(T25)處理 DeadLock(死鎖)

CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc

==

(T25)處理 DeadLock(死鎖)

==

==

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.

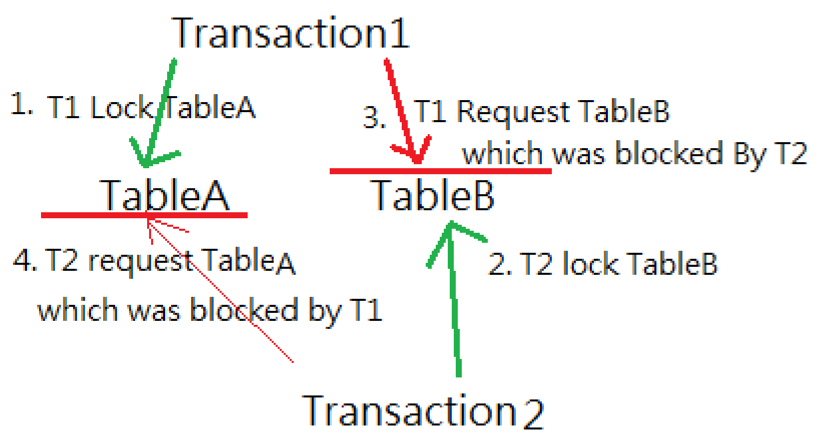
2.
Logging Dead locks
2.1.
Syntax:
--DBCC Traceon(1222, -1)
Turn On the trace flag
--DBCC TraceStatus(1222, -1)
Check the Trace Status
...Deadlock occur...
--execute sp_readerrorlog
Read the Error log.
--DBCC Traceoff(1222, -1)
Turn Off the trace flag
...
--EXECUTE sp_readerrorlog;
To read the error 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;
```

==

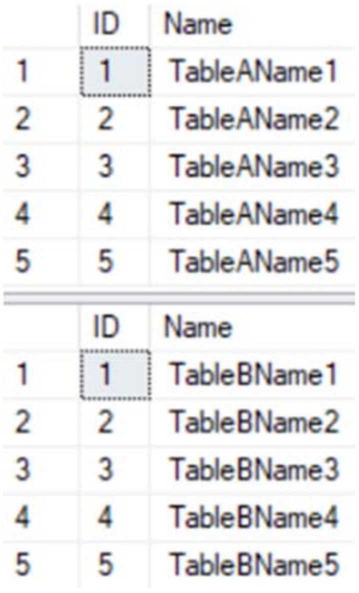
1. Deadlock Example


Transaction1
1. T1 Lock TableA
3. T1 Request TableB which was blocked By T2
TableA
TableB
4. T2 request TableA which was blocked by T1
2. T2 lock TableB
Transaction 2

--==

```
--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
```

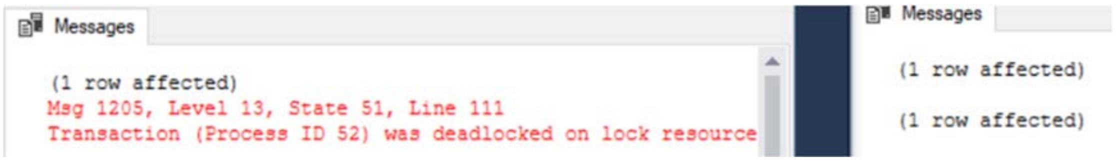


	ID	Name
1	1	TableAName1
2	2	TableAName2
3	3	TableAName3
4	4	TableAName4
5	5	TableAName5

	ID	Name
1	1	TableBName1
2	2	TableBName2
3	3	TableBName3
4	4	TableBName4
5	5	TableBName5

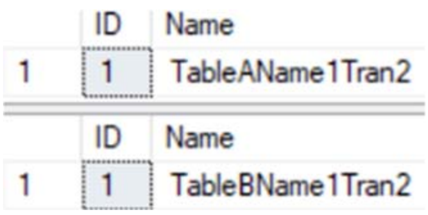
```
--======================================================================
--T025_01_02
--Dead Lock Example
----------------------------------------------------------------------
--T025_01_02_01
-- Transaction1
BEGIN TRAN;
UPDATE  TableA
SET     [Name] += ' Tran1'
WHERE   ID = 1;
-- Do something
WAITFOR 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;
```



```
Messages

(1 row affected)
Msg 1205, Level 13, State 51, Line 111
Transaction (Process ID 52) was deadlocked on lock resource
```

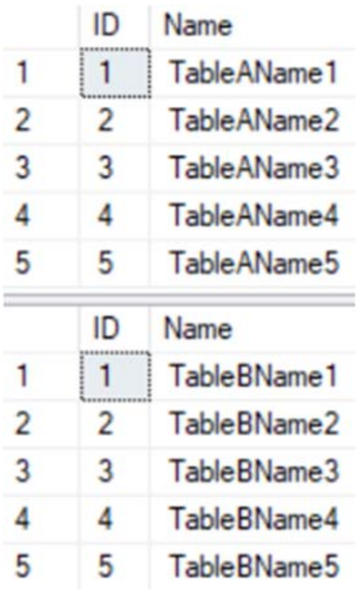
```
Messages

(1 row affected)

(1 row affected)
```



	ID	Name
1	1	TableAName1Tran2

	ID	Name
1	1	TableBName1Tran2

```
/*
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
```



	ID	Name
1	1	TableAName1
2	2	TableAName2
3	3	TableAName3
4	4	TableAName4
5	5	TableAName5

	ID	Name
1	1	TableBName1
2	2	TableBName2
3	3	TableBName3
4	4	TableBName4
5	5	TableBName5

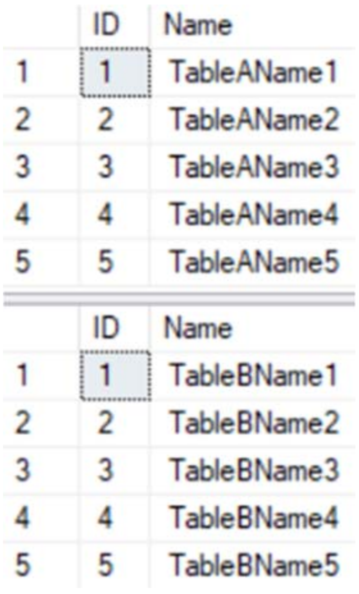
==

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
```

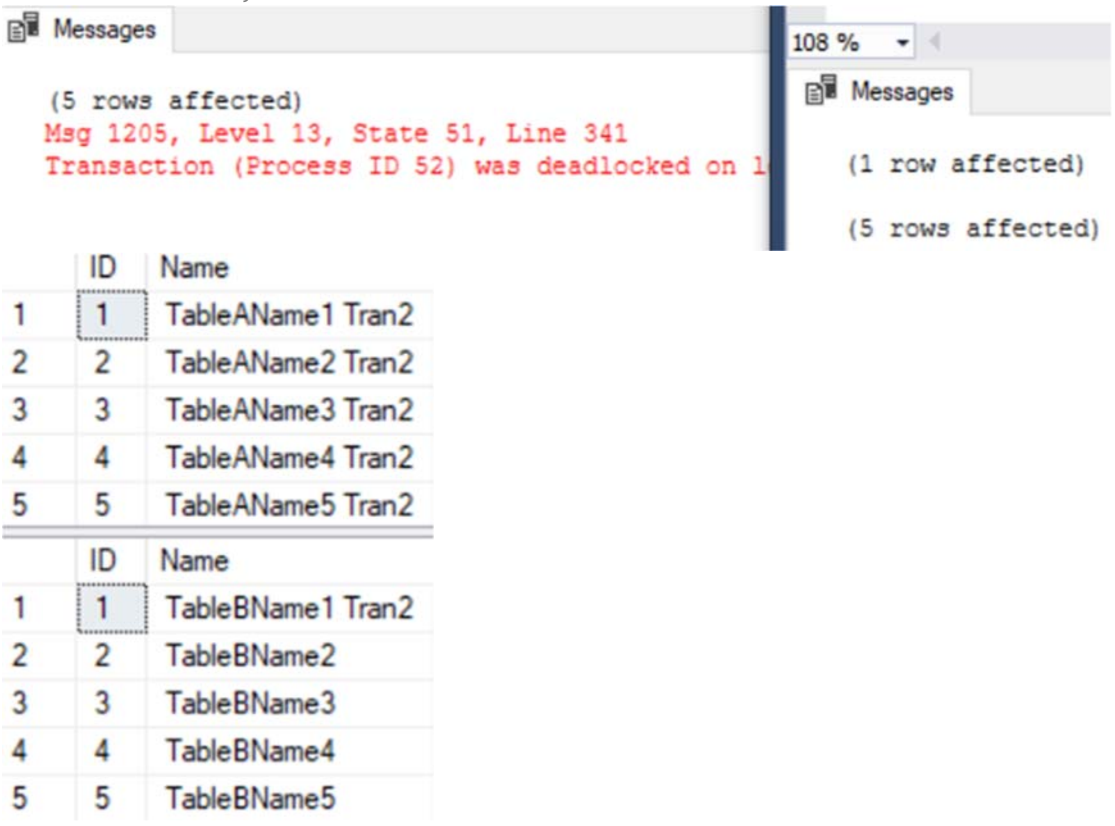


	ID	Name
1	1	TableAName1
2	2	TableAName2
3	3	TableAName3
4	4	TableAName4
5	5	TableAName5

	ID	Name
1	1	TableBName1
2	2	TableBName2
3	3	TableBName3
4	4	TableBName4
5	5	TableBName5

```
--============================================================================
--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;
```



```
(5 rows affected)
Msg 1205, Level 13, State 51, Line 341
Transaction (Process ID 52) was deadlocked on l
```

```
(1 row affected)

(5 rows affected)
```

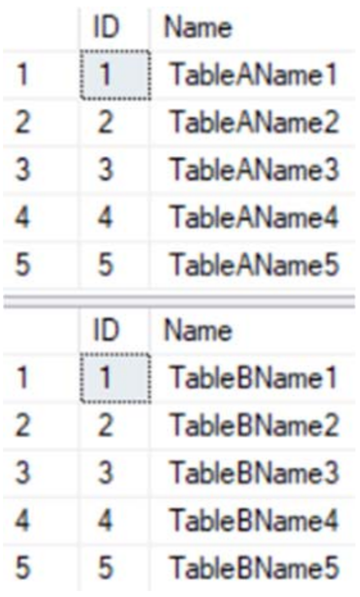
	ID	Name
1	1	TableAName1 Tran2
2	2	TableAName2 Tran2
3	3	TableAName3 Tran2
4	4	TableAName4 Tran2
5	5	TableAName5 Tran2

	ID	Name
1	1	TableBName1 Tran2
2	2	TableBName2
3	3	TableBName3
4	4	TableBName4
5	5	TableBName5

```
/*
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    *
```

```sql
                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
```



	ID	Name
1	1	TableAName1
2	2	TableAName2
3	3	TableAName3
4	4	TableAName4
5	5	TableAName5

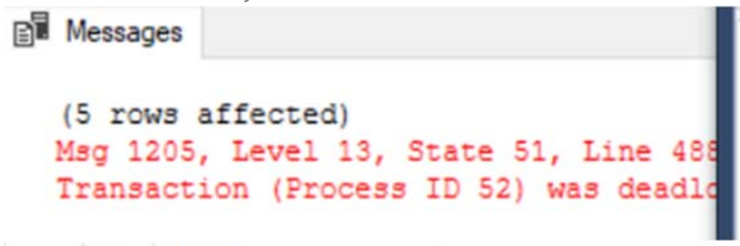
	ID	Name
1	1	TableBName1
2	2	TableBName2
3	3	TableBName3
4	4	TableBName4
5	5	TableBName5

===

3. Deadlock Priority : different DEADLOCK_PRIORITY

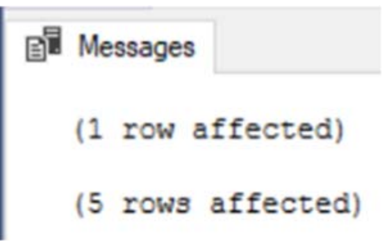
```
--=====================================================================
--T025_03_Deadlock Priority : different DEADLOCK_PRIORITY
--=====================================================================
/*
--SET DEADLOCK_PRIORITY HIGH;
if both transaction has the different DEADLOCK_PRIORITY,
the transaction 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;
```



