code

8.3.1. Web.config

8.3.2. WebForm1.aspx

8.3.3. WebForm1.aspx.cs

8.3.4. WebForm2.aspx

8.3.5. WebForm2.aspx.cs  
=======================================================================

0. Summary

1.

DEADLOCK\_PRIORITY

1.1.

--SET DEADLOCK\_PRIORITY LOW;

--SET DEADLOCK\_PRIORITY -5;

--SET DEADLOCK\_PRIORITY NORMAL;

--SET DEADLOCK\_PRIORITY 0;

--SET DEADLOCK\_PRIORITY HIGH;

--SET DEADLOCK\_PRIORITY 5;

The default value of DEADLOCK\_PRIORITY is 0 which means NORMAL.

DEADLOCK\_PRIORITY value can between -10 to 10.

DEADLOCK\_PRIORITY value,-5 means LOW, 5 means HIGH

1.2.

deadlock victim selection:

1.2.1.

if both transaction has the different DEADLOCK\_PRIORITY,

the transaction with the lowest DEADLOCK\_PRIORITY will be the deadlock victim.

1.2.2.

if both transaction has the same DEADLOCK\_PRIORITY,

the transaction that is least expensive to rollback will be the deadlock victim.

1.2.3.

if both transaction has the same DEADLOCK\_PRIORITY and same cost to roll back,

the transaction will be chosen randomly to be the deadlock victim.

-------------------------------------------------------

2.

Logging Dead locks

2.1.

Syntax:

--DBCC Traceon(1222, -1)

Turn On the trace flag

--DBCC TraceStatus(1222, -1)

Check the Trace Status

...Deadlock occur...

--execute sp\_readerrorlog

Read the Error log.

--DBCC Traceoff(1222, -1)

Turn Off the trace flag

...

--EXECUTE sp\_readerrorlog;

To read the error log

2.2.

DBCC means Database Console Command.

SQL Server trace flag 1222 to write the deadlock information

to the SQL Server error log is one of the ways to

track down the queries that are causing deadlocks.

2.3.

-1 parameter means set the flag to global level.

Without -1 parameter means the flag is only valid at the current session level.

-------------------------------------------------------

3.

--BEGIN

--    BEGIN TRY

--        BEGIN TRAN;

--        --...Do Something...

--        COMMIT TRANSACTION;

--    END TRY

--    BEGIN CATCH

--        --\*\*\*\*

--        --Check if dead lock exists, ERROR\_NUMBER 1205 is deadlock error flag

--        IF ( ERROR\_NUMBER() = 1205 )

--            BEGIN

--                --...Do Something...

--            END;

--    END CATCH;

--END;

==================================================

1. Deadlock Example

A picture containing timeline

Description automatically generated

--=====================================================================

--T025\_01\_DeadlockExample

--=====================================================================

--=====================================================================

--T025\_01\_01

--Create Sample Data

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableA' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableA;

        DROP TABLE TableA;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableB' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableB;

        DROP TABLE TableB;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE TableA

    (

      ID INT IDENTITY

             PRIMARY KEY ,

      Name NVARCHAR(50)

    );

GO -- Run the previous command and begins new batch

INSERT  INTO TableA

VALUES  ( 'TableAName1' );

INSERT  INTO TableA

VALUES  ( 'TableAName2' );

INSERT  INTO TableA

VALUES  ( 'TableAName3' );

INSERT  INTO TableA

VALUES  ( 'TableAName4' );

INSERT  INTO TableA

VALUES  ( 'TableAName5' );

GO -- Run the previous command and begins new batch

CREATE TABLE TableB

    (

      ID INT IDENTITY

             PRIMARY KEY ,

      Name NVARCHAR(50)

    );

INSERT  INTO TableB

VALUES  ( 'TableBName1' );

INSERT  INTO TableB

VALUES  ( 'TableBName2' );

INSERT  INTO TableB

VALUES  ( 'TableBName3' );

INSERT  INTO TableB

VALUES  ( 'TableBName4' );

INSERT  INTO TableB

VALUES  ( 'TableBName5' );

GO -- Run the previous command and begins new batch

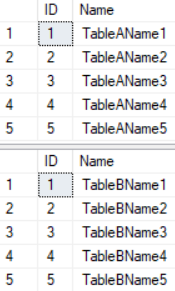
SELECT  \*

FROM    TableA;

SELECT  \*

FROM    TableB;

GO -- Run the previous command and begins new batch



--=====================================================================

--T025\_01\_02

--Dead Lock Example

---------------------------------------------------------------------

--T025\_01\_02\_01

-- Transaction1

BEGIN TRAN;

UPDATE  TableA

SET     [Name] += ' Tran1'

WHERE   ID = 1;

-- Do something

WAITFOR DELAY '00:00:4';

UPDATE  TableB

SET     [Name] += ' Tran1'

WHERE   ID = 1;

COMMIT TRANSACTION;

GO -- Run the previous command and begins new batch

---------------------------------------------------------------------

--T025\_01\_02\_02

-- Transaction2

BEGIN TRAN;

UPDATE  TableB

SET     [Name] += 'Tran2'

WHERE   ID = 1;

-- Do something

WAITFOR DELAY '00:00:4';

UPDATE  TableA

SET     [Name] += 'Tran2'

WHERE   ID = 1;

COMMIT TRANSACTION;

GO -- Run the previous command and begins new batch

---------------------------------------------------------------------

--T025\_01\_02\_03

--Check result

SELECT  \*

FROM    dbo.TableA

WHERE   ID = 1;

SELECT  \*

FROM    dbo.TableB

WHERE   ID = 1;

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

/\*

1.

Execute Transaction1 first, then in the mean time, execute Transaction2.

1.1.

Transaction1 will start to update TableA ID=1 record,

so TableA ID=1 is locked by Transaction1.

Transaction2 will start to update TableB ID=1 record,

so TableB ID=1 is locked by Transaction2.

1.2.

Both Transaction1 and Transaction2 has to do something

and wait for a few seconds.

1.3.

Transaction1 will start to update TableB ID=1 record,

but TableB ID=1 is locked by Transaction2 at that moment.

Transaction2 will start to update TableA ID=1 record,

but TableA ID=1 is locked by Transaction1 at that moment.

1.4.

After a few seconds, one of Transaction will complete successfully,

while the other one will be made the deadlock victim.

\*/

--=====================================================================

--T025\_01\_03

--clean up

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableA' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableA;

        DROP TABLE TableA;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableB' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableB;

        DROP TABLE TableB;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE TableA

    (

      ID INT IDENTITY

             PRIMARY KEY ,

      Name NVARCHAR(50)

    );

GO -- Run the previous command and begins new batch

INSERT  INTO TableA

VALUES  ( 'TableAName1' );

INSERT  INTO TableA

VALUES  ( 'TableAName2' );

INSERT  INTO TableA

VALUES  ( 'TableAName3' );

INSERT  INTO TableA

VALUES  ( 'TableAName4' );

INSERT  INTO TableA

VALUES  ( 'TableAName5' );

GO -- Run the previous command and begins new batch

CREATE TABLE TableB

    (

      ID INT IDENTITY

             PRIMARY KEY ,

      Name NVARCHAR(50)

    );

INSERT  INTO TableB

VALUES  ( 'TableBName1' );

INSERT  INTO TableB

VALUES  ( 'TableBName2' );

INSERT  INTO TableB

VALUES  ( 'TableBName3' );

INSERT  INTO TableB

VALUES  ( 'TableBName4' );

INSERT  INTO TableB

VALUES  ( 'TableBName5' );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    TableA;

SELECT  \*

FROM    TableB;

GO -- Run the previous command and begins new batch

Table

Description automatically generated with medium confidence

==================================================

2. Deadlock Priority : Same Deadlock Priority, different expensive to rollback

--=====================================================================

--T025\_02\_Deadlock Priority : Same Deadlock Priority, different expensive to rollback

--=====================================================================

/\*

1.

DEADLOCK\_PRIORITY

1.1.

--SET DEADLOCK\_PRIORITY LOW;

--SET DEADLOCK\_PRIORITY -5;

--SET DEADLOCK\_PRIORITY NORMAL;

--SET DEADLOCK\_PRIORITY 0;

--SET DEADLOCK\_PRIORITY HIGH;

--SET DEADLOCK\_PRIORITY 5;

The default value of DEADLOCK\_PRIORITY is 0 which means NORMAL.

DEADLOCK\_PRIORITY value can between -10 to 10.

DEADLOCK\_PRIORITY value,-5 means LOW, 5 means HIGH

1.2.

deadlock victim selection:

1.2.1.

if both transaction has the different DEADLOCK\_PRIORITY,

the transaction with the lowest DEADLOCK\_PRIORITY will be the deadlock victim.

1.2.2.

if both transaction has the same DEADLOCK\_PRIORITY,

the transaction that is least expensive to rollback will be the deadlock victim.

1.2.3.

if both transaction has the same DEADLOCK\_PRIORITY and same cost to roll back,

the transaction will be chosen randomly to be the deadlock victim.

2.

--SET DEADLOCK\_PRIORITY NORMAL;

If DEADLOCK\_PRIORITY is the same,

the transaction that is least expensive to rollback is selected as the deadlock victim

\*/

--==========================================================================

--T025\_02\_01

--Create Sample Data

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableA' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableA;

        DROP TABLE TableA;

    END;

GO -- Run the previous command and begins new batch

--clean up

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableB' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableB;

        DROP TABLE TableB;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE TableA

    (

      ID INT IDENTITY

             PRIMARY KEY ,

      Name NVARCHAR(50)

    );

GO -- Run the previous command and begins new batch

INSERT  INTO TableA

VALUES  ( 'TableAName1' );

INSERT  INTO TableA

VALUES  ( 'TableAName2' );

INSERT  INTO TableA

VALUES  ( 'TableAName3' );

INSERT  INTO TableA

VALUES  ( 'TableAName4' );

INSERT  INTO TableA

VALUES  ( 'TableAName5' );

GO -- Run the previous command and begins new batch

CREATE TABLE TableB

    (

      ID INT IDENTITY

             PRIMARY KEY ,

      Name NVARCHAR(50)

    );

INSERT  INTO TableB

VALUES  ( 'TableBName1' );

INSERT  INTO TableB

VALUES  ( 'TableBName2' );

INSERT  INTO TableB

VALUES  ( 'TableBName3' );

INSERT  INTO TableB

VALUES  ( 'TableBName4' );

INSERT  INTO TableB

VALUES  ( 'TableBName5' );

GO -- Run the previous command and begins new batch

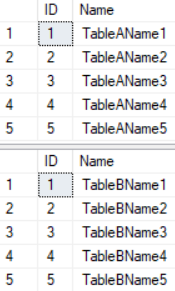
SELECT  \*

FROM    TableA;

SELECT  \*

FROM    TableB;

GO -- Run the previous command and begins new batch



--==========================================================================

--T025\_02\_02

/\*

--SET DEADLOCK\_PRIORITY NORMAL;

If DEADLOCK\_PRIORITY is the same,

the transaction that is least expensive to rollback is selected as the deadlock victim

