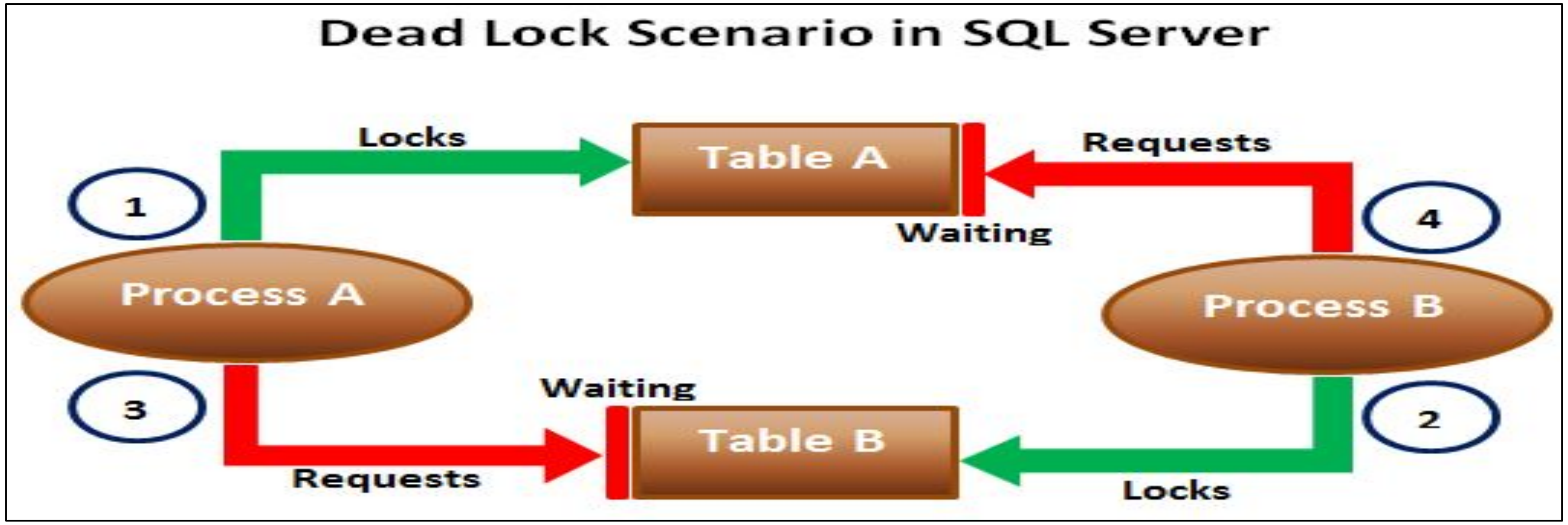
Database Design & Applications

Deadlock

When can a deadlock occur



- In a database, a deadlock occurs when two or more processes have a resource locked, and each process requests a lock on the resource that another process has already locked.
- Neither of the transactions here can move forward, as each one is waiting for the other to release the lock.

Deadlock Scenario in SQLServer

- Let us look at this in action. We will use the following 2 tables for this example.
- SQL script to create the tables and populate them with test data.

Table A	
Id	**Name**
1	Mark

Table B	
Id	**Name**
1	Mary

```
Create table TableA
(
   Id int identity primary key,
   Name nvarchar(50)
)
Insert into TableA values ('Mark')
Create table TableB
(
   Id int identity primary key,
   Name nvarchar(50)
)
Insert into TableB values ('Mary')
```

Deadlock Scenario in SQLServer

The following 2 transactions will result in a dead lock. Open 2 instances of SQL Server Management studio. From the first window execute Transaction 1 code and from the second window execute Transaction 2 code.

-- Transaction 1	-- Transaction 2
Begin Tran	Begin Tran
Update TableA Set Name = 'Mark Transaction 1' where Id = 1	Update TableB Set Name = 'Mark Transaction 2' where Id = 1
-- From Transaction 2 window execute the first update statement	-- From Transaction 1 window execute the second update statement
Update TableB Set Name = 'Mary Transaction 1' where Id = 1	Update TableA Set Name = 'Mary Transaction 2' where Id = 1
-- From Transaction 2 window execute the second update statement	-- After a few seconds notice that one of the transactions complete
Commit Transaction	-- successfully while the other transaction is made the deadlock victim
	Commit Transaction

Dead Detection

- How SQL Server detects deadlocks
  - Lock monitor thread in SQL Server, runs every 5 seconds by default to detect if there are any deadlocks.
  - If the lock monitor thread finds deadlocks, the deadlock detection interval will drop from 5 seconds to as low as 100 milliseconds depending on the frequency of deadlocks.
  - If the lock monitor thread stops finding deadlocks, the Database Engine increases the intervals between searches to 5 seconds.
- What happens when a deadlock is detected
  - When a deadlock is detected, the Database Engine ends the deadlock by choosing one of the threads as the deadlock victim.
  - The deadlock victim's transaction is then rolled back and returns a 1205 error to the application.
  - Rolling back the transaction of the deadlock victim releases all locks held by that transaction.
  - This allows the other transactions to become unblocked and move forward.