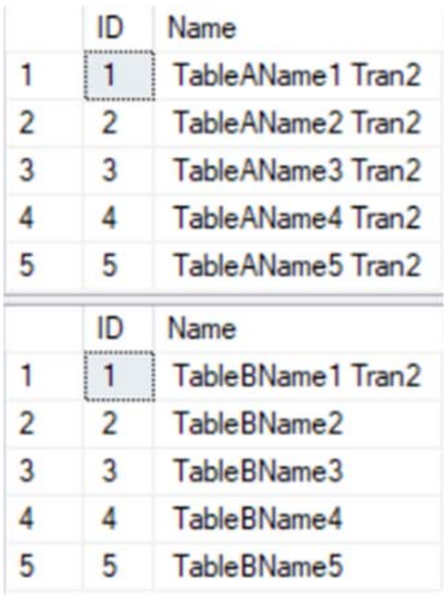
Messages

```
(5 rows affected)
Msg 1205, Level 13, State 51, Line 48
Transaction (Process ID 52) was deadlo
```



Messages

```
(1 row affected)

(5 rows affected)
```



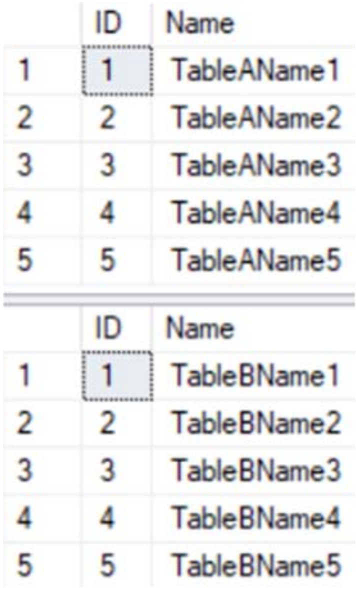
	ID	Name
1	1	TableAName1 Tran2
2	2	TableAName2 Tran2
3	3	TableAName3 Tran2
4	4	TableAName4 Tran2
5	5	TableAName5 Tran2

	ID	Name
1	1	TableBName1 Tran2
2	2	TableBName2
3	3	TableBName3
4	4	TableBName4
5	5	TableBName5

```
/*
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
```



	ID	Name
1	1	TableAName1
2	2	TableAName2
3	3	TableAName3
4	4	TableAName4
5	5	TableAName5

	ID	Name
1	1	TableBName1
2	2	TableBName2
3	3	TableBName3
4	4	TableBName4
5	5	TableBName5

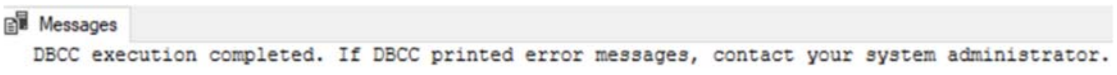
4. Deadlock Analysis And Prevention

```
--======================================================================
--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.
*/
```

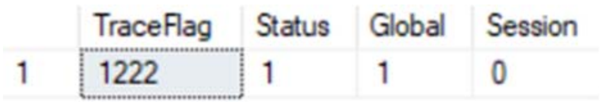
4.1. Logging Dead locks

```
--======================================================================
--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
```



```
Messages
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
```

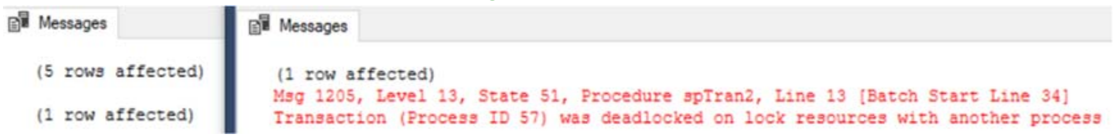
```
--------------------------
--T025_04_01_02
--Check the Trace Status.
DBCC TRACESTATUS(1222, -1);
GO -- Run the previous command and begins new batch
```



	TraceFlag	Status	Global	Session
1	1222	1	1	0

```
/*
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.
```

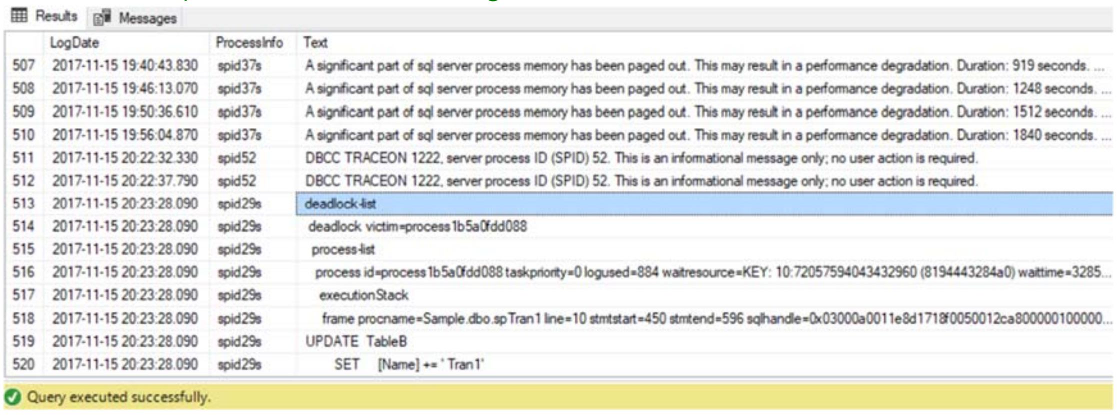
```
*/

--------------------------
--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
```



```
Messages
(5 rows affected)

(1 row affected)
```

```
Messages
(1 row affected)
Msg 1205, Level 13, State 51, Procedure spTran2, Line 13 [Batch Start Line 34]
Transaction (Process ID 57) was deadlocked on lock resources with another process
```

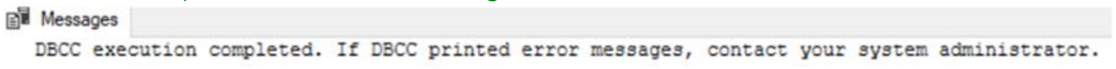
```
--------------------------
--T025_04_01_05
--To read the error log
EXECUTE sp_readerrorlog;
GO -- Run the previous command and begins new batch
```



	LogDate	ProcessInfo	Text
507	2017-11-15 19:40:43.830	spid37s	A significant part of sql server process memory has been paged out. This may result in a performance degradation. Duration: 919 seconds. ...
508	2017-11-15 19:46:13.070	spid37s	A significant part of sql server process memory has been paged out. This may result in a performance degradation. Duration: 1248 seconds. ...
509	2017-11-15 19:50:36.610	spid37s	A significant part of sql server process memory has been paged out. This may result in a performance degradation. Duration: 1512 seconds. ...
510	2017-11-15 19:56:04.870	spid37s	A significant part of sql server process memory has been paged out. This may result in a performance degradation. Duration: 1840 seconds. ...
511	2017-11-15 20:22:32.330	spid52	DBCC TRACEON 1222, server process ID (SPID) 52. This is an informational message only; no user action is required.
512	2017-11-15 20:22:37.790	spid52	DBCC TRACEON 1222, server process ID (SPID) 52. This is an informational message only; no user action is required.
513	2017-11-15 20:23:28.090	spid29s	deadlock-list
514	2017-11-15 20:23:28.090	spid29s	deadlock victim=process1b5a0fdd088
515	2017-11-15 20:23:28.090	spid29s	process-list
516	2017-11-15 20:23:28.090	spid29s	process id=process1b5a0fdd088 taskpriority=0 logused=884 waitresource=KEY: 10:72057594043432960 (8194443284a0) waittime=3285...
517	2017-11-15 20:23:28.090	spid29s	executionStack
518	2017-11-15 20:23:28.090	spid29s	frame procname=Sample.dbo.spTran1 line=10 stmtstart=450 stmtend=596 sqlhandle=0x03000a0011e8d1718f0050012ca800000100000...
519	2017-11-15 20:23:28.090	spid29s	UPDATE TableB
520	2017-11-15 20:23:28.090	spid29s	SET [Name] += ' Tran1'

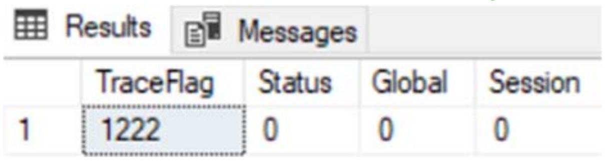
Query executed successfully.

```
--------------------------
--T025_04_01_06
--Turn Off the trace flag
DBCC TRACEOFF(1222, -1);
GO -- Run the previous command and begins new batch
```



```
Messages
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
```

```
--------------------------
--T025_04_01_06
--Check the Trace Status.
DBCC TRACESTATUS(1222, -1);
GO -- Run the previous command and begins new batch
```



	TraceFlag	Status	Global	Session
1	1222	0	0	0

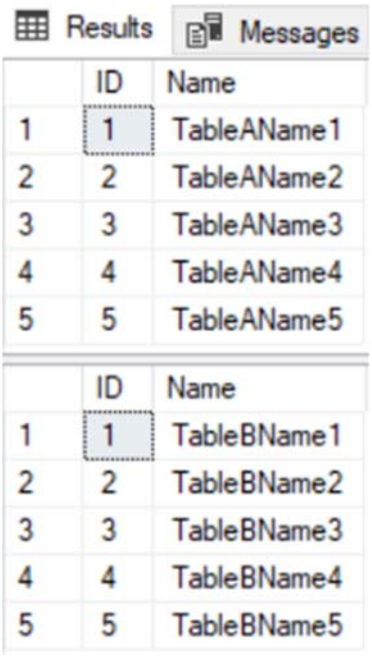
```
/*
1.
Logging Dead locks
1.1.
--DBCC Traceon(1222, -1)
Turn On the trace flag
--DBCC TraceStatus(1222, -1)
Check the Trace Status
...Deadlock occur...
--execute sp_readerrorlog
Read the Error log.
--DBCC Traceoff(1222, -1)
```

```
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;
```

```sql
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
```



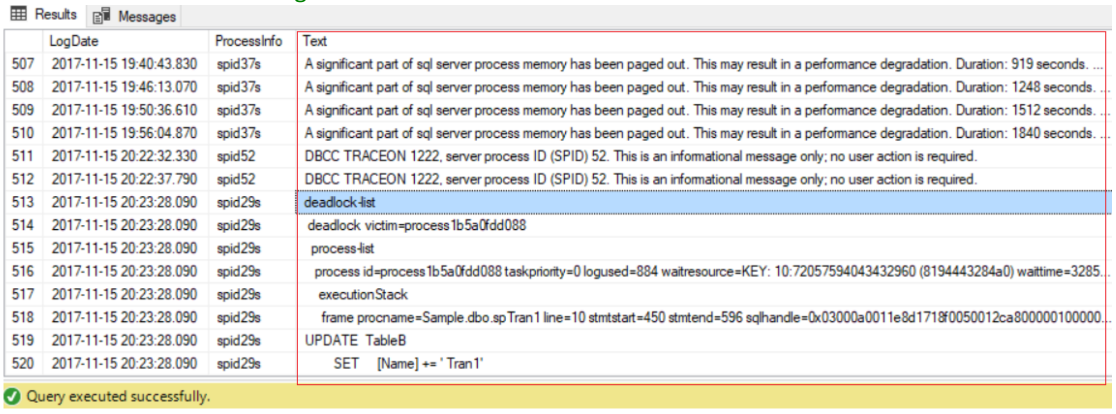
Results | Messages

	ID	Name
1	1	TableAName1
2	2	TableAName2
3	3	TableAName3
4	4	TableAName4
5	5	TableAName5

	ID	Name
1	1	TableBName1
2	2	TableBName2
3	3	TableBName3
4	4	TableBName4
5	5	TableBName5

4.3. Deadlock Analysis And Prevention

```
--======================================================================
--T025_04_03
--Deadlock Analysis And Prevention
EXECUTE sp_readerrorlog;
--To read the error log
```