\*/

--------------------------

--T025\_02\_02\_01

-- Transaction1

BEGIN TRAN;

UPDATE  TableA

SET     [Name] += ' Tran1'

WHERE   ID IN ( 1, 2, 3, 4, 5 );

-- Do something

WAITFOR DELAY '00:00:4';

UPDATE  TableB

SET     [Name] += ' Tran1'

WHERE   ID = 1;

COMMIT TRANSACTION;

GO -- Run the previous command and begins new batch

--------------------------

--T025\_02\_02\_02

-- Transaction2

BEGIN TRAN;

UPDATE  TableB

SET     [Name] += ' Tran2'

WHERE   ID = 1;

-- Do something

WAITFOR DELAY '00:00:4';

UPDATE  TableA

SET     [Name] += ' Tran2'

WHERE   ID IN ( 1, 2, 3, 4, 5 );

COMMIT TRANSACTION;

GO -- Run the previous command and begins new batch

--------------------------

--T025\_02\_02\_03

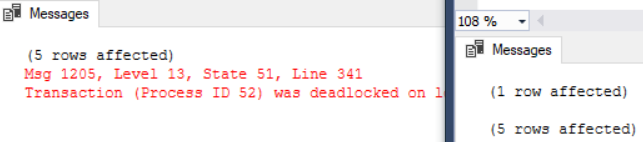
--Check result

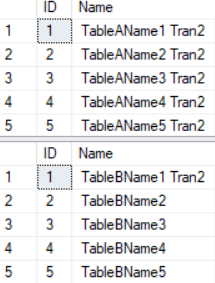
SELECT  \*

FROM    dbo.TableA;

SELECT  \*

FROM    dbo.TableB;





/\*

1.

Execute Transaction1 first, then in the mean time, execute Transaction2.

1.1.

Transaction1 will start to update TableA ID=1,2,3,4,5 record,

so TableA ID=1,2,3,4,5 are locked by Transaction1.

Transaction2 will start to update TableB ID=1 record,

so TableB ID=1 is locked by Transaction2.

1.2.

Both Transaction1 and Transaction2 has to do something

and wait for a few seconds.

1.3.

Transaction1 will start to update TableB ID=1 record,

but TableB ID=1 is locked by Transaction2 at that moment.

Transaction2 will start to update TableA ID=1 record,

but TableA ID=1,2,3,4,5 are locked by Transaction1 at that moment.

1.4.

Both the transaction have the same default DEADLOCK\_PRIORITY NORMAL.

Transaction2 is least expensive to roll back.

After a few seconds, Transaction1 will complete successfully,

and Transaction2 will be the deadlock victim.

1.5.

Transaction1 one output:

--(5 rows affected)

--(1 row affected)

Transaction2 one output:

--(1 row affected)

--Msg 1205, Level 13, State 51, Line 9

--Transaction (Process ID 55) was deadlocked on lock resources

--with another process and has been chosen as the deadlock victim.

-- Rerun the transaction.

\*/

--==========================================================================

--T025\_02\_03

--Clean up

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableA' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableA;

        DROP TABLE TableA;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableB' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableB;

        DROP TABLE TableB;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE TableA

    (

      ID INT IDENTITY

             PRIMARY KEY ,

      Name NVARCHAR(50)

    );

GO -- Run the previous command and begins new batch

INSERT  INTO TableA

VALUES  ( 'TableAName1' );

INSERT  INTO TableA

VALUES  ( 'TableAName2' );

INSERT  INTO TableA

VALUES  ( 'TableAName3' );

INSERT  INTO TableA

VALUES  ( 'TableAName4' );

INSERT  INTO TableA

VALUES  ( 'TableAName5' );

GO -- Run the previous command and begins new batch

CREATE TABLE TableB

    (

      ID INT IDENTITY

             PRIMARY KEY ,

      Name NVARCHAR(50)

    );

INSERT  INTO TableB

VALUES  ( 'TableBName1' );

INSERT  INTO TableB

VALUES  ( 'TableBName2' );

INSERT  INTO TableB

VALUES  ( 'TableBName3' );

INSERT  INTO TableB

VALUES  ( 'TableBName4' );

INSERT  INTO TableB

VALUES  ( 'TableBName5' );

GO -- Run the previous command and begins new batch

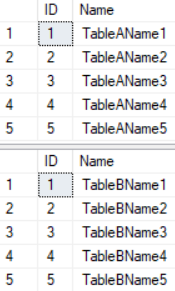
SELECT  \*

FROM    TableA;

SELECT  \*

FROM    TableB;

GO -- Run the previous command and begins new batch



==================================================

3. Deadlock Priority : different DEADLOCK\_PRIORITY

--=====================================================================

--T025\_03\_Deadlock Priority : different DEADLOCK\_PRIORITY

--=====================================================================

/\*

--SET DEADLOCK\_PRIORITY HIGH;

if both transaction has the different DEADLOCK\_PRIORITY,

the transaction with the lowest DEADLOCK\_PRIORITY will be the deadlock victim.

\*/

--=====================================================================

--T025\_03\_01

--Deadlock Priority : different DEADLOCK\_PRIORITY

--------------------------

--T025\_03\_01\_01

-- Transaction1

BEGIN TRAN;

UPDATE  TableA

SET     [Name] += ' Tran1'

WHERE   ID IN ( 1, 2, 3, 4, 5 );

-- Do something

WAITFOR DELAY '00:00:4';

UPDATE  TableB

SET     [Name] += ' Tran1'

WHERE   ID = 1;

COMMIT TRANSACTION;

GO -- Run the previous command and begins new batch

--------------------------

--T025\_03\_01\_02

-- Transaction2

SET DEADLOCK\_PRIORITY HIGH;

GO -- Run the previous command and begins new batch

BEGIN TRAN;

UPDATE  TableB

SET     [Name] += ' Tran2'

WHERE   ID = 1;

-- Do something

WAITFOR DELAY '00:00:4';

UPDATE  TableA

SET     [Name] += ' Tran2'

WHERE   ID IN ( 1, 2, 3, 4, 5 );

COMMIT TRANSACTION;

GO -- Run the previous command and begins new batch

--------------------------

--T025\_03\_01\_03

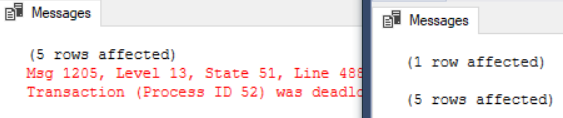
--Check result

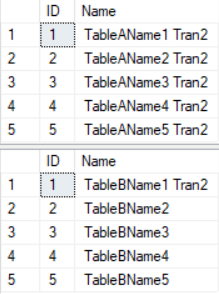
SELECT  \*

FROM    dbo.TableA;

SELECT  \*

FROM    dbo.TableB;





/\*

1.

Execute Transaction1 first, then in the mean time, execute Transaction2.

1.1.

Transaction1 will start to update TableA ID=1,2,3,4,5 record,

so TableA ID=1,2,3,4,5 are locked by Transaction1.

Transaction2 will start to update TableB ID=1 record,

so TableB ID=1 is locked by Transaction2.

1.2.

Both Transaction1 and Transaction2 has to do something

and wait for a few seconds.

1.3.

Transaction1 will start to update TableB ID=1 record,

but TableB ID=1 is locked by Transaction2 at that moment.

Transaction2 will start to update TableA ID=1 record,

but TableA ID=1,2,3,4,5 are locked by Transaction1 at that moment.

1.4.

if both transaction has the different DEADLOCK\_PRIORITY,

the transaction with the lowest DEADLOCK\_PRIORITY will be the deadlock victim.

In this case, Transaction2 has higher DEADLOCK\_PRIORITY.

Thus, Transaction1 will be the deadlock victim.

1.5.

Transaction1 one output:

--(1 row affected)

--Msg 1205, Level 13, State 51, Line 9

--Transaction (Process ID 55) was deadlocked on lock resources

--with another process and has been chosen as the deadlock victim.

-- Rerun the transaction.

Transaction2 one output:

--(5 rows affected)

--(1 row affected)

\*/

--=====================================================================

--T025\_03\_02

--clean up

SET DEADLOCK\_PRIORITY NORMAL;

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableA' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableA;

        DROP TABLE TableA;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableB' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableB;

        DROP TABLE TableB;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE TableA

    (

      ID INT IDENTITY

             PRIMARY KEY ,

      Name NVARCHAR(50)

    );

GO -- Run the previous command and begins new batch

INSERT  INTO TableA

VALUES  ( 'TableAName1' );

INSERT  INTO TableA

VALUES  ( 'TableAName2' );

INSERT  INTO TableA

VALUES  ( 'TableAName3' );

INSERT  INTO TableA

VALUES  ( 'TableAName4' );

INSERT  INTO TableA

VALUES  ( 'TableAName5' );

GO -- Run the previous command and begins new batch

CREATE TABLE TableB

    (

      ID INT IDENTITY

             PRIMARY KEY ,

      Name NVARCHAR(50)

    );

INSERT  INTO TableB

VALUES  ( 'TableBName1' );

INSERT  INTO TableB

VALUES  ( 'TableBName2' );

INSERT  INTO TableB

VALUES  ( 'TableBName3' );

INSERT  INTO TableB

VALUES  ( 'TableBName4' );

INSERT  INTO TableB

VALUES  ( 'TableBName5' );

GO -- Run the previous command and begins new batch

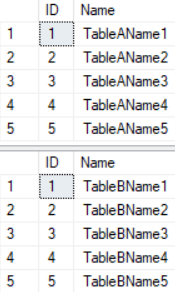
SELECT  \*

FROM    TableA;

SELECT  \*

FROM    TableB;

GO -- Run the previous command and begins new batch



4. Deadlock Analysis And Prevention

--=====================================================================

--T025\_04\_Deadlock Analysis And Prevention

--=====================================================================

/\*

1.

Logging Dead locks

1.1.

Syntax:

--DBCC Traceon(1222, -1)

Turn On the trace flag

--DBCC TraceStatus(1222, -1)

Check the Trace Status

Status==1 means trace flag is enabled.

...Deadlock occur...

--execute sp\_readerrorlog

Read the Error log.

Search for "deadlock-list" which

contains dead lock information.

--DBCC Traceoff(1222, -1)

Turn Off the trace flag

Status==0 means trace flag is disabled.

...

--EXECUTE sp\_readerrorlog;

To read the error log

1.2.

DBCC means Database Console Command.

SQL Server trace flag 1222 to write the deadlock information

to the SQL Server error log is one of the ways to

track down the queries that are causing deadlocks.

1.3.

-1 parameter means set the flag to global level.

Without -1 parameter means the flag is only valid at the current session level.

\*/

4.1. Logging Dead locks

--=====================================================================

--T025\_04\_01

--Logging Dead locks

--------------------------

--T025\_04\_01\_01

--Turn On the trace flag

DBCC TRACEON(1222, -1);

GO -- Run the previous command and begins new batch



--------------------------

--T025\_04\_01\_02

--Check the Trace Status.

DBCC TRACESTATUS(1222, -1);

GO -- Run the previous command and begins new batch



/\*

1.

