

Concurrent restore deadlock on master database

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:31 AM PST

Contents

- [Issue](#)
- [Investigation/Analysis](#)
- [Mitigation](#)
- [Internal reference](#)

Issue

Customer performing multiple database restores in parallel (restore using RESTORE DATABASE command). Some restores eventually fail with a deadlock exception.

```
System.Data.SqlClient.SqlException (0x80131904): Transaction (Process ID 119) was deadlocked on lock resources  
RESTORE DATABASE is terminating abnormally.
```



Investigation/Analysis

When the [deadlock](#) details are obtained from customer side, we can see that the deadlock is occurring on the master database, more precisely on the **sys.sysdbreg** table.

From our side, we can see two different transactions involved on the deadlock.

One transaction is holding an Exclusive lock on a DATABASE and tries to obtain a Shared lock on a object. This object contains an Exclusive lock held by a transaction that tries to obtain another exclusive lock on the database.

```


<deadlock>
  <victim-list>
    <victimProcess id="process2a692c2c4e8"/>
  </victim-list>
  <process-list>
    <process id="process2a692c2c4e8" taskpriority="0" logused="0" waitresource="KEY: 1:562949955256320 (a9cf246a
      <stackFrames>
        (..)
      </stackFrames>
    </process>
    <process id="process2a852e6e108" taskpriority="0" logused="10000" waitresource="DATABASE: 32760:0 " waittime
      <stackFrames>
        (...)
      </stackFrames>
    </process>
  </process-list>
  <resource-list>
    <keylock hobtid="562949955256320" dbid="1" objectname="filtered" indexname="filtered" id="lock2a796f7db00" m
      <owner-list>
        <owner id="process2a852e6e108" mode="X"/>
      </owner-list>
      <waiter-list>
        <waiter id="process2a692c2c4e8" mode="S" requestType="wait"/>
      </waiter-list>
    </keylock>
    <databaselock subresource="FULL" dbid="32760" dbname="filtered" lockPartition="0" id="lock2a7e7c46700" mode=
      <owner-list>
        <owner id="process2a692c2c4e8" mode="S"/>
        <owner id="process2a692c2c4e8" mode="X"/>
      </owner-list>
      <waiter-list>
        <waiter id="process2a852e6e108" mode="X" requestType="wait"/>
      </waiter-list>
    </databaselock>
  </resource-list>
</deadlock>

```

When a RESTORE DATABASE command is issued, the process must access multiple system tables to start and complete the operation. When multiple parallel RESTORES are issued, it means that different operations will try to access the same system tables and, like expected, place locks on those resources (even if briefly).

Mitigation

Two possible paths for mitigation:

- use [retry logic](#) 
- scale up the Managed Instance to Business Critical so that the restore operation is done on faster storage.

Internal reference

[363335455](#) 

How good have you found this content?



-