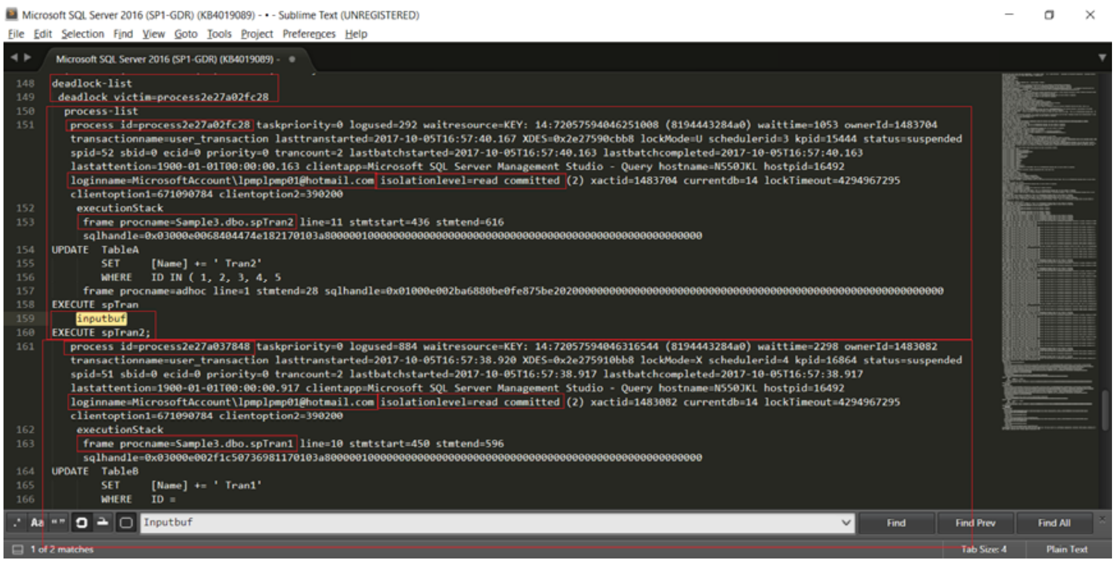


Results | Messages

	LogDate	ProcessInfo	Text
507	2017-11-15 19:40:43.830	spid37s	A significant part of sql server process memory has been paged out. This may result in a performance degradation. Duration: 919 seconds. ...
508	2017-11-15 19:46:13.070	spid37s	A significant part of sql server process memory has been paged out. This may result in a performance degradation. Duration: 1248 seconds. ...
509	2017-11-15 19:50:36.610	spid37s	A significant part of sql server process memory has been paged out. This may result in a performance degradation. Duration: 1512 seconds. ...
510	2017-11-15 19:56:04.870	spid37s	A significant part of sql server process memory has been paged out. This may result in a performance degradation. Duration: 1840 seconds. ...
511	2017-11-15 20:22:32.330	spid52	DBCC TRACEON 1222, server process ID (SPID) 52. This is an informational message only; no user action is required.
512	2017-11-15 20:22:37.790	spid52	DBCC TRACEON 1222, server process ID (SPID) 52. This is an informational message only; no user action is required.
513	2017-11-15 20:23:28.090	spid29s	deadlock-list
514	2017-11-15 20:23:28.090	spid29s	deadlock victim=process1b5a0fdd088
515	2017-11-15 20:23:28.090	spid29s	process-list
516	2017-11-15 20:23:28.090	spid29s	process id=process1b5a0fdd088 taskpriority=0 logused=884 waitresource=KEY: 10:72057594043432960 (8194443284a0) waittime=3285...
517	2017-11-15 20:23:28.090	spid29s	executionStack
518	2017-11-15 20:23:28.090	spid29s	frame procname=Sample.dbo.spTran1 line=10 stmtstart=450 stmtend=596 sqlhandle=0x03000a0011e8d1718f0050012ca800000100000...
519	2017-11-15 20:23:28.090	spid29s	UPDATE TableB
520	2017-11-15 20:23:28.090	spid29s	SET [Name] += ' Tran1'

Query executed successfully.

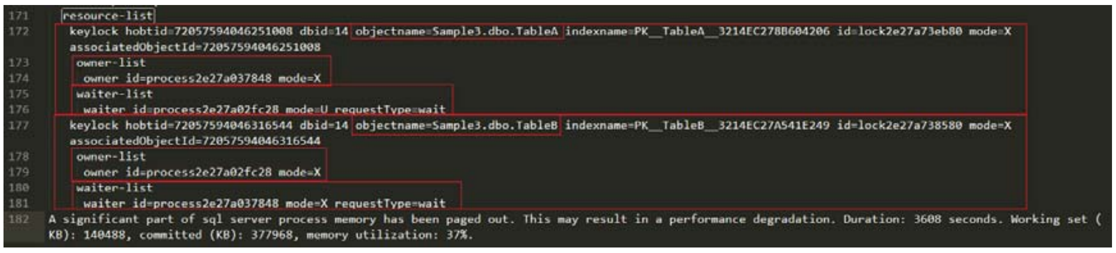
```
/*
1.
--To read the error log
execute sp_readerrorlog
1.1.
Then copy the Text column into sublime or Notepad++
1.2.
There are 3 important sections, deadlock victim, process-list, and resource-list
in deadlock information from the Text of sp_readerrorlog.
-----------
1.2.1.
deadlock victim :
deadlock 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.
-----------
```


Microsoft SQL Server 2016 (SP1-GDR) (KB4019089) - • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Microsoft SQL Server 2016 (SP1-GDR) (KB4019089) -
Find
Find Prev
Find All
1 of 2 matches
Tab Size: 4
Plain Text

```
deadlock-list
 deadlock victim=process2e27a02fc28
  process-list
   process id=process2e27a02fc28 taskpriority=0 logused=292 waitresource=KEY: 14:72057594046251008 (8194443284a0) waittime=1053 ownerId=1483704
   transactionname=user_transaction lasttranstarted=2017-10-05T16:57:40.167 XDES=0x2e27590cbb8 lockMode=U schedulerid=3 kpid=15444 status=suspended
   spid=52 sbid=0 ecid=0 priority=0 trancount=2 lastbatchstarted=2017-10-05T16:57:40.163 lastbatchcompleted=2017-10-05T16:57:40.163
   lastattention=1900-01-01T00:00:00.163 clientapp=Microsoft SQL Server Management Studio - Query hostname=N550JKL hostpid=16492
   loginname=MicrosoftAccount\lpmplpmp01@hotmail.com isolationlevel=read committed (2) xactid=1483704 currentdb=14 lockTimeout=4294967295
   clientoption1=671090784 clientoption2=390200
    executionStack
     frame procname=Sample3.dbo.spTran2 line=11 stmtstart=436 stmtend=616
     sqlhandle=0x03000e0068404474e182170103a8000001000000000000000000000000000000000000000000000000000000
UPDATE  TableA
        SET     [Name] += ' Tran2'
        WHERE   ID IN ( 1, 2, 3, 4, 5
     frame procname=adhoc line=1 stmtend=28 sqlhandle=0x01000e002ba6880be0fe875be202000000000000000000000000000000000000000000000000000000000000
EXECUTE spTran
    inputbuf
EXECUTE spTran2;
   process id=process2e27a037848 taskpriority=0 logused=884 waitresource=KEY: 14:72057594046316544 (8194443284a0) waittime=2298 ownerId=1483082
   transactionname=user_transaction lasttranstarted=2017-10-05T16:57:38.920 XDES=0x2e275910bb8 lockMode=X schedulerid=4 kpid=16864 status=suspended
   spid=51 sbid=0 ecid=0 priority=0 trancount=2 lastbatchstarted=2017-10-05T16:57:38.917 lastbatchcompleted=2017-10-05T16:57:38.917
   lastattention=1900-01-01T00:00:00.917 clientapp=Microsoft SQL Server Management Studio - Query hostname=N550JKL hostpid=16492
   loginname=MicrosoftAccount\lpmplpmp01@hotmail.com isolationlevel=read committed (2) xactid=1483082 currentdb=14 lockTimeout=4294967295
   clientoption1=671090784 clientoption2=390200
    executionStack
     frame procname=Sample3.dbo.spTran1 line=10 stmtstart=450 stmtend=596
     sqlhandle=0x03000e002f1c50736981170103a8000001000000000000000000000000000000000000000000000000000000
UPDATE  TableB
        SET     [Name] += ' Tran1'
        WHERE   ID =
```

```
Inputbuf
```

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



```
  resource-list
   keylock hobtid=72057594046251008 dbid=14 objectname=Sample3.dbo.TableA indexname=PK__TableA__3214EC278B604206 id=lock2e27a73eb80 mode=X
   associatedObjectId=72057594046251008
    owner-list
     owner id=process2e27a037848 mode=X
    waiter-list
     waiter id=process2e27a02fc28 mode=U requestType=wait
   keylock hobtid=72057594046316544 dbid=14 objectname=Sample3.dbo.TableB indexname=PK__TableB__3214EC27A541E249 id=lock2e27a738580 mode=X
   associatedObjectId=72057594046316544
    owner-list
     owner id=process2e27a02fc28 mode=X
    waiter-list
     waiter id=process2e27a037848 mode=X requestType=wait
A significant part of sql server process memory has been paged out. This may result in a performance degradation. Duration: 3608 seconds. Working set (
KB): 140488, committed (KB): 377968, memory utilization: 37%.
```

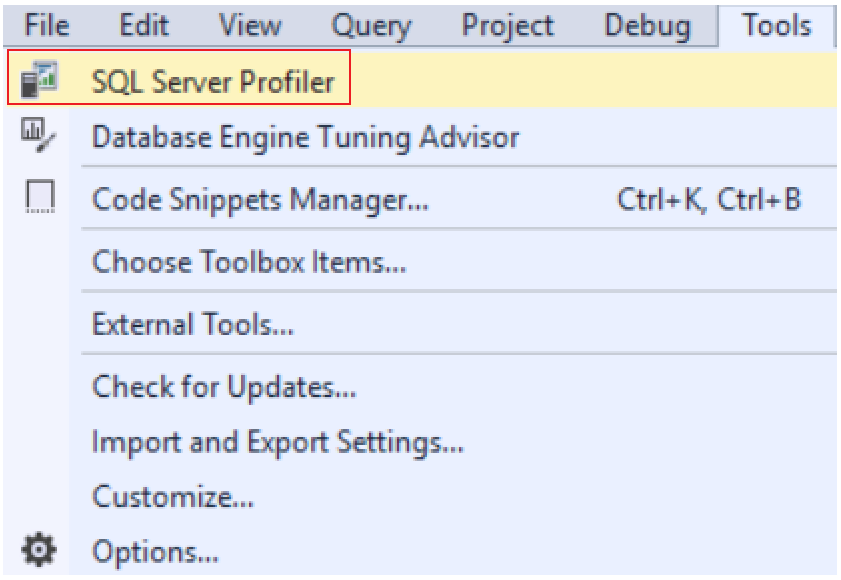
1.2.3.
resource-list :
resource-list contains the list of Database Objects resource of the process
which participate the deadlock.
There are some important sections regarding deadlock.
1.2.3.1.
objectname
objectname is the Database Objects resource name of the process which participate the deadlock.
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

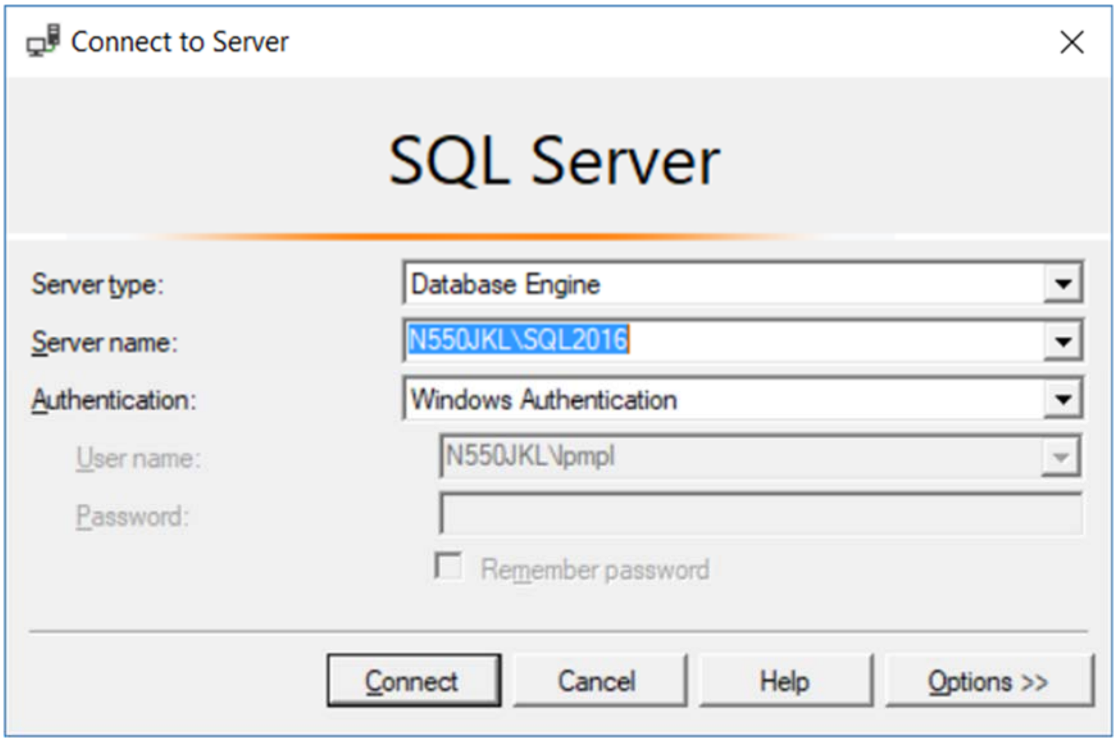
```
S means Shared lock
U means update lock
X means exclusive lock
...ect.
1.2.3.3.
waiter-list
--waiter-list
--   waiter id=process2e27a02fc28 mode=U requestType=wait
waiter-list contains the "owning process id", the "owning process lock mode", and the "requestType"
目前是哪個 process 正在等待這個 object
1.2.3.3.1.
"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
*/
```