Logging Dead locks

1.1.

Syntax:

--DBCC Traceon(1222, -1)

Turn On the trace flag

--DBCC TraceStatus(1222, -1)

Check the Trace Status

Status==1 means trace flag is enabled.

...Deadlock occur...

--execute sp\_readerrorlog

Read the Error log.

Search for "deadlock-list" which

contains dead lock information.

--DBCC Traceoff(1222, -1)

Turn Off the trace flag

Status==0 means trace flag is disabled.

...

--EXECUTE sp\_readerrorlog;

To read the error log

1.2.

DBCC means Database Console Command.

SQL Server trace flag 1222 to write the deadlock information

to the SQL Server error log is one of the ways to

track down the queries that are causing deadlocks.

1.3.

-1 parameter means set the flag to global level.

Without -1 parameter means the flag is only valid at the current session level.

\*/

--------------------------

--T025\_04\_01\_03

-- Transaction1

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spTran1' ) )

    BEGIN

        DROP PROCEDURE spTran1;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spTran1

AS

    BEGIN

        BEGIN TRAN;

        UPDATE  TableA

        SET     [Name] += ' Tran1'

        WHERE   ID IN ( 1, 2, 3, 4, 5 );

             -- Do something

        WAITFOR DELAY '00:00:4';

        UPDATE  TableB

        SET     [Name] += ' Tran1'

        WHERE   ID = 1;

        COMMIT TRANSACTION;

    END;

GO -- Run the previous command and begins new batch

EXECUTE spTran1;

GO -- Run the previous command and begins new batch

--------------------------

--T025\_04\_01\_04

-- Transaction2

--If store procedure exists then DROP it

--IF OBJECT\_ID('spTran2') IS NOT NULL

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spTran2' ) )

    BEGIN

        DROP PROCEDURE spTran2;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spTran2

AS

    BEGIN

        BEGIN TRAN;

        UPDATE  TableB

        SET     [Name] += ' Tran2'

        WHERE   ID = 1;

             -- Do something

        WAITFOR DELAY '00:00:4';

        UPDATE  TableA

        SET     [Name] += ' Tran2'

        WHERE   ID IN ( 1, 2, 3, 4, 5 );

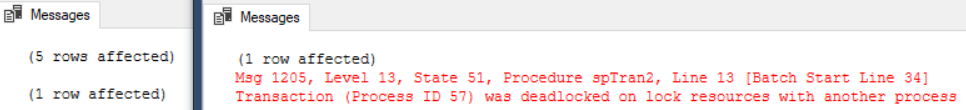
        COMMIT TRANSACTION;

    END;

GO -- Run the previous command and begins new batch

EXECUTE spTran2;

GO -- Run the previous command and begins new batch



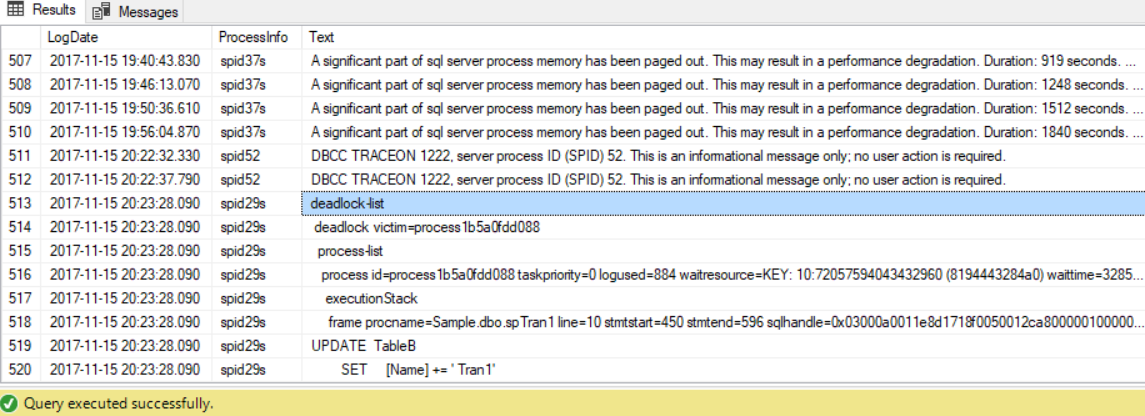
--------------------------

--T025\_04\_01\_05

--To read the error log

EXECUTE sp\_readerrorlog;

GO -- Run the previous command and begins new batch



--------------------------

--T025\_04\_01\_06

--Turn Off the trace flag

DBCC TRACEOFF(1222, -1);

GO -- Run the previous command and begins new batch



--------------------------

--T025\_04\_01\_06

--Check the Trace Status.

DBCC TRACESTATUS(1222, -1);

GO -- Run the previous command and begins new batch

Table

Description automatically generated

/\*

1.

Logging Dead locks

1.1.

--DBCC Traceon(1222, -1)

Turn On the trace flag

--DBCC TraceStatus(1222, -1)

Check the Trace Status

...Deadlock occur...

--execute sp\_readerrorlog

Read the Error log.

--DBCC Traceoff(1222, -1)

Turn Off the trace flag

1.2.

DBCC means Database Console Command.

SQL Server trace flag 1222 to write the deadlock information

to the SQL Server error log is one of the ways to

track down the queries that are causing deadlocks.

1.3.

-1 parameter means set the flag to global level.

Without -1 parameter means the flag is only valid at the current session level.

2.

Execute Transaction1 first, then in the mean time, execute Transaction2.

2.1.

Transaction1 will start to update TableA ID=1,2,3,4,5 record,

so TableA ID=1,2,3,4,5 are locked by Transaction1.

Transaction2 will start to update TableB ID=1 record,

so TableB ID=1 is locked by Transaction2.

2.2.

Both Transaction1 and Transaction2 has to do something

and wait for a few seconds.

2.3.

Transaction1 will start to update TableB ID=1 record,

but TableB ID=1 is locked by Transaction2 at that moment.

Transaction2 will start to update TableA ID=1 record,

but TableA ID=1,2,3,4,5 are locked by Transaction1 at that moment.

2.4.

Both the transaction have the same default DEADLOCK\_PRIORITY NORMAL.

Transaction2 is least expensive to roll back.

After a few seconds, Transaction1 will complete successfully,

and Transaction2 will be the deadlock victim.

2.5.

Transaction1 one output:

--(5 rows affected)

--(1 row affected)

Transaction2 one output:

--(1 row affected)

--Msg 1205, Level 13, State 51, Line 9

--Transaction (Process ID 55) was deadlocked on lock resources

--with another process and has been chosen as the deadlock victim.

-- Rerun the transaction.

\*/

4.2. Clean up

--=====================================================================

--T025\_04\_02

--clean up

SET DEADLOCK\_PRIORITY NORMAL;

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableA' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableA;

        DROP TABLE TableA;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableB' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableB;

        DROP TABLE TableB;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE TableA

(

  ID INT IDENTITY

         PRIMARY KEY ,

  Name NVARCHAR(50)

);

GO -- Run the previous command and begins new batch

INSERT  INTO TableA

VALUES  ( 'TableAName1' );

INSERT  INTO TableA

VALUES  ( 'TableAName2' );

INSERT  INTO TableA

VALUES  ( 'TableAName3' );

INSERT  INTO TableA

VALUES  ( 'TableAName4' );

INSERT  INTO TableA

VALUES  ( 'TableAName5' );

GO -- Run the previous command and begins new batch

CREATE TABLE TableB

(

  ID INT IDENTITY

         PRIMARY KEY ,

  Name NVARCHAR(50)

);

INSERT  INTO TableB

VALUES  ( 'TableBName1' );

INSERT  INTO TableB

VALUES  ( 'TableBName2' );

INSERT  INTO TableB

VALUES  ( 'TableBName3' );

INSERT  INTO TableB

VALUES  ( 'TableBName4' );

INSERT  INTO TableB

VALUES  ( 'TableBName5' );

GO -- Run the previous command and begins new batch

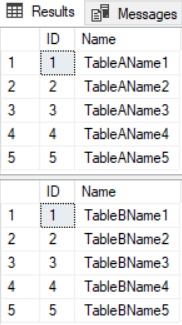
SELECT  \*

FROM    TableA;

SELECT  \*

FROM    TableB;

GO -- Run the previous command and begins new batch



4.3. Deadlock Analysis And Prevention

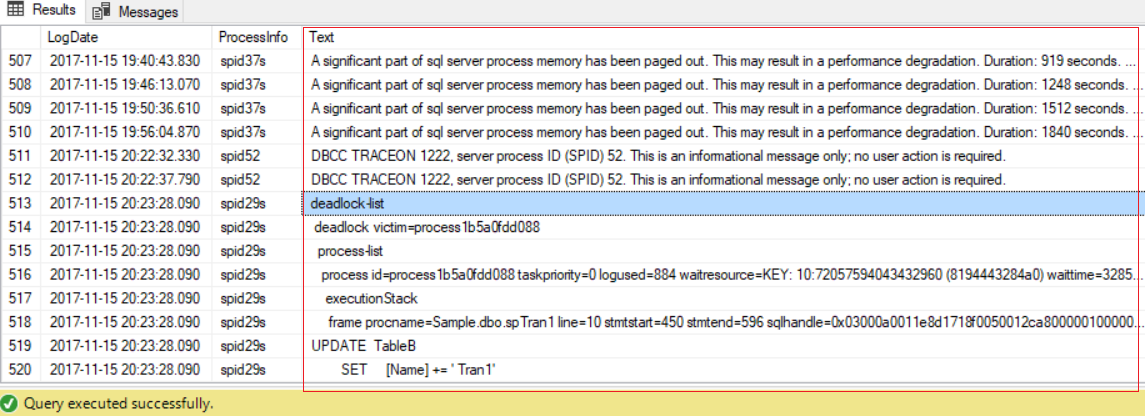
--=====================================================================

--T025\_04\_03

--Deadlock Analysis And Prevention

EXECUTE sp\_readerrorlog;

--To read the error log



/\*

1.

--To read the error log

execute sp\_readerrorlog

1.1.

Then copy the Text column into sublime or Notepad++

1.2.

There are 3 important sections, deadlock victim, process-list, and resource-list

in deadlock information from the Text of sp\_readerrorlog.

-----------

1.2.1.

deadlock victim :

deadlock victim contains the deadlock victim process id.

E.g.