Deadlock Priority

- By default, SQL Server chooses a transaction as the deadlock victim that is least expensive to roll back.
- However, a user can specify the priority of sessions in a deadlock situation using the SET DEADLOCK_PRIORITY statement.
- The session with the lowest deadlock priority is chosen as the deadlock victim.
- **Example :**

  SET DEADLOCK_PRIORITY NORMAL
- **DEADLOCK_PRIORITY**
  1. The default is Normal
  2. Can be set to LOW, NORMAL, or HIGH
  3. Can also be set to a integer value in the range of -10 to 10.
- **LOW** : -5
- **NORMAL** : 0
- **HIGH** : 5

Deadlock Victim Selection

- What is the deadlock victim selection criteria
  1. If the DEADLOCK_PRIORITY is different, the session with the lowest priority is selected as the victim.
  2. If both the sessions have the same priority, the transaction that is least expensive to rollback is selected as the victim
  3. If both the sessions have the same deadlock priority and the same cost, a victim is chosen randomly

Deadlock Victim Selection: Example

SQL Script to setup the tables for the examples

```
Create table TableA
(
   Id int identity primary key,
   Name nvarchar(50)
)
Insert into TableA values ('Mark')
Insert into TableA values ('Ben')
Insert into TableA values ('Todd')
Insert into TableA values ('Pam')
Insert into TableA values ('Sara')
```

```
Create table TableB
(
   Id int identity primary key,
   Name nvarchar(50)
)
Insert into TableB values ('Mary')
```

Deadlock Victim Selection: Example

- Open 2 instances of SQL Server Management studio.
- From the first window execute Transaction 1 code and from the second window execute Transaction 2 code.
- We have not explicitly set DEADLOCK_PRIORITY, so both the sessions have the default DEADLOCK_PRIORITY which is NORMAL.
- So in this case SQL Server is going to choose Transaction 2 as the deadlock victim as it is the least expensive one to rollback.

```
-- Transaction 1
Begin Tran
Update TableA Set Name = Name + ' Transaction 1' where Id IN (1, 2, 3,
4, 5)

-- From Transaction 2 window execute the first update statement

Update TableB Set Name = Name + ' Transaction 1' where Id = 1

-- From Transaction 2 window execute the second update statement
Commit Transaction
```

```
-- Transaction 2
Begin Tran
Update TableB Set Name = Name + ' Transaction 2' where Id = 1

-- From Transaction 1 window execute the second update statement

Update TableA Set Name = Name + ' Transaction 2' where Id IN (1, 2, 3, 4, 5)

-- After a few seconds notice that this transaction will be chosen as the
deadlock
-- victim as it is less expensive to rollback this transaction than Transaction
1
Commit Transaction
```

Setting Deadlock Priority

- In the following example we have set DEADLOCK_PRIORITY of Transaction 2 to HIGH.
- Transaction 1 will be chosen as the deadlock victim, because it's DEADLOCK_PRIORITY (Normal) is lower than the DEADLOCK_PRIORITY of Transaction 2.

```
-- Transaction 1
Begin Tran
Update TableA Set Name = Name + ' Transaction 1' where Id IN (1, 2, 3,
4, 5)

-- From Transaction 2 window execute the first update statement

Update TableB Set Name = Name + ' Transaction 1' where Id = 1

-- From Transaction 2 window execute the second update statement
Commit Transaction
```

```
-- Transaction 2
SET DEADLOCK_PRIORITY HIGH
Begin Tran
Update TableB Set Name = Name + ' Transaction 2' where Id = 1

-- From Transaction 1 window execute the second update statement

Update TableA Set Name = Name + ' Transaction 2' where Id IN (1, 2, 3, 4,
5)

-- After a few seconds notice that Transaction 2 will be chosen as the
-- deadlock victim as it's DEADLOCK_PRIORITY (Normal) is lower than the
-- DEADLOCK_PRIORITY this transaction (HIGH)
Commit Transaction
```

Keeping track of deadlocks

- There are various tools that can be used to obtain the details of deadlocks. These include trace flags 1204 and 1222. You can also capture the deadlock graph event using SQL Profiler.
- Personally I find that when I suspect that deadlocking is occurring in my server, that setting up and extended event session to log the deadlock graph each time it happens is the easiest.
- From SQL Server 2012 onwards this can be done in SQL Server Management Studio under Management \ Extended Events

How to minimize deadlocks

- Always try to hold locks for as short a period as possible.
- Always access resources in the same order
- Ensure that you don’t have to wait on user input in the middle of a transaction. First, get all the information you need and then submit the transaction
- Use READ COMMITTED SNAPSHOT ISOLATION or SNAPSHOT ISOLATION

THANK YOU!