Connect to Server

SQL Server

Server type: Database Engine

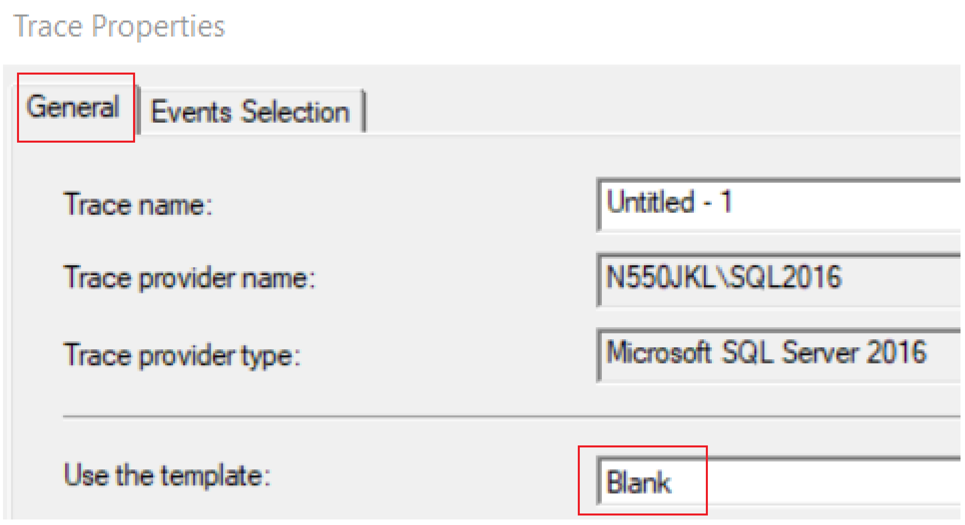
Server name: N550JKL\SQL2016

Authentication: Windows Authentication

User name: N550JKL\pmpl

Password:

Remember password

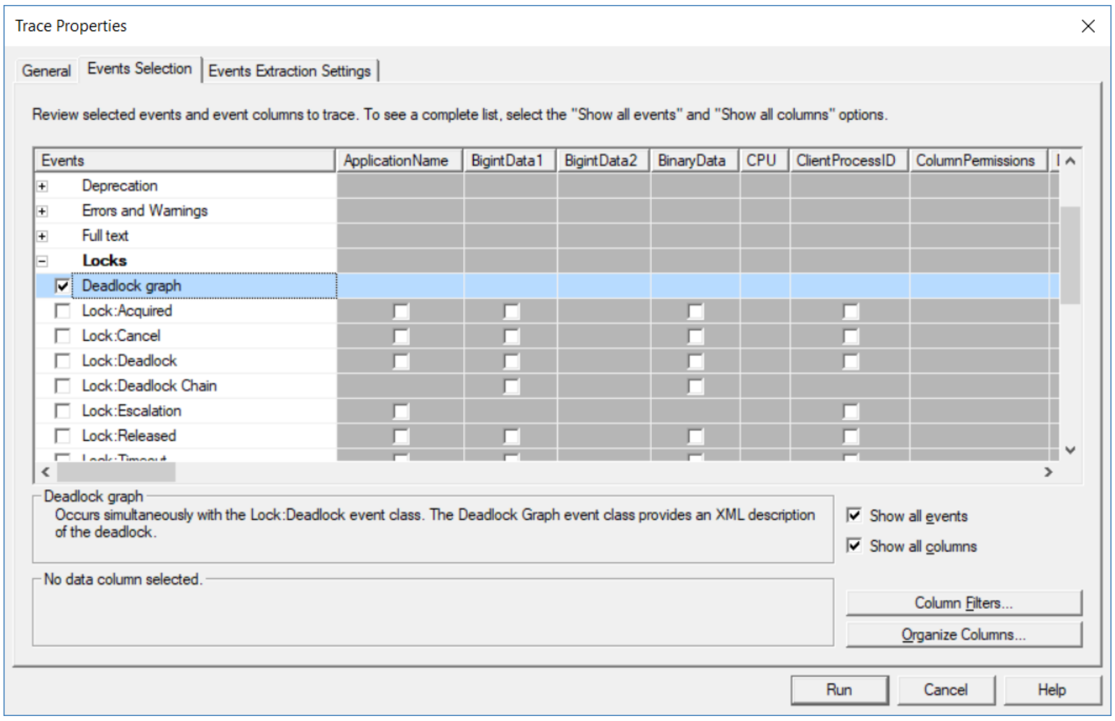

Trace Properties

Trace name: Untitled - 1

Trace provider name: N550JKL\SQL2016

Trace provider type: Microsoft SQL Server 2016

Use the template: Blank



Trace Properties

General | Events Selection | Events Extraction Settings

Review selected events and event columns to trace. To see a complete list, select the "Show all events" and "Show all columns" options.

Events	ApplicationName	BigintData1	BigintData2	BinaryData	CPU	ClientProcessID	ColumnPermissions
+ Deprecation							
+ Errors and Warnings							
+ Full text							
− **Locks**							
☑ Deadlock graph							
☐ Lock:Acquired	☐	☐		☐		☐	
☐ Lock:Cancel	☐	☐		☐		☐	
☐ Lock:Deadlock	☐	☐		☐		☐	
☐ Lock:Deadlock Chain		☐		☐			
☐ Lock:Escalation	☐					☐	
☐ Lock:Released	☐	☐		☐		☐	

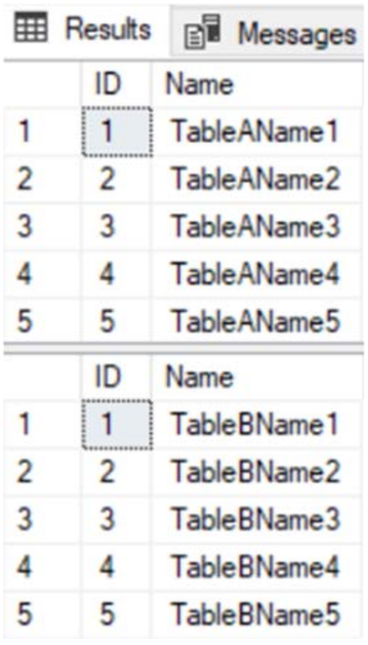
Deadlock graph
Occurs simultaneously with the Lock:Deadlock event class. The Deadlock Graph event class provides an XML description of the deadlock.

No data column selected.

☑ Show all events
☑ Show all columns

```
--======================================================================
--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
```

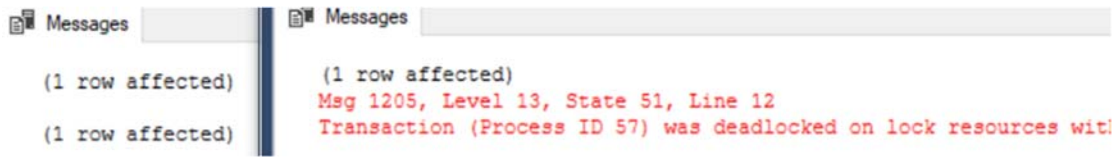


	ID	Name
1	1	TableAName1
2	2	TableAName2
3	3	TableAName3
4	4	TableAName4
5	5	TableAName5

	ID	Name
1	1	TableBName1
2	2	TableBName2
3	3	TableBName3
4	4	TableBName4
5	5	TableBName5

```
--=======================================================================
--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
```



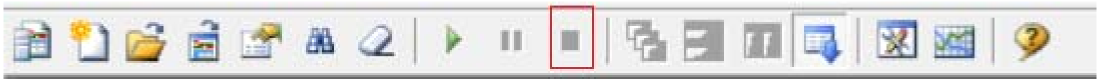
```
Messages

   (1 row affected)

   (1 row affected)
```

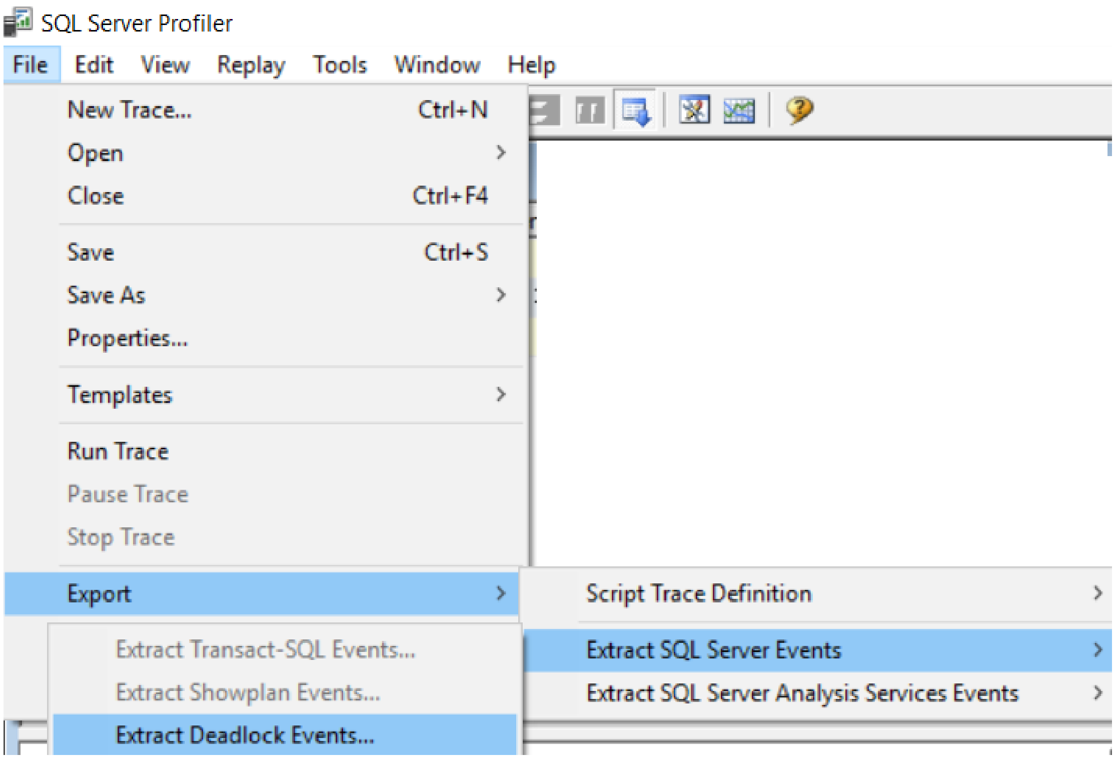
```
Messages

 (1 row affected)
Msg 1205, Level 13, State 51, Line 12
Transaction (Process ID 57) was deadlocked on lock resources wit
```