--deadlock victim=process2e27a02fc28

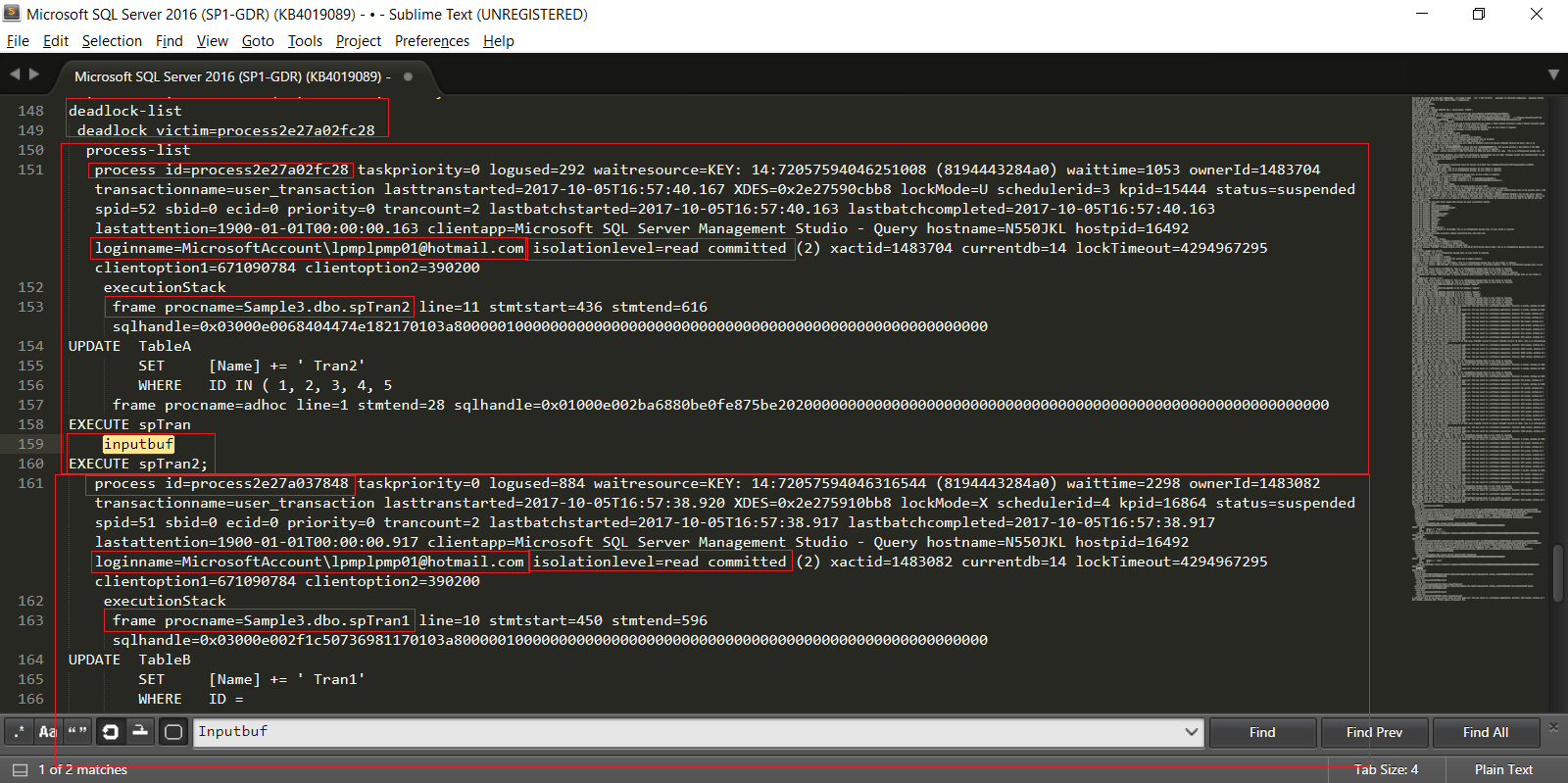
Ctrl+F to search keyword "deadlock victim"

Get the processID, process2e27a02fc28.

Ctrl+F to search "process2e27a02fc28" in the process-list

which contains the list of processes that participated in the deadlock.

-----------



1.2.2.

process-list:

process-list contains the list of participated processes of the deadlock.

There are some important sections regarding deadlock.

1.2.2.1.

loginname

loginname is the user loginname who perform the process.

E.g. MicrosoftAccount\UserName

1.2.2.2.

isolationlevel

isolationlevel is the thansaction isolation level of the used process.

E.g. read committed.

1.2.2.3.

procname

procname is the stored procedure name of the process.

E.g. spTran2

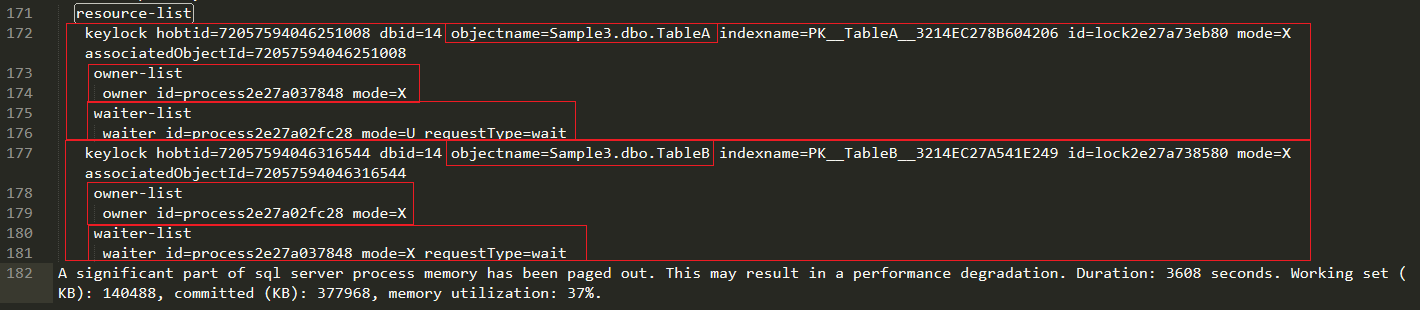
1.2.2.4.

Inputbuf

Inputbuf is the code of the process when the deadlock occured.

E.g. EXECUTE spTran2

-----------



1.2.3.

resource-list :

resource-list contains the list of Database Objects resource of the process

which participate the deadlock.

There are some important sections regarding deadlock.

1.2.3.1.

objectname

objectname is the Database Objects resource name of the process which participate the deadlock.

1.2.3.2.

owner-list

--owner-list

--   owner id=process2e27a037848 mode=X

owner-list contains the "owning process id" and the "owning process lock mode".

lock mode means how the Database Objects resource can be accessed by the current transaction.

這個object目前被哪個process給lock住了

S means Shared lock

U means update lock

X means exclusive lock

...ect.

1.2.3.3.

waiter-list

--waiter-list

--   waiter id=process2e27a02fc28 mode=U requestType=wait

waiter-list contains the "owning process id", the "owning process lock mode", and the "requestType"

目前是哪個process正在等待這個object

1.2.3.3.1.

"waiter id=process2e27a02fc28" is the "owning process id"

means the process that wants to acquire a lock on the resource.

1.2.3.3.2.

"mode=U" means the "owning process lock mode" is update lock which

means that process was doing update and get the block by the lock.

1.2.3.3.3.

"requestType=wait" means the that process was requested to wait the lock.

\*/

======================================================================

5. Deadlock Analysis And Prevention

--=====================================================================

--T025\_05\_Sql Profiler Capturing Deadlocks

--=====================================================================

--Add Deadlock graph event to the trace in SQL profiler

--=====================================================================

--T025\_05\_01

/\*

Step01:

Add Deadlock graph event to the trace in SQL profiler

Tools --> SQL Server Profiler

--> Select the database to connect

--> File --> New Trace

--> In Trace Properties window  --> In General Tab -->

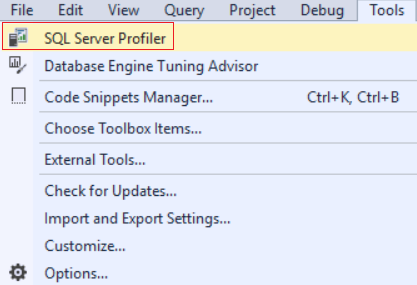
Use the template : Blank

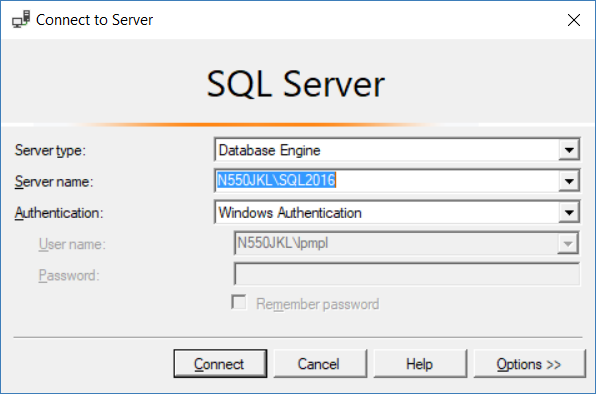
--> In Events Selection Tab

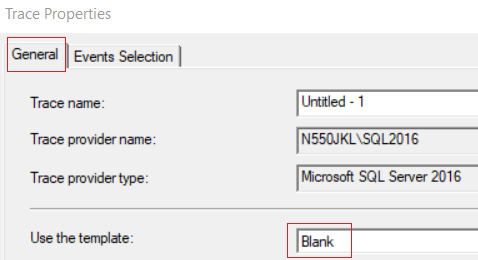
Locks --> Select  "Deadlock graph"

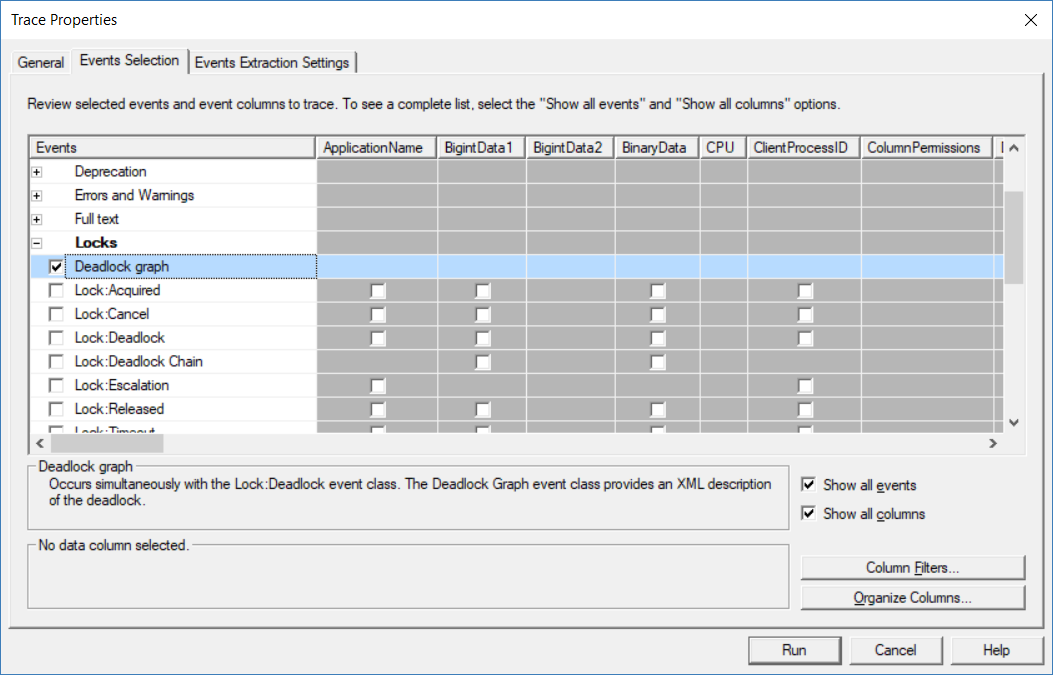
--> Run

\*/









--=====================================================================

--T025\_05\_02

--Step02:

--CREATE sample data

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableA' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableA;

        DROP TABLE TableA;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableB' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableB;

        DROP TABLE TableB;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE TableA

(

  ID INT IDENTITY

         PRIMARY KEY ,

  Name NVARCHAR(50)

);

INSERT  INTO TableA

VALUES  ( 'TableAName1' );

INSERT  INTO TableA

VALUES  ( 'TableAName2' );

INSERT  INTO TableA

VALUES  ( 'TableAName3' );

INSERT  INTO TableA

VALUES  ( 'TableAName4' );

INSERT  INTO TableA

VALUES  ( 'TableAName5' );

GO -- Run the previous command and begins new batch

CREATE TABLE TableB

(

  ID INT IDENTITY

         PRIMARY KEY ,

  Name NVARCHAR(50)

);

INSERT  INTO TableB

VALUES  ( 'TableBName1' );

INSERT  INTO TableB

VALUES  ( 'TableBName2' );

INSERT  INTO TableB

VALUES  ( 'TableBName3' );

INSERT  INTO TableB

VALUES  ( 'TableBName4' );

INSERT  INTO TableB

VALUES  ( 'TableBName5' );

GO -- Run the previous command and begins new batch

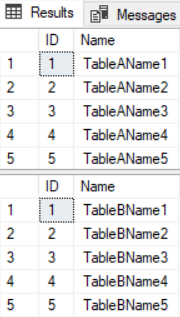
SELECT  \*

FROM    TableA;

SELECT  \*

FROM    TableB;

GO -- Run the previous command and begins new batch



--=====================================================================

--T025\_05\_03

----Step03: Transaction1

BEGIN TRAN;

UPDATE  TableA

SET     [Name] += ' Tran1'

WHERE   ID = 1;

-- Do something

WAITFOR DELAY '00:00:4';

UPDATE  TableB

SET     [Name] += ' Tran1'

WHERE   ID = 1;

COMMIT TRANSACTION;

GO -- Run the previous command and begins new batch

--=====================================================================

--T025\_05\_04

----Step03: Transaction2

BEGIN TRAN;

UPDATE  TableB

SET     [Name] += 'Tran2'

WHERE   ID = 1;

-- Do something

WAITFOR DELAY '00:00:4';

UPDATE  TableA

SET     [Name] += 'Tran2'

WHERE   ID = 1;

COMMIT TRANSACTION;

GO -- Run the previous command and begins new batch

A picture containing text

Description automatically generated

/\*

1.

Create the sample data first, then Open another Query for Transaction2

Execute Transaction1 first, then in the mean time, execute Transaction2.

1.1.

Transaction1 will start to update TableA ID=1 record,

so TableA ID=1 is locked by Transaction1.

Transaction2 will start to update TableB ID=1 record,

so TableB ID=1 is locked by Transaction2.

1.2.

Both Transaction1 and Transaction2 has to do something

and wait for a few seconds.

1.3.

Transaction1 will start to update TableB ID=1 record,

but TableB ID=1 is locked by Transaction2 at that moment.

Transaction2 will start to update TableA ID=1 record,

but TableA ID=1 is locked by Transaction1 at that moment.

1.4.

After a few seconds, one of Transaction will complete successfully,

while the other one will be made the deadlock victim.

\*/

--=====================================================================

--T025\_05\_05

/\*

There are several ways to track deadlock.

In previous example,

we use Logging Dead locks wtih the trace flag 1222

Here we use SQL profiler.

1.

Step01:

Add Deadlock graph event to the trace in SQL profiler

Tools --> SQL Server Profiler

--> Select the database to connect

--> File --> New Trace

--> In Trace Properties window  --> In General Tab -->

Use the template : Blank

--> In Events Selection Tab

Locks --> Select  "Deadlock graph"

--> Run

2.

Step02:

Create the sample data

3.

Step03:

Open another Query for Transaction2

Execute Transaction1 first, then go straight execute Transaction2.

4.

Step04:

In SQL Profile

--> Press Stop

--> File --> Export --> Extract SQL Server Events --> Extract Deadlock Events

-->

FileName : D:\DeadLockSample\DeadLockSample

Save as Type : Deadlock XML file (\*.xdl)

Export: each event in a separate file

-->

It will become one file, DeadLockSample\_1.xdl

5.

Step05:

Go back to SQL Profiler.

Select "Deadlock Graph" in the even class

Then you can see the deadlock graph.

--------------------

6.

6.1.

The oval with the blue cross on the deadlock graph

represents the deadlock victim transaction.

6.2.

The other oval on the deadlock graph

represents the transaction that executed successfully.

6.3.

When mouse point to the oval,

the pop out little window will display the SQL code that caused the deadlock.

6.4.

The oval represents the process node.

5.4.1.

--Server Process Id :

You may also see the Server Process Id

from the information bar at the bottom of SSMS.

6.4.2.

--Deadlock Priority : 0

0 means DEADLOCK\_PRIORITY is NORMAL

Revise the following :

--SET DEADLOCK\_PRIORITY LOW;

--SET DEADLOCK\_PRIORITY -5;

--SET DEADLOCK\_PRIORITY NORMAL;

--SET DEADLOCK\_PRIORITY 0;

--SET DEADLOCK\_PRIORITY HIGH;

--SET DEADLOCK\_PRIORITY 5;

The default value of DEADLOCK\_PRIORITY is 0 which means NORMAL.

DEADLOCK\_PRIORITY value can between -10 to 10.

DEADLOCK\_PRIORITY value,-5 means LOW, 5 means HIGH

6.4.3.

--Log Used :

Log Used represents the transaction log space used.

More Log Used means more expensive to roll back.

The deadlock victim is always the less Log Used

which means less expensive to roll back.

7.

The rectangles represent the resource nodes.

--HoBt ID : 72057594041663488

HoBt ID is Heap Or Binary Tree ID.

--SELECT  \*

--FROM    sys.partitions

--WHERE   hobt\_id = 72057594046644224;

**use "hobt\_id" to query sys.partitions**

to find the database objects involved in the deadlock.

**--SELECT  OBJECT\_NAME([object\_id])**

**--FROM    sys.partitions**

**--WHERE   hobt\_id = 72057594046644224;**

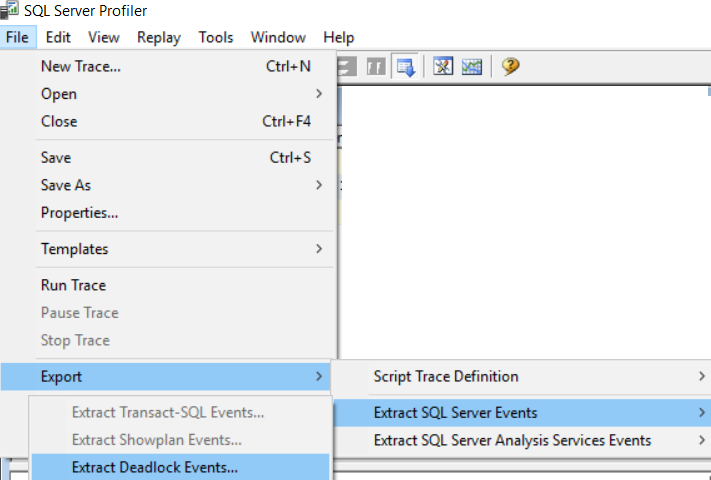
Use OBJECT\_NAME([object\_id]) to find out the database object name

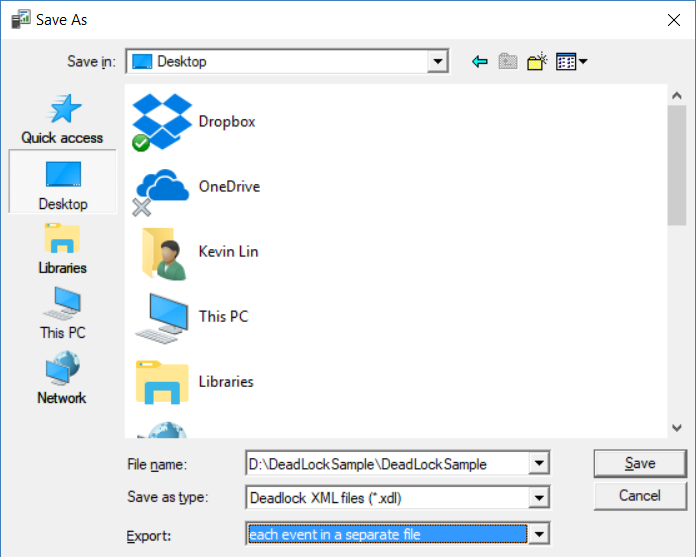
involved in the deadlock.

In this case that will return TableA

\*/





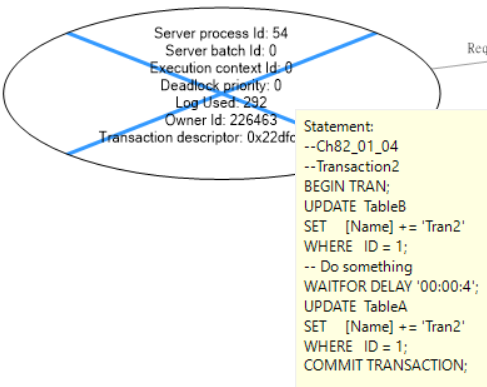


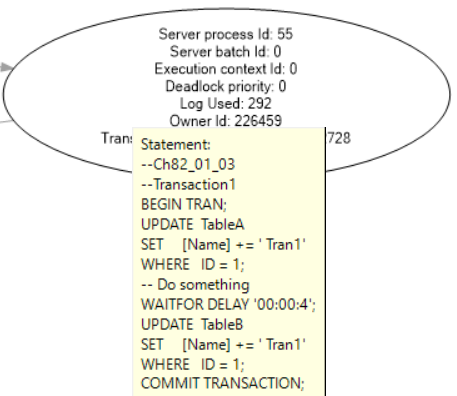
Graphical user interface, application, Word

Description automatically generated

Graphical user interface, table

Description automatically generated with medium confidence





Graphical user interface, application

Description automatically generated

--=====================================================================

--T025\_05\_06

--Step06: find the database object name involved in the deadlock.