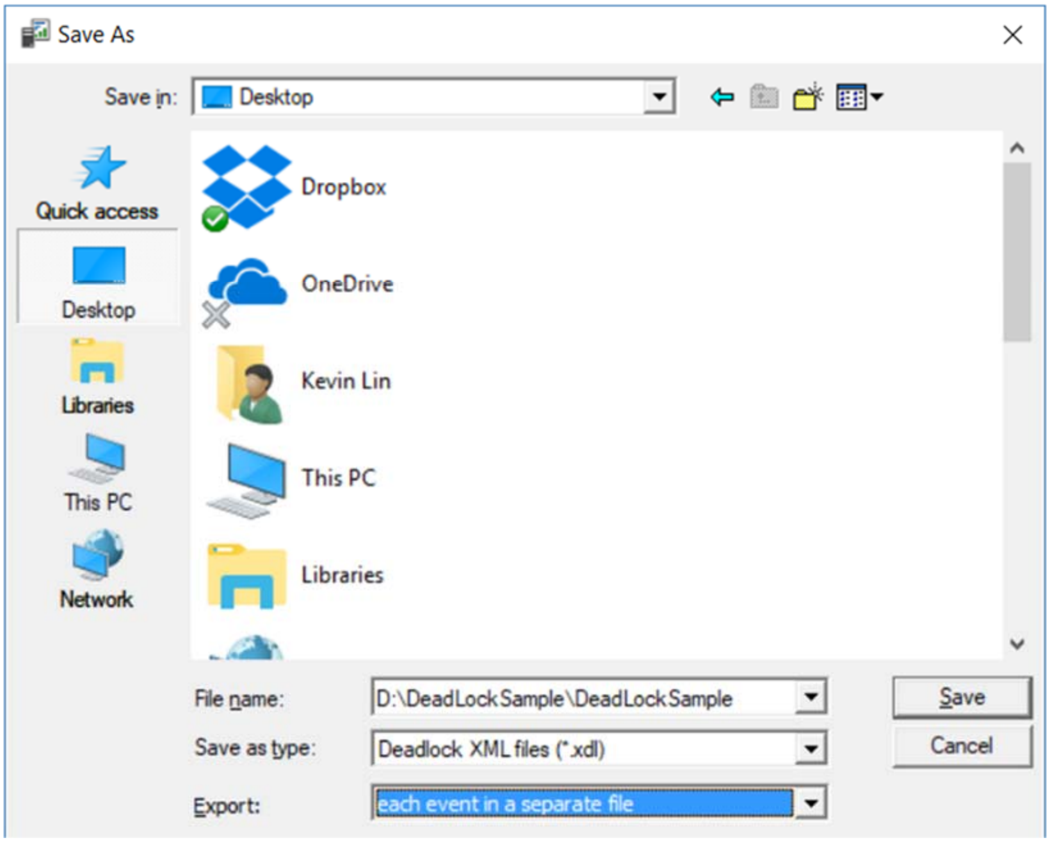
```
/*
1.
Create the sample data first, then Open another Query for Transaction2
Execute Transaction1 first, then in the mean time, execute Transaction2.
1.1.
Transaction1 will start to update TableA ID=1 record,
so TableA ID=1 is locked by Transaction1.
Transaction2 will start to update TableB ID=1 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.
```

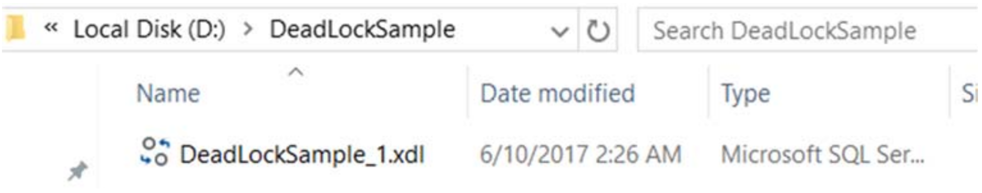
```
--SELECT  OBJECT_NAME([object_id])
--FROM    sys.partitions
--WHERE   hobt_id = 72057594046644224;
Use OBJECT_NAME([object_id]) to find out the database object name
involved in the deadlock.
In this case that will return TableA
*/
```



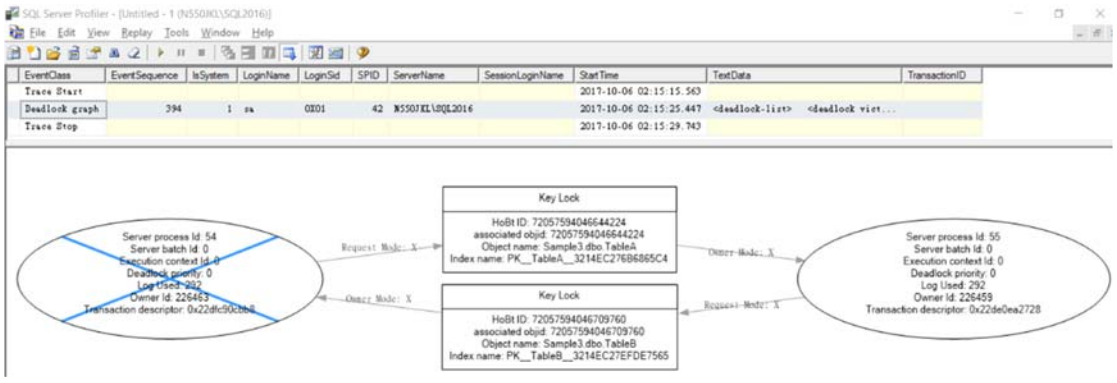




File name: D:\DeadLockSample\DeadLockSample
Save as type: Deadlock XML files (*.xdl)
Export: each event in a separate file

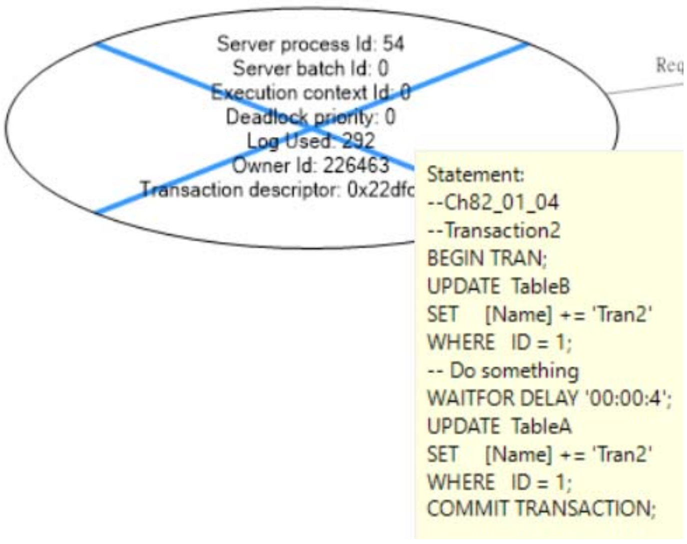


Name	Date modified	Type	Si
DeadLockSample_1.xdl	6/10/2017 2:26 AM	Microsoft SQL Ser...	


SQL Server Profiler - [Untitled - 1 (N550JKL\SQL2016)]
File Edit View Replay Tools Window Help

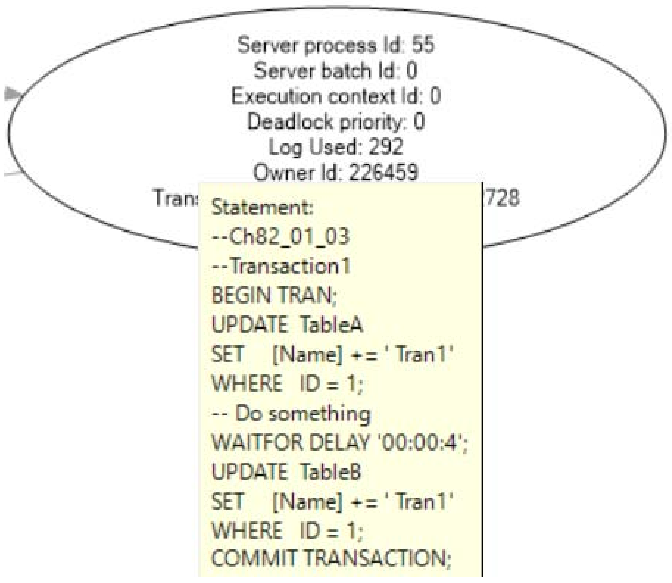
	EventClass	EventSequence	IsSystem	LoginName	LoginSid	SPID	ServerName	SessionLoginName	StartTime	TextData	TransactionID
	Trace Start								2017-10-06 02:15:15.563		
	Deadlock graph	394	1	sa	0X01	42	N550JKL\SQL2016		2017-10-06 02:15:25.447	`<deadlock-list>  <deadlock vict...`	
	Trace Stop								2017-10-06 02:15:29.743		

Server process Id: 54
Server batch Id: 0
Execution context Id: 0
Deadlock priority: 0
Log Used: 292
Owner Id: 226463
Transaction descriptor: 0x22dfc90cbb8
Request Mode: X
Key Lock
HoBt ID: 72057594046644224
associated objid: 72057594046644224
Object name: Sample3.dbo.TableA
Index name: PK__TableA__3214EC276B6865C4
Owner Mode: X
Key Lock
HoBt ID: 72057594046709760
associated objid: 72057594046709760
Object name: Sample3.dbo.TableB
Index name: PK__TableB__3214EC27EFDE7565
Request Mode: X
Owner Mode: X
Server process Id: 55
Server batch Id: 0
Execution context Id: 0
Deadlock priority: 0
Log Used: 292
Owner Id: 226459
Transaction descriptor: 0x22de0ea2728



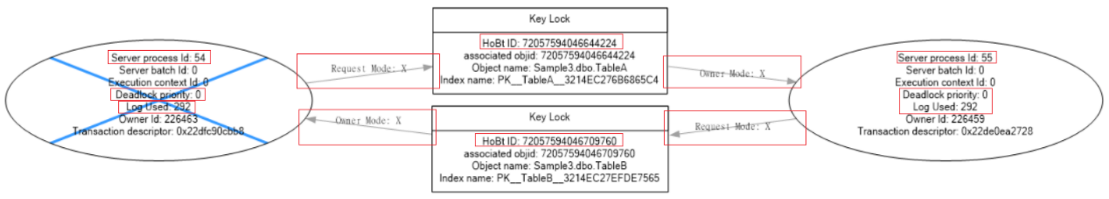
Statement:

```
--Ch82_01_04
--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;
```

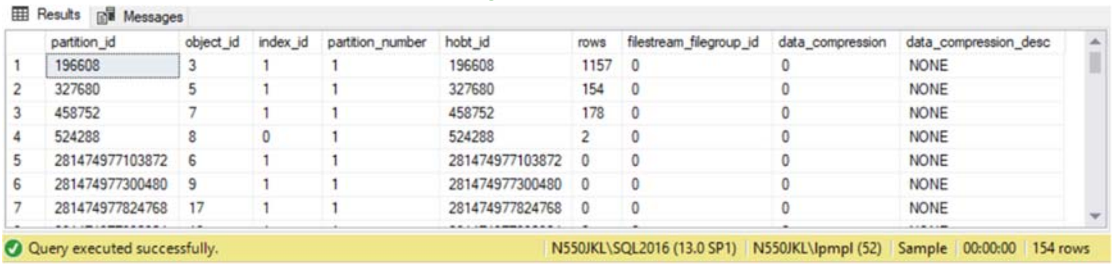


```
Server process Id: 55
Server batch Id: 0
Execution context Id: 0
Deadlock priority: 0
Log Used: 292
Owner Id: 226459
Tran                                    728
```

```
Statement:
--Ch82_01_03
--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;
```


Server process Id: 54
Server batch Id: 0
Execution context Id: 0
Deadlock priority: 0
Log Used: 292
Owner Id: 226463
Transaction descriptor: 0x22dfc90cbb8
Request Mode: X
Owner Mode: X
Key Lock
HoBt ID: 72057594046644224
associated objid: 72057594046644224
Object name: Sample3.dbo.TableA
Index name: PK__TableA__3214EC276B6865C4
Key Lock
HoBt ID: 72057594046709760
associated objid: 72057594046709760
Object name: Sample3.dbo.TableB
Index name: PK__TableB__3214EC27EFDE7565
Owner Mode: X
Request Mode: X
Server process Id: 55
Server batch Id: 0
Execution context Id: 0
Deadlock priority: 0
Log Used: 292
Owner Id: 226459
Transaction descriptor: 0x22de0ea2728

```
--========================================================================
--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
```

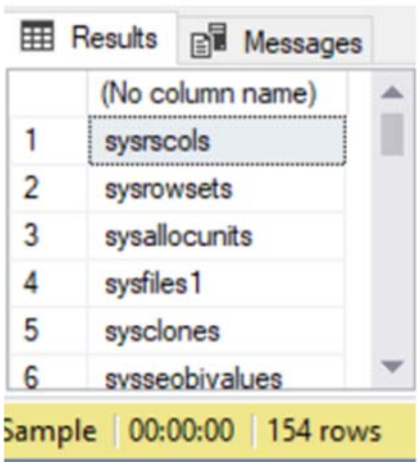


Results | Messages

	partition_id	object_id	index_id	partition_number	hobt_id	rows	filestream_filegroup_id	data_compression	data_compression_desc
1	196608	3	1	1	196608	1157	0	0	NONE
2	327680	5	1	1	327680	154	0	0	NONE
3	458752	7	1	1	458752	178	0	0	NONE
4	524288	8	0	1	524288	2	0	0	NONE
5	281474977103872	6	1	1	281474977103872	0	0	0	NONE
6	281474977300480	9	1	1	281474977300480	0	0	0	NONE
7	281474977824768	17	1	1	281474977824768	0	0	0	NONE

Query executed successfully. | N550JKL\SQL2016 (13.0 SP1) | N550JKL\lpmpl (52) | Sample | 00:00:00 | 154 rows

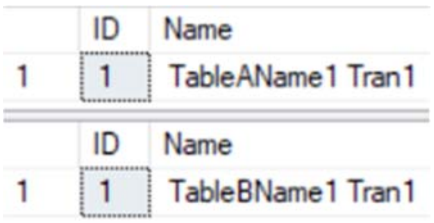
```
SELECT  OBJECT_NAME([object_id])
FROM    sys.partitions
GO -- Run the previous command and begins new batch
```


Results | Messages

```
     (No column name)
1    sysrscols
2    sysrowsets
3    sysallocunits
4    sysfiles1
5    sysclones
6    sysseobjvalues
```

Sample | 00:00:00 | 154 rows

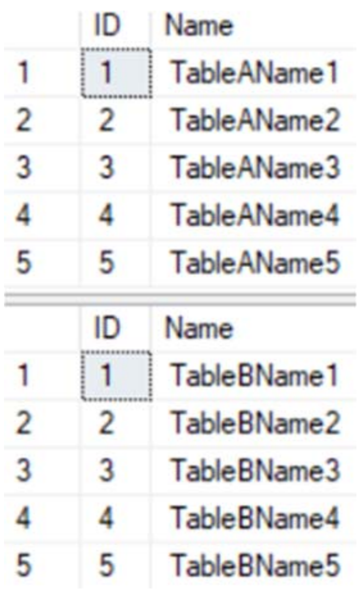
```
--============================================================================
--T025_05_07
--Step07: Check result
SELECT  *
FROM    dbo.TableA
WHERE   ID = 1;
SELECT  *
FROM    dbo.TableB
WHERE   ID = 1;
```



```
     ID   Name
1    1    TableAName1 Tran1

     ID   Name
1    1    TableBName1 Tran1
```

```
--============================================================================
--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)
);
```

```sql
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
```



	ID	Name
1	1	TableAName1
2	2	TableAName2
3	3	TableAName3
4	4	TableAName4
5	5	TableAName5

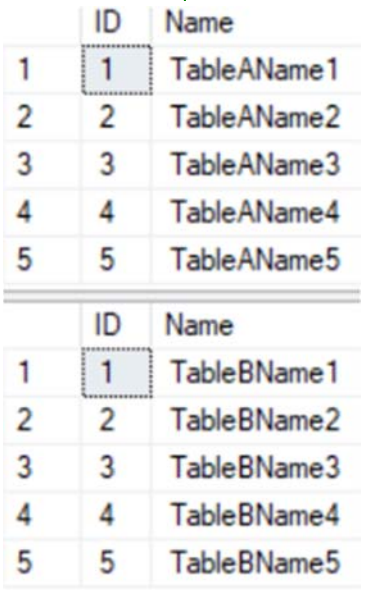
	ID	Name
1	1	TableBName1
2	2	TableBName2
3	3	TableBName3
4	4	TableBName4
5	5	TableBName5

==

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' );
```

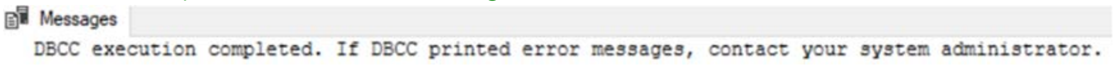
```
INSERT  INTO TableB
VALUES  ( 'TableBName3' );
INSERT  INTO TableB
VALUES  ( 'TableBName4' );
INSERT  INTO TableB
VALUES  ( 'TableBName5' );
GO -- Run the previous command and begins new batch
SELECT  *
FROM    TableA;
SELECT  *
FROM    TableB;
GO -- Run the previous command and begins new batch
```



	ID	Name
1	1	TableAName1
2	2	TableAName2
3	3	TableAName3
4	4	TableAName4
5	5	TableAName5

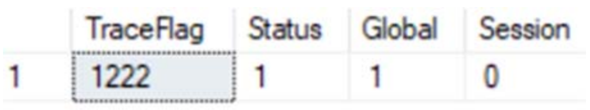
	ID	Name
1	1	TableBName1
2	2	TableBName2
3	3	TableBName3
4	4	TableBName4
5	5	TableBName5

```
--==============================================================================
--T025_06_02
--Turn On the trace flag
DBCC TRACEON(1222, -1);
GO -- Run the previous command and begins new batch
```



```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
```

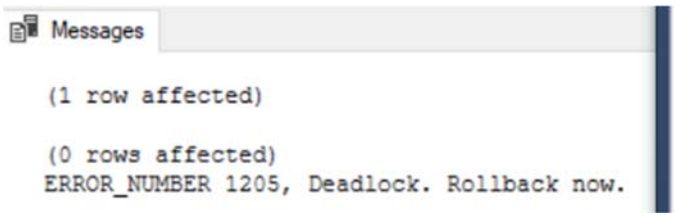
```
--Check the Trace Status.
DBCC TRACESTATUS(1222, -1);
GO -- Run the previous command and begins new batch
--Return 1 means Trace flag is enabled.
```



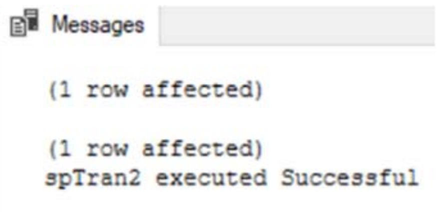
	TraceFlag	Status	Global	Session
1	1222	1	1	0

```
--==============================================================================
--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' ) )
```

```sql
    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
```

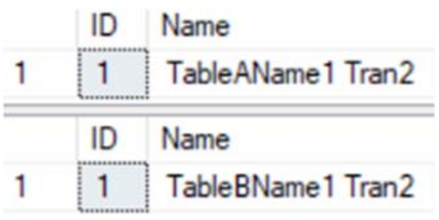


```
(1 row affected)

(0 rows affected)
ERROR_NUMBER 1205, Deadlock. Rollback now.
```



```
(1 row affected)

(1 row affected)
spTran2 executed Successful
```

```
--------------------------
--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
```

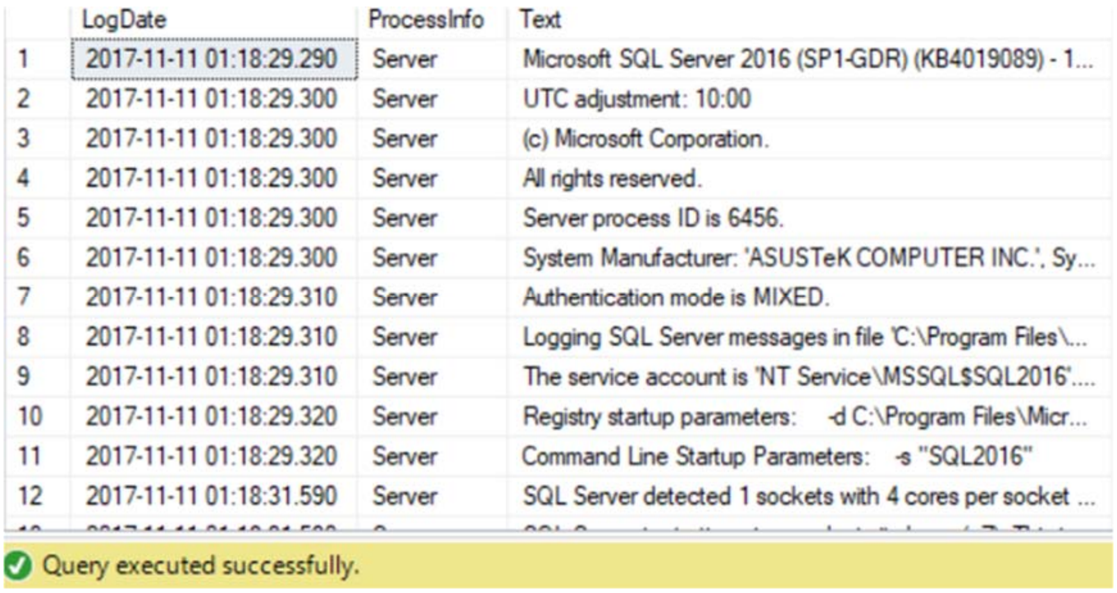


	ID	Name
1	1	TableAName1 Tran2

	ID	Name
1	1	TableBName1 Tran2

```
/*
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)
```

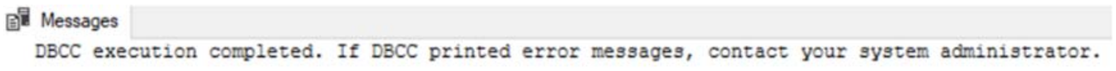
```
Turn Off the trace flag
...
--EXECUTE sp_readerrorlog;
To read the error log
1.2.
DBCC means Database Console Command.
SQL Server trace flag 1222 to write the deadlock information
to the SQL Server error log is one of the ways to
track down the queries that are causing deadlocks.
1.3.
-1 parameter 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
```



	LogDate	ProcessInfo	Text
1	2017-11-11 01:18:29.290	Server	Microsoft SQL Server 2016 (SP1-GDR) (KB4019089) - 1...
2	2017-11-11 01:18:29.300	Server	UTC adjustment: 10:00
3	2017-11-11 01:18:29.300	Server	(c) Microsoft Corporation.
4	2017-11-11 01:18:29.300	Server	All rights reserved.
5	2017-11-11 01:18:29.300	Server	Server process ID is 6456.
6	2017-11-11 01:18:29.300	Server	System Manufacturer: 'ASUSTeK COMPUTER INC.', Sy...
7	2017-11-11 01:18:29.310	Server	Authentication mode is MIXED.
8	2017-11-11 01:18:29.310	Server	Logging SQL Server messages in file 'C:\Program Files\...
9	2017-11-11 01:18:29.310	Server	The service account is 'NT Service\MSSQL$SQL2016'....
10	2017-11-11 01:18:29.320	Server	Registry startup parameters: -d C:\Program Files\Micr...
11	2017-11-11 01:18:29.320	Server	Command Line Startup Parameters: -s "SQL2016"
12	2017-11-11 01:18:31.590	Server	SQL Server detected 1 sockets with 4 cores per socket ...

Query executed successfully.

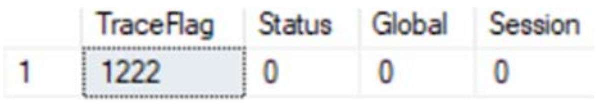
```
--==========================================================================
--T025_06_05
--Turn Off the trace flag
DBCC TRACEOFF(1222, -1);
GO -- Run the previous command and begins new batch
```



Messages

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
```

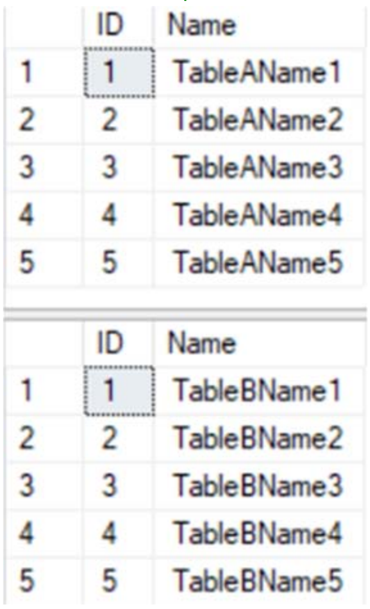
```
--Check the Trace Status.
DBCC TRACESTATUS(1222, -1);
GO -- Run the previous command and begins new batch
--0 means the trace flag is disabled.
```