SELECT  \*

FROM    sys.partitions

WHERE   hobt\_id = 72057594046644224;

SELECT  OBJECT\_NAME([object\_id])

FROM    sys.partitions

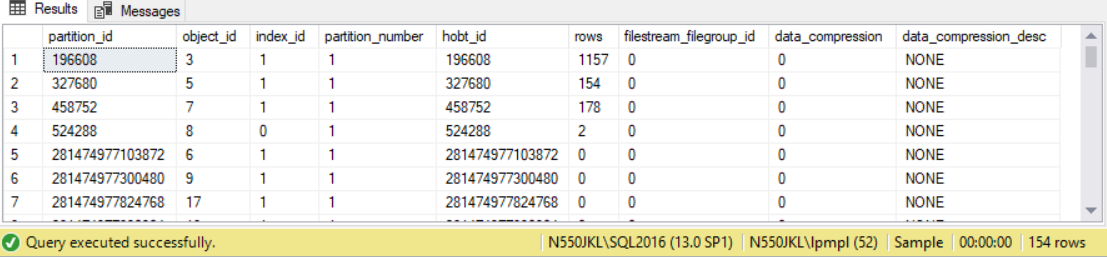
WHERE   hobt\_id = 72057594046644224;

-----------------------------------------------------------------

SELECT  \*

FROM    sys.partitions

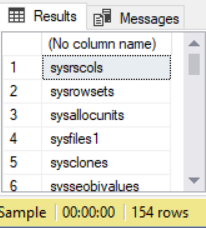
GO -- Run the previous command and begins new batch



SELECT  OBJECT\_NAME([object\_id])

FROM    sys.partitions

GO -- Run the previous command and begins new batch



--=====================================================================

--T025\_05\_07

--Step07: Check result

SELECT  \*

FROM    dbo.TableA

WHERE   ID = 1;

SELECT  \*

FROM    dbo.TableB

WHERE   ID = 1;

Graphical user interface, application

Description automatically generated

--=====================================================================

--T025\_05\_08

----Step08: clean up

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableA' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableA;

        DROP TABLE TableA;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableB' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableB;

        DROP TABLE TableB;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE TableA

(

  ID INT IDENTITY

         PRIMARY KEY ,

  Name NVARCHAR(50)

);

INSERT  INTO TableA

VALUES  ( 'TableAName1' );

INSERT  INTO TableA

VALUES  ( 'TableAName2' );

INSERT  INTO TableA

VALUES  ( 'TableAName3' );

INSERT  INTO TableA

VALUES  ( 'TableAName4' );

INSERT  INTO TableA

VALUES  ( 'TableAName5' );

GO -- Run the previous command and begins new batch

CREATE TABLE TableB

(

  ID INT IDENTITY

         PRIMARY KEY ,

  Name NVARCHAR(50)

);

INSERT  INTO TableB

VALUES  ( 'TableBName1' );

INSERT  INTO TableB

VALUES  ( 'TableBName2' );

INSERT  INTO TableB

VALUES  ( 'TableBName3' );

INSERT  INTO TableB

VALUES  ( 'TableBName4' );

INSERT  INTO TableB

VALUES  ( 'TableBName5' );

GO -- Run the previous command and begins new batch

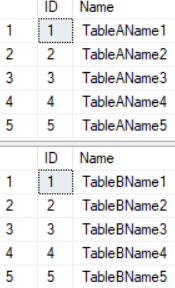
SELECT  \*

FROM    TableA;

SELECT  \*

FROM    TableB;

GO -- Run the previous command and begins new batch



======================================================================

6. Deadlock Error Handling

--=====================================================================

--T025\_06\_Deadlock Error Handling

--=====================================================================

--===========================================================================

--T025\_06\_01

--Create Sample data

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableA' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableA;

        DROP TABLE TableA;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableB' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableB;

        DROP TABLE TableB;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE TableA

(

  ID INT IDENTITY

         PRIMARY KEY ,

  Name NVARCHAR(50)

);

GO -- Run the previous command and begins new batch

INSERT  INTO TableA

VALUES  ( 'TableAName1' );

INSERT  INTO TableA

VALUES  ( 'TableAName2' );

INSERT  INTO TableA

VALUES  ( 'TableAName3' );

INSERT  INTO TableA

VALUES  ( 'TableAName4' );

INSERT  INTO TableA

VALUES  ( 'TableAName5' );

GO -- Run the previous command and begins new batch

CREATE TABLE TableB

(

  ID INT IDENTITY

         PRIMARY KEY ,

  Name NVARCHAR(50)

);

GO -- Run the previous command and begins new batch

INSERT  INTO TableB

VALUES  ( 'TableBName1' );

INSERT  INTO TableB

VALUES  ( 'TableBName2' );

INSERT  INTO TableB

VALUES  ( 'TableBName3' );

INSERT  INTO TableB

VALUES  ( 'TableBName4' );

INSERT  INTO TableB

VALUES  ( 'TableBName5' );

GO -- Run the previous command and begins new batch

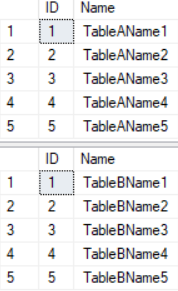
SELECT  \*

FROM    TableA;

SELECT  \*

FROM    TableB;

GO -- Run the previous command and begins new batch



--===========================================================================

--T025\_06\_02

--Turn On the trace flag

DBCC TRACEON(1222, -1);

GO -- Run the previous command and begins new batch



--Check the Trace Status.

DBCC TRACESTATUS(1222, -1);

GO -- Run the previous command and begins new batch

--Return 1 means Trace flag is enabled.

Graphical user interface

Description automatically generated with medium confidence

--===========================================================================

--T025\_06\_03

--Dead Lock Stored Procedure example.

----------------------------------------------------------------------------

--T025\_06\_03\_01

--Transaction1

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spTran1' ) )

    BEGIN

        DROP PROCEDURE spTran1;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spTran1

AS

    BEGIN

        BEGIN TRY

              BEGIN TRAN;

            UPDATE  TableA

            SET     [Name] += ' Tran1'

            WHERE   ID = 1;

                    -- Do something

            WAITFOR DELAY '00:00:4';

            UPDATE  TableB

            SET     [Name] += ' Tran1'

            WHERE   ID = 1;

            COMMIT TRANSACTION;

            PRINT 'spTran1 executed Successful';

        END TRY

        BEGIN CATCH

                    --Check if dead lock exists, ERROR\_NUMBER 1205 is deadlock error flag

            IF ( ERROR\_NUMBER() = 1205 )

                BEGIN

                    PRINT 'ERROR\_NUMBER 1205, Deadlock. Rollback now.';

                END;

                    -- Rollback the transaction

            ROLLBACK;

        END CATCH;

    END;

GO -- Run the previous command and begins new batch

EXECUTE spTran1;

GO -- Run the previous command and begins new batch

----------------------------------------------------------------------------

--T025\_06\_03\_02

--Transaction2

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spTran2' ) )

    BEGIN

        DROP PROCEDURE spTran2;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spTran2

AS

    BEGIN

        BEGIN TRY

              BEGIN TRAN;

            UPDATE  TableB

            SET     [Name] += ' Tran2'

            WHERE   ID = 1;

                    -- Do something

            WAITFOR DELAY '00:00:4';

            UPDATE  TableA

            SET     [Name] += ' Tran2'

            WHERE   ID = 1;

            COMMIT TRANSACTION;

            PRINT 'spTran2 executed Successful';

        END TRY

        BEGIN CATCH

                    --Check if dead lock exists, ERROR\_NUMBER 1205 is deadlock error flag

            IF ( ERROR\_NUMBER() = 1205 )

                BEGIN

                    PRINT 'ERROR\_NUMBER 1205, Deadlock. Rollback now.';

                END;

                    -- Rollback the transaction

            ROLLBACK;

        END CATCH;

    END;

GO -- Run the previous command and begins new batch

EXECUTE spTran2;

GO -- Run the previous command and begins new batch

Graphical user interface, text, application, Word

Description automatically generated

--------------------------

--T025\_06\_03\_03

SELECT  \*

FROM    dbo.TableA

WHERE   ID = 1;

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    dbo.TableB

WHERE   ID = 1;

GO -- Run the previous command and begins new batch

Graphical user interface, application

Description automatically generated

/\*

1.

Logging Dead locks

1.1.

Syntax:

--DBCC Traceon(1222, -1)

Turn On the trace flag

--DBCC TraceStatus(1222, -1)

Check the Trace Status

...Deadlock occur...

--execute sp\_readerrorlog

Read the Error log.

--DBCC Traceoff(1222, -1)

Turn Off the trace flag

...

--EXECUTE sp\_readerrorlog;

To read the error log

1.2.

DBCC means Database Console Command.

SQL Server trace flag 1222 to write the deadlock information

to the SQL Server error log is one of the ways to

track down the queries that are causing deadlocks.

1.3.

-1 parameter means set the flag to global level.

Without -1 parameter means the flag is only valid at the current session level.

--------------------------------------------

2.

--BEGIN

--    BEGIN TRY

--           BEGIN TRAN;

--           --...Do Something...

--           COMMIT TRANSACTION;

--    END TRY

--    BEGIN CATCH

--           --\*\*\*\*

--           --Check if dead lock exists, ERROR\_NUMBER 1205 is deadlock error flag

--        IF ( ERROR\_NUMBER() = 1205 )

--            BEGIN

--                --...Do Something...

--            END;

--    END CATCH;

--END;

--------------------------------------------

3.

Execute Transaction1 first, then go straight to execute Transaction2.

3.1.

Transaction1 will start to update TableA ID=1 record,

so TableA ID=1 are locked by Transaction1.

Transaction2 will start to update TableB ID=1 record,

so TableB ID=1 is locked by Transaction2.

3.2.

Both Transaction1 and Transaction2 has to do something

and wait for a few seconds.

3.3.

Transaction1 will start to update TableB ID=1 record,

but TableB ID=1 is locked by Transaction2 at that moment.

Transaction2 will start to update TableA ID=1 record,

but TableA ID=1 are locked by Transaction1 at that moment.

3.4.

Both transactions have the same default DEADLOCK\_PRIORITY NORMAL.

Both transactions have similar expensive to rollback.

Thus, One of transaction will be chosen the deadlock victim ramdomly.

The other one will be executed successfully.

3.5.

Transaction1 one output:

--(1 row affected)

--(1 row affected)

--spTran1 executed Successful

Transaction2 one output:

--(1 row affected)

--(0 row affected)

--ERROR\_NUMBER 1205, Deadlock. Rollback now.

\*/

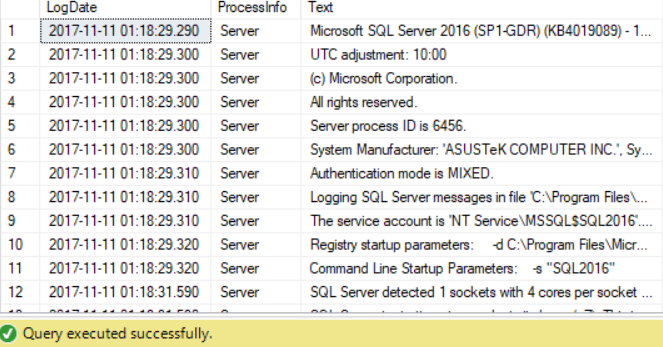
--===========================================================================

--T025\_06\_04

EXECUTE sp\_readerrorlog;

GO -- Run the previous command and begins new batch

--To read the error log



--===========================================================================

--T025\_06\_05

--Turn Off the trace flag

DBCC TRACEOFF(1222, -1);

GO -- Run the previous command and begins new batch



--Check the Trace Status.

DBCC TRACESTATUS(1222, -1);

GO -- Run the previous command and begins new batch

--0 means the trace flag is disabled.