	TraceFlag	Status	Global	Session
1	1222	0	0	0

```
--==========================================================================
--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
```



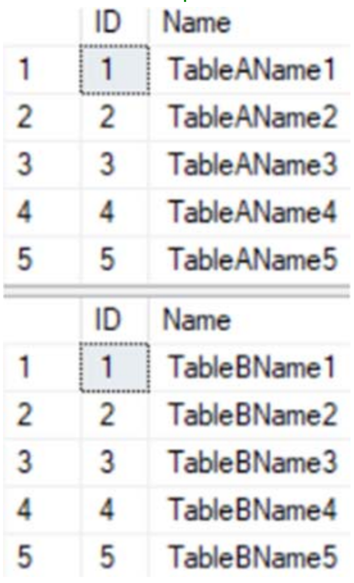
```
   ID  Name
1  1   TableAName1
2  2   TableAName2
3  3   TableAName3
4  4   TableAName4
5  5   TableAName5

   ID  Name
1  1   TableBName1
2  2   TableBName2
3  3   TableBName3
4  4   TableBName4
5  5   TableBName5
```

==

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
```



	ID	Name
1	1	TableAName1
2	2	TableAName2
3	3	TableAName3
4	4	TableAName4
5	5	TableAName5

	ID	Name
1	1	TableBName1
2	2	TableBName2
3	3	TableBName3
4	4	TableBName4
5	5	TableBName5

```
--========================================================================
--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
```

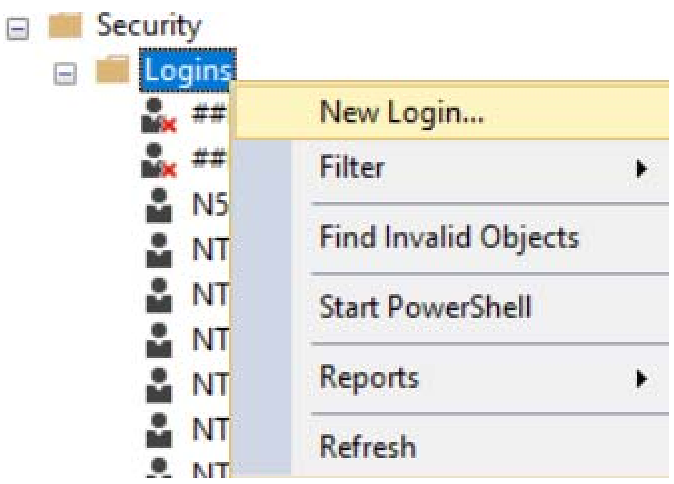
==

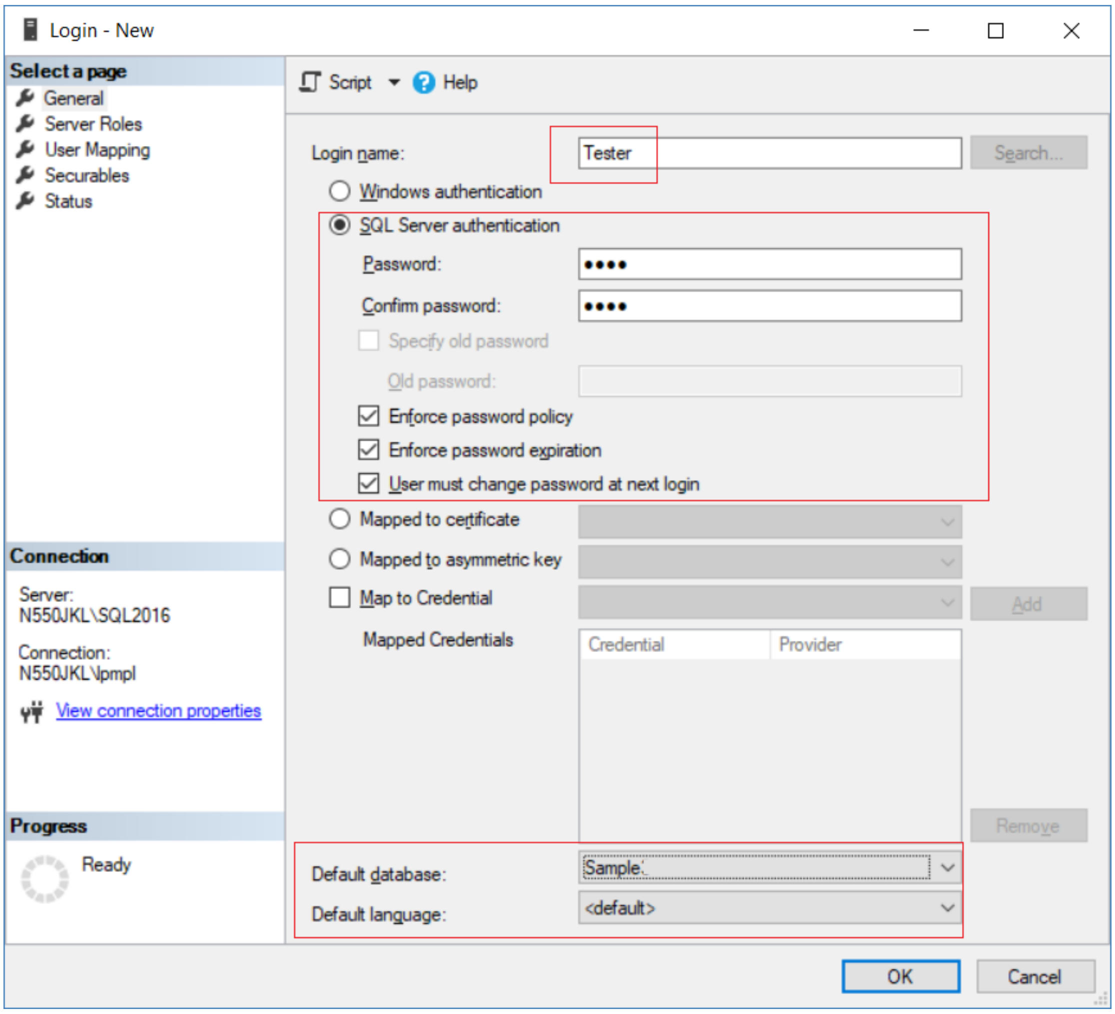
8. Asp.Net Handling Deadlocks

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.





Login name: Tester

( ) Windows authentication

(●) SQL Server authentication

Password: ••••

Confirm password: ••••

☐ Specify old password

Old password:

☑ Enforce password policy

☑ Enforce password expiration

☑ User must change password at next login

( ) Mapped to certificate

( ) Mapped to asymmetric key

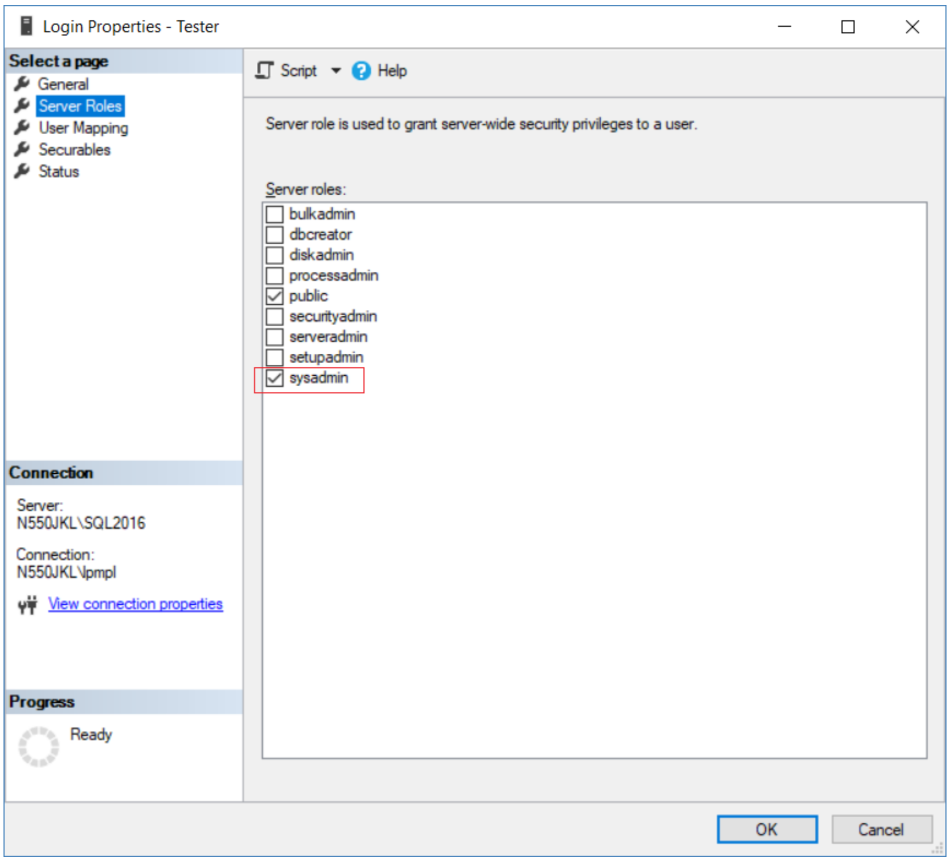
☐ Map to Credential

Server: N550JKL\SQL2016

Connection: N550JKL\pmpl

Default database: Sample

Default language: <default>



Server role is used to grant server-wide security privileges to a user.

Server roles:

☐ bulkadmin

☐ dbcreator

☐ diskadmin

☐ processadmin

☑ public

☐ securityadmin

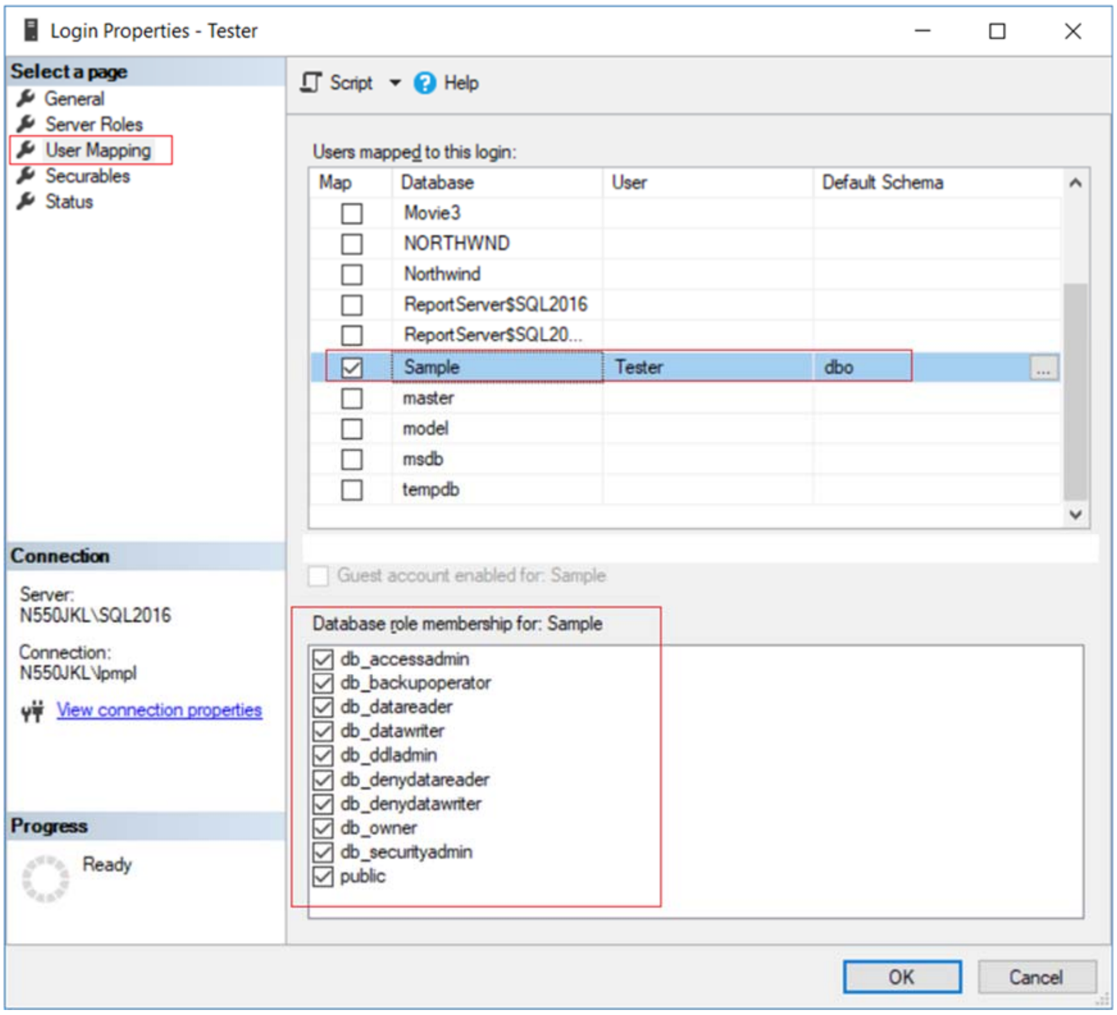
☐ serveradmin

☐ setupadmin

☑ sysadmin

Server: N550JKL\SQL2016

Connection: N550JKL\pmpl



Server:
N550JKL\SQL2016

Connection:
N550JKL\lpmpl

Users mapped to this login:

Map	Database	User	Default Schema
☐	Movie3		
☐	NORTHWND		
☐	Northwind		
☐	ReportServer$SQL2016		
☐	ReportServer$SQL20...		
☑	Sample	Tester	dbo
☐	master		
☐	model		
☐	msdb		
☐	tempdb		

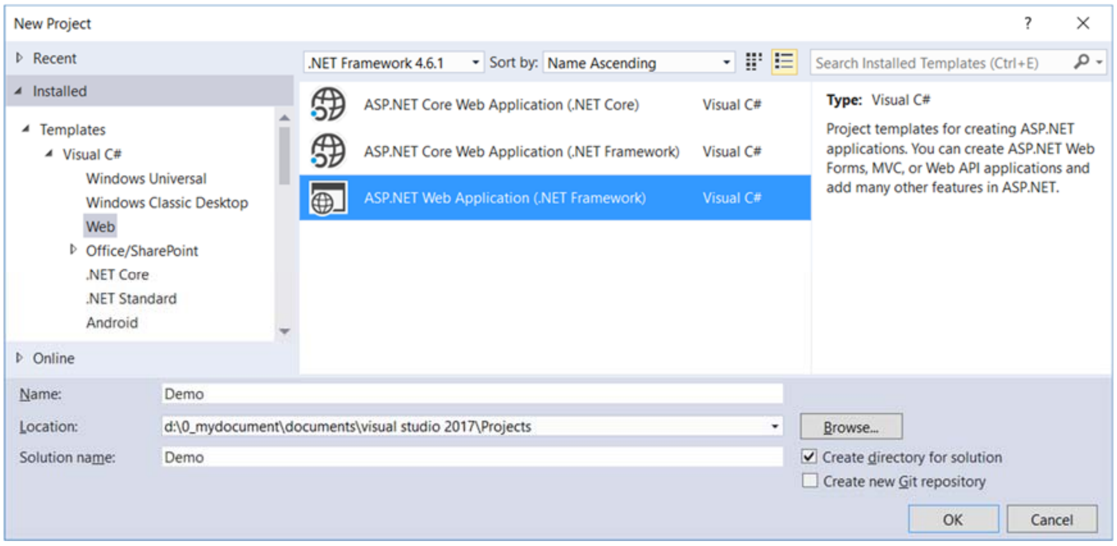
☐ Guest account enabled for: Sample

Database role membership for: Sample

- ☑ db_accessadmin
- ☑ db_backupoperator
- ☑ db_datareader
- ☑ db_datawriter
- ☑ db_ddladmin
- ☑ db_denydatareader
- ☑ db_denydatawriter
- ☑ db_owner
- ☑ db_securityadmin
- ☑ public

8.2. Create Web Application

New Project --> Web --> ASP.NET Web Application (.Net Framework)
-->
Name:
Demo
--> Web Forms --> OK



ASP.NET Core Web Application (.NET Core)	Visual C#
ASP.NET Core Web Application (.NET Framework)	Visual C#
ASP.NET Web Application (.NET Framework)	Visual C#

Type: Visual C#

Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

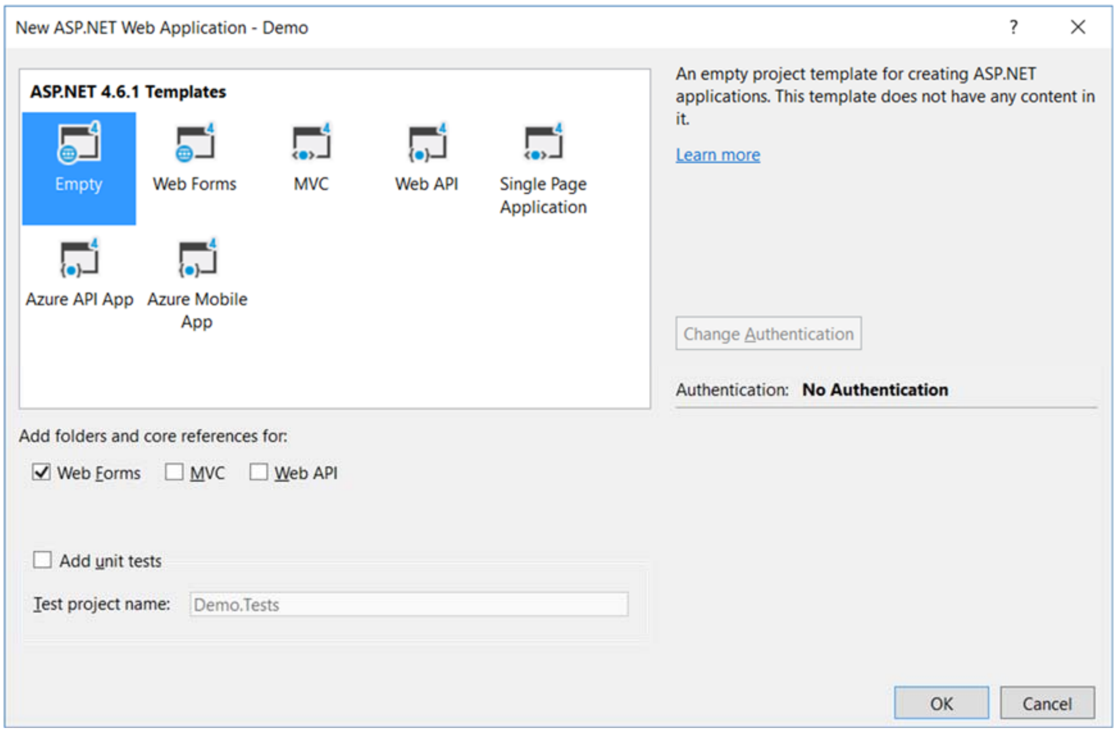
Name: Demo

Location: d:\0_mydocument\documents\visual studio 2017\Projects

Solution name: Demo

☑ Create directory for solution

☐ Create new Git repository


New ASP.NET Web Application - Demo

ASP.NET 4.6.1 Templates

An empty project template for creating ASP.NET applications. This template does not have any content in it.

Authentication: **No Authentication**

Add folders and core references for:

[✓ Web Forms] MVC Web API

Add unit tests

Test project name: Demo.Tests

8.3. Code

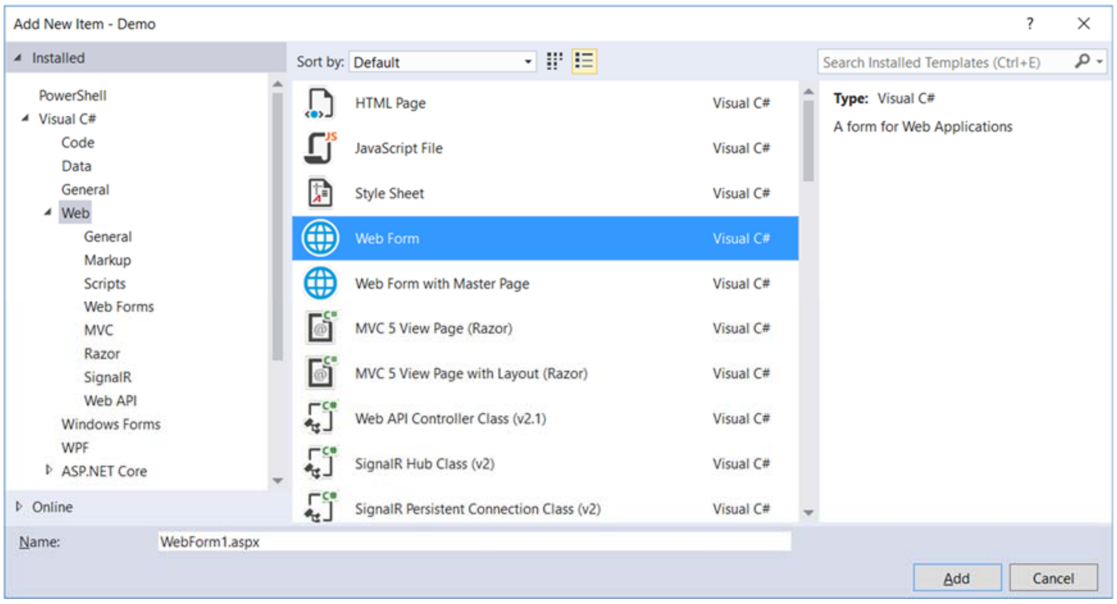
8.3.1. Web.config

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


Add New Item - Demo
Installed
PowerShell
Visual C#
Code
Data
General
Web
General
Markup
Scripts
Web Forms
MVC
Razor
SignalR
Web API
Windows Forms
WPF
ASP.NET Core
Online
HTML Page Visual C#
JavaScript File Visual C#
Style Sheet Visual C#
Web Form Visual C#
Web Form with Master Page Visual C#
MVC 5 View Page (Razor) Visual C#
MVC 5 View Page with Layout (Razor) Visual C#
Web API Controller Class (v2.1) Visual C#
SignalR Hub Class (v2) Visual C#
SignalR Persistent Connection Class (v2) Visual C#
Search Installed Templates (Ctrl+E)
Add
Cancel

Sort by: Default

Type: Visual C#

A form for Web Applications

Name: WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Demo.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">;
<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

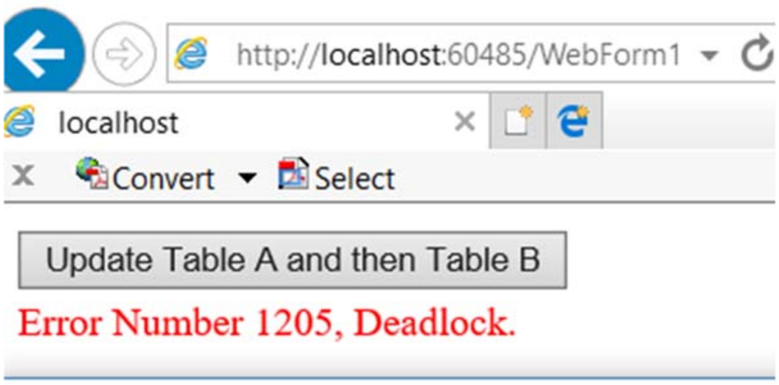
```
using System;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Web.UI;
namespace Demo
{
    public partial class WebForm1 : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void 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 = 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

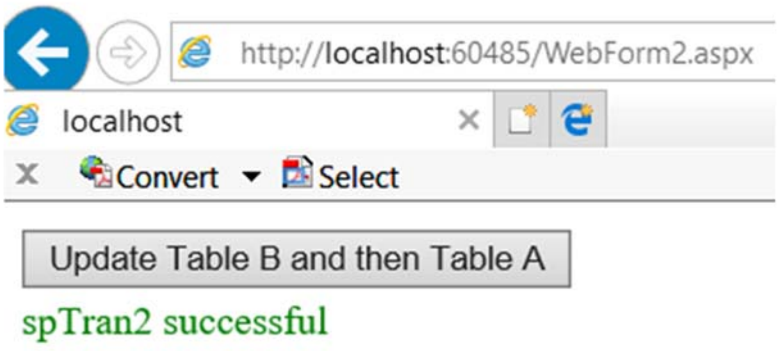
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs" Inherits="Demo.WebForm2" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <table>
            <tr>
                <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 = 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;
            }
        }
    }
}
```


http://localhost:60485/WebForm1
localhost
Convert
Select
Update Table A and then Table B

Error Number 1205, Deadlock.


http://localhost:60485/WebForm2.aspx
localhost
Convert
Select
Update Table B and then Table A

spTran2 successful