--===========================================================================

--T025\_06\_06

--Clean up

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableA' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableA;

        DROP TABLE TableA;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableB' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableB;

        DROP TABLE TableB;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE TableA

(

  ID INT IDENTITY

         PRIMARY KEY ,

  Name NVARCHAR(50)

);

GO -- Run the previous command and begins new batch

INSERT  INTO TableA

VALUES  ( 'TableAName1' );

INSERT  INTO TableA

VALUES  ( 'TableAName2' );

INSERT  INTO TableA

VALUES  ( 'TableAName3' );

INSERT  INTO TableA

VALUES  ( 'TableAName4' );

INSERT  INTO TableA

VALUES  ( 'TableAName5' );

GO -- Run the previous command and begins new batch

CREATE TABLE TableB

(

  ID INT IDENTITY

         PRIMARY KEY ,

  Name NVARCHAR(50)

);

GO -- Run the previous command and begins new batch

INSERT  INTO TableB

VALUES  ( 'TableBName1' );

INSERT  INTO TableB

VALUES  ( 'TableBName2' );

INSERT  INTO TableB

VALUES  ( 'TableBName3' );

INSERT  INTO TableB

VALUES  ( 'TableBName4' );

INSERT  INTO TableB

VALUES  ( 'TableBName5' );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    TableA;

SELECT  \*

FROM    TableB;

GO -- Run the previous command and begins new batch

Graphical user interface, application, table

Description automatically generated with medium confidence

======================================================================

7. Asp.Net Handling Deadlocks

--=====================================================================

--T025\_07\_AdoDet Handling Deadlocks

--=====================================================================

--=====================================================================

--T025\_07\_01

--Create Sample data

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableA' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableA;

        DROP TABLE TableA;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TableB' ) )

    BEGIN

        TRUNCATE TABLE dbo.TableB;

        DROP TABLE TableB;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE TableA

(

  ID INT IDENTITY

         PRIMARY KEY ,

  Name NVARCHAR(50)

);

GO -- Run the previous command and begins new batch

INSERT  INTO TableA

VALUES  ( 'TableAName1' );

INSERT  INTO TableA

VALUES  ( 'TableAName2' );

INSERT  INTO TableA

VALUES  ( 'TableAName3' );

INSERT  INTO TableA

VALUES  ( 'TableAName4' );

INSERT  INTO TableA

VALUES  ( 'TableAName5' );

GO -- Run the previous command and begins new batch

CREATE TABLE TableB

(

  ID INT IDENTITY

         PRIMARY KEY ,

  Name NVARCHAR(50)

);

GO -- Run the previous command and begins new batch

INSERT  INTO TableB

VALUES  ( 'TableBName1' );

INSERT  INTO TableB

VALUES  ( 'TableBName2' );

INSERT  INTO TableB

VALUES  ( 'TableBName3' );

INSERT  INTO TableB

VALUES  ( 'TableBName4' );

INSERT  INTO TableB

VALUES  ( 'TableBName5' );

GO -- Run the previous command and begins new batch

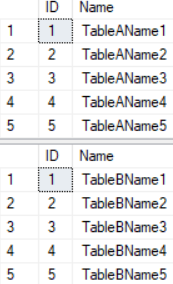
SELECT  \*

FROM    TableA;

SELECT  \*

FROM    TableB;

GO -- Run the previous command and begins new batch



--=====================================================================

--T025\_07\_02

--Transaction1

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spTran1' ) )

    BEGIN

        DROP PROCEDURE spTran1;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spTran1

AS

    BEGIN

        BEGIN TRAN;

        UPDATE  TableA

        SET     [Name] += ' Tran1'

        WHERE   ID = 1;

             -- Do something

        WAITFOR DELAY '00:00:4';

        UPDATE  TableB

        SET     [Name] += ' Tran1'

        WHERE   ID = 1;

        COMMIT TRANSACTION;

    END;

GO -- Run the previous command and begins new batch

--=====================================================================

--T025\_07\_03

--Transaction2 :

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spTran2' ) )

    BEGIN

        DROP PROCEDURE spTran2;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spTran2

AS

    BEGIN

        BEGIN TRAN;

        UPDATE  TableB

        SET     [Name] += ' Tran2'

        WHERE   ID = 1;

             -- Do something

        WAITFOR DELAY '00:00:4';

        UPDATE  TableA

        SET     [Name] += ' Tran2'

        WHERE   ID = 1;

        COMMIT TRANSACTION;

    END;

GO -- Run the previous command and begins new batch

/\*

1.

Logging Dead locks

1.1.

Syntax:

--DBCC Traceon(1222, -1)

Turn On the trace flag

--DBCC TraceStatus(1222, -1)

Check the Trace Status

...Deadlock occur...

--execute sp\_readerrorlog

Read the Error log.

--DBCC Traceoff(1222, -1)

Turn Off the trace flag

...

--EXECUTE sp\_readerrorlog;

To read the error log

1.2.

DBCC means Database Console Command.

SQL Server trace flag 1222 to write the deadlock information

to the SQL Server error log is one of the ways to

track down the queries that are causing deadlocks.

1.3.

-1 parameter means set the flag to global level.

Without -1 parameter means the flag is only valid at the current session level.

-------------------------------------------------

2.

--BEGIN

--    BEGIN TRY

--           BEGIN TRAN;

--           --...Do Something...

--           COMMIT TRANSACTION;

--    END TRY

--    BEGIN CATCH

--           --\*\*\*\*

--           --Check if dead lock exists, ERROR\_NUMBER 1205 is deadlock error flag

--        IF ( ERROR\_NUMBER() = 1205 )

--            BEGIN

--                --...Do Something...

--            END;

--    END CATCH;

--END;

-------------------------------------------

3.

Execute Transaction1 first, then go straight to execute Transaction2.

3.1.

Transaction1 will start to update TableA ID=1 record,

so TableA ID=1 are locked by Transaction1.

Transaction2 will start to update TableB ID=1 record,

so TableB ID=1 is locked by Transaction2.

3.2.

Both Transaction1 and Transaction2 has to do something

and wait for a few seconds.

3.3.

Transaction1 will start to update TableB ID=1 record,

but TableB ID=1 is locked by Transaction2 at that moment.

Transaction2 will start to update TableA ID=1 record,

but TableA ID=1 are locked by Transaction1 at that moment.

3.4.

Both transactions have the same default DEADLOCK\_PRIORITY NORMAL.

Both transactions have similar expensive to rollback.

Thus, One of transaction will be chosen the deadlock victim ramdomly.

The other one will be executed successfully.

3.5.

Transaction1 one output:

--(1 row affected)

--(1 row affected)

--spTran1 executed Successful

Transaction2 one output:

--(1 row affected)

--(0 row affected)

--ERROR\_NUMBER 1205, Deadlock. Rollback now.

\*/

--=====================================================================

--T025\_07\_04

--Check result.

SELECT  \*

FROM    dbo.TableA;

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    dbo.TableB;

GO -- Run the previous command and begins new batch

======================================================================

8. Asp.Net Handling Deadlocks

8.1. Set up SQL Authentication

In SQL server

Object Explorer --> Security --> Logins --> New Logins

-->

General Tab

Login Name :

Tester

Password:

1234

Default Database:

**Sample**

-->

Server Roles Tab

Select

**sysadmin**

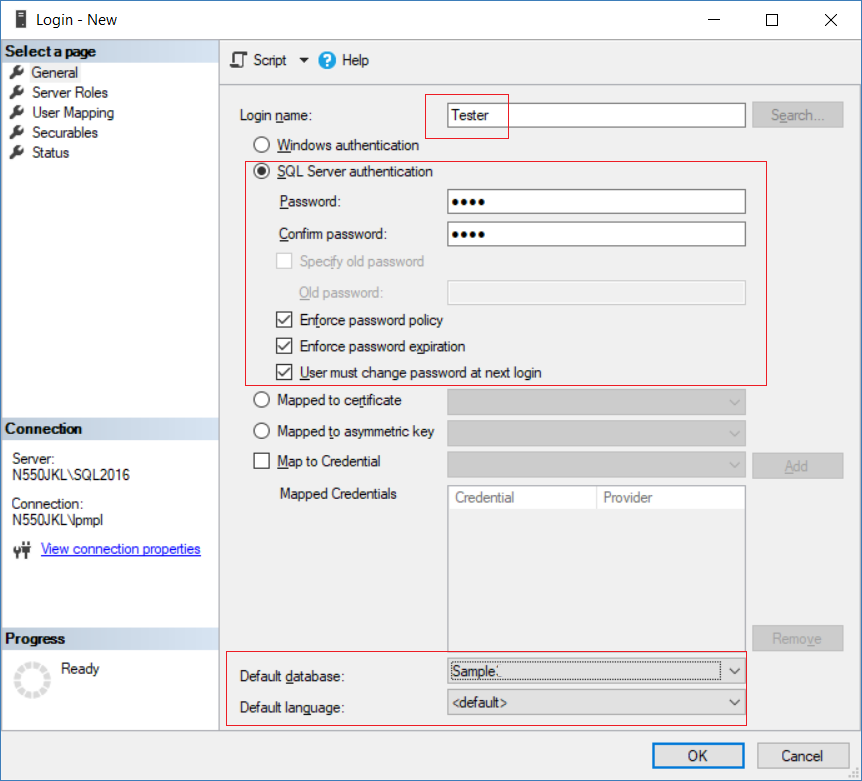
-->

User Mapping Tab

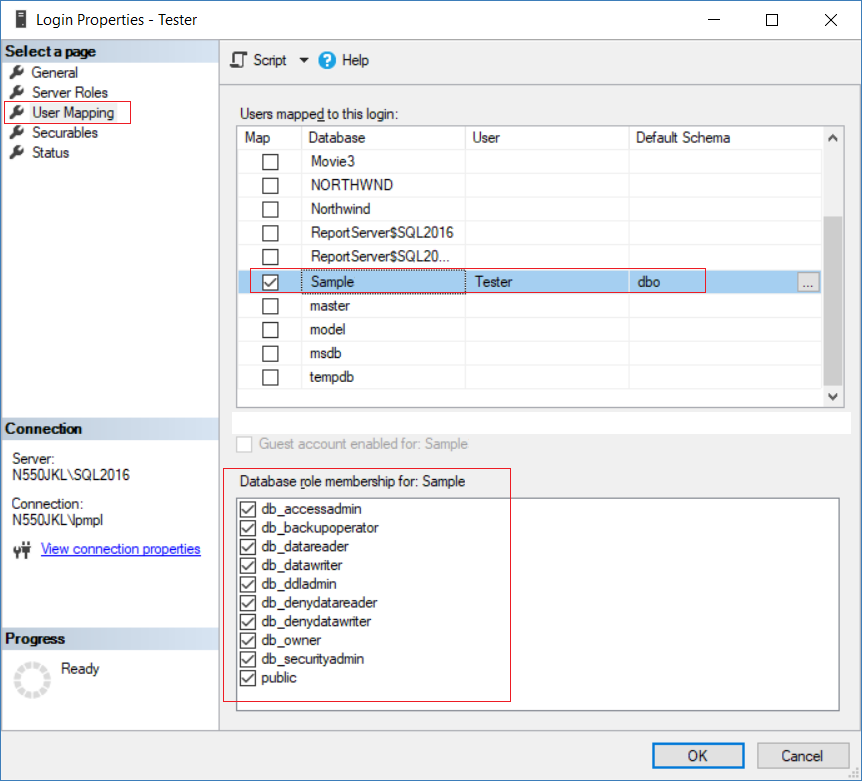
Select **Sample**

Select every Roles.









8.2. Create Web Application

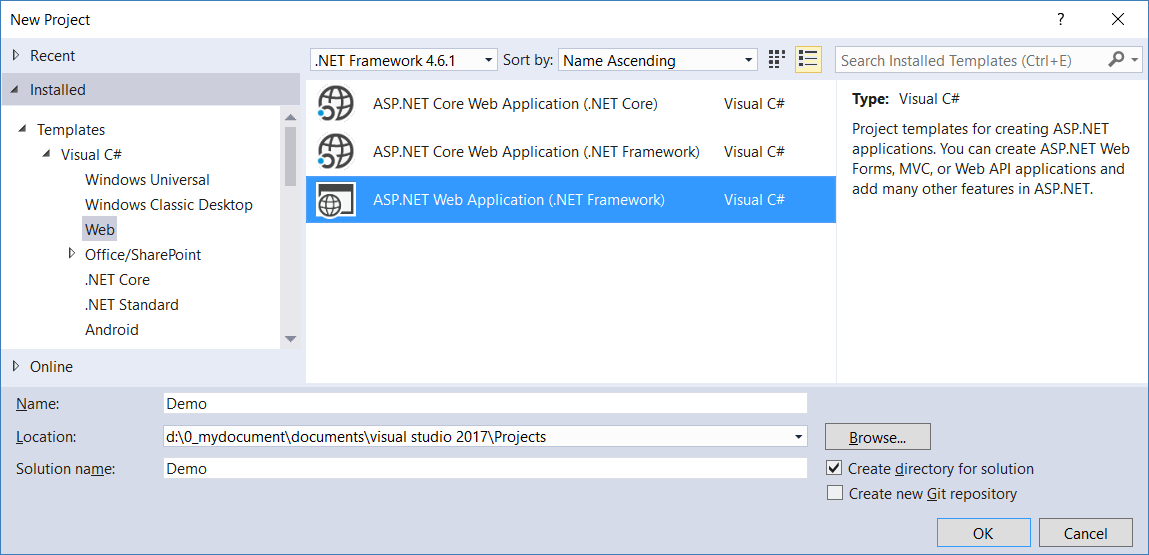
New Project --> Web --> [ASP.NET](http://asp.net/) Web Application (.Net Framework)

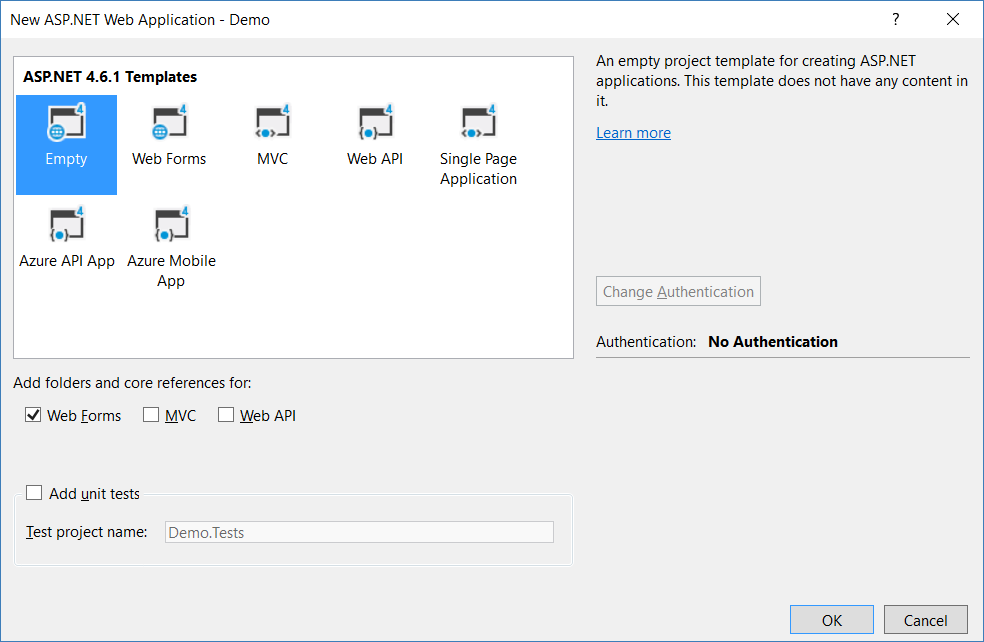
-->

Name:

Demo

--> Web Forms --> OK





8.3. Code

8.3.1. Web.config

Add connection String

<configuration>

  <connectionStrings>

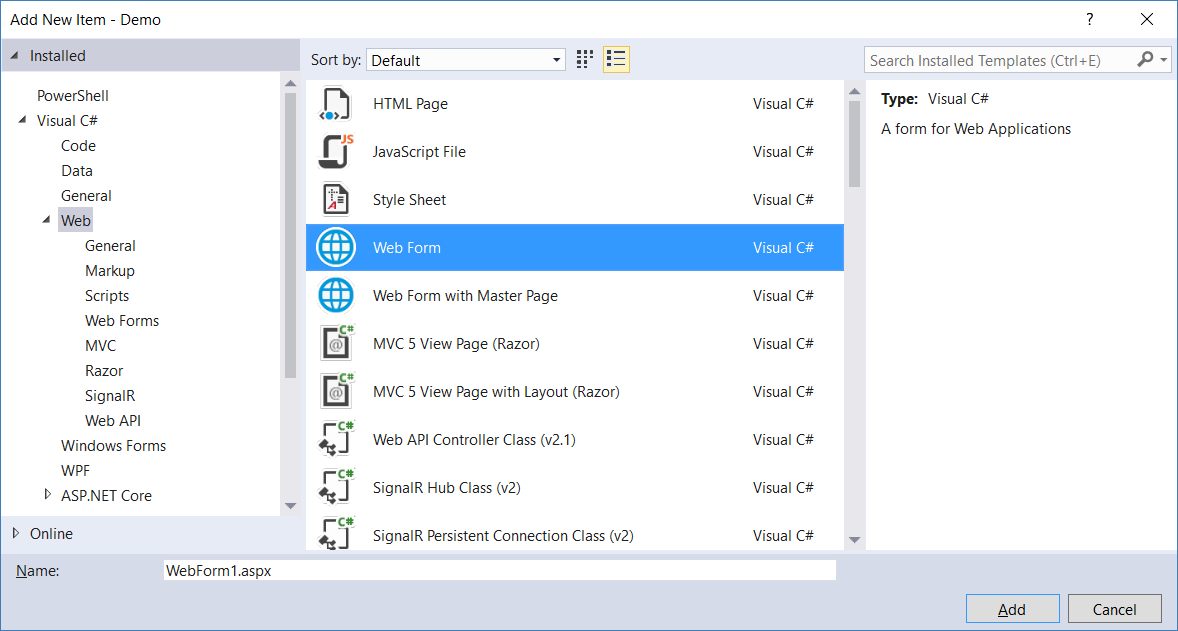
    <add name="SampleConnectionString" connectionString="Data Source=N550JKL\SQL2016;Initial Catalog=Sample;User ID=Tester;Password=1234"

        providerName="System.Data.SqlClient" />

  </connectionStrings>

.....

8.3.2. WebForm1.aspx



<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Demo.WebForm1" %>

<!DOCTYPE html>

<html xmlns="[http://www.w3.org/1999/xhtml">](http://www.w3.org/1999/xhtml%22%3E);

<head runat="server">

    <title></title>

</head>

<body>

    <form id="form1" runat="server">

        <table>

            <tr>

                <td>

                    <asp:Button ID="BtnTran1" runat="server"

                        Text="Update Table A and then Table B"

                        OnClick="BtnTran1\_Click" />

                </td>

            </tr>

            <tr>

                <td>

                    <asp:Label ID="Tran1Label" runat="server"></asp:Label>

                </td>

            </tr>

        </table>

    </form>

</body>

</html>

8.3.3. WebForm1.aspx.cs

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Web.UI;

namespace Demo

{

    public partial class WebForm1 : Page

    {

        protected void Page\_Load(object sender, EventArgs e)

        {

        }

        protected void BtnTran1\_Click(object sender, EventArgs e)

        {

            try

            {

                string cs = ConfigurationManager.ConnectionStrings["SampleConnectionString"].ConnectionString;

                using (var con = new SqlConnection(cs))

                {

                    var cmd = new SqlCommand("spTran1", con);

[cmd.CommandType](http://cmd.commandtype/) = CommandType.StoredProcedure;

                    con.Open();

                    cmd.ExecuteNonQuery();

                    Tran1Label.Text = "spTran1 successful";

                    Tran1Label.ForeColor = Color.Green;

                }

            }

            catch (SqlException ex)

            {

                Tran1Label.Text = ex.Number == 1205 ? "Error Number 1205, Deadlock." : ex.Message;

                Tran1Label.ForeColor = Color.Red;

            }

        }

    }

}

8.3.4. WebForm2.aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs" Inherits="Demo.WebForm2" %>

<!DOCTYPE html>

<html xmlns="[http://www.w3.org/1999/xhtml">](http://www.w3.org/1999/xhtml%22%3E);

<head runat="server">

    <title></title>

</head>

<body>

    <form id="form1" runat="server">

        <table>

            <tr>

                <td>

                    <asp:Button ID="BtnTran2" runat="server"

                        Text="Update Table B and then Table A"

                        OnClick="BtnTran2\_Click" />

                </td>

            </tr>

            <tr>

                <td>

                    <asp:Label ID="Tran2Label" runat="server"></asp:Label>

                </td>

            </tr>

        </table>

    </form>

</body>

</html>

8.3.5. WebForm2.aspx.cs

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Web.UI;

namespace Demo

{

    public partial class WebForm2 : Page

    {

        protected void Page\_Load(object sender, EventArgs e)

        {

        }

        protected void BtnTran2\_Click(object sender, EventArgs e)

        {

            try

            {

                string cs = ConfigurationManager.ConnectionStrings["SampleConnectionString"].ConnectionString;

                using (var con = new SqlConnection(cs))

                {

                    var cmd = new SqlCommand("spTran2", con);

[cmd.CommandType](http://cmd.commandtype/) = CommandType.StoredProcedure;

                    con.Open();

                    cmd.ExecuteNonQuery();

                    Tran2Label.Text = "spTran2 successful";

                    Tran2Label.ForeColor = Color.Green;

                }

            }

            catch (SqlException ex)

            {

                Tran2Label.Text = ex.Number == 1205 ? "Error Number 1205, Deadlock." : ex.Message;

                Tran2Label.ForeColor = Color.Red;

            }

        }

    }

}

Graphical user interface, text, application, chat or text message, email

Description automatically generated