**HAV-NEMOQ Database Maintenance**

Last updated: November 16, 2022

1. **Purpose:**

To maintain the validity of information that resides in a database, it is imperative that the logical and physical consistency of every database object be verified according to the business’s Recovery Time Objectives (RTO). Failure to perform these operations can result in corrupt or lost data. For this reason, it is best practices to run database integrity checks (DBCC CHECKDB) on a regular basis.

This task is normally accomplished in full blown (e.g., Standard or Enterprise) versions of SQL Server by creating a SQL job Agent. However, this feature is not available in SQL Express. The HAV-Nemoq SQL instance here at GLFHC is an Express version. In this case, these operations must be conducted through a Windows Scheduled Task.

This document describes how that task was constructed.

II **Methods and Procedures:**

To achieve this goal, two directories named Code and Output must be created. All source code is stored in the Code folder whereas the resulting output files are placed in the Output folder. A Powershell script was then crafted to run an integrity check on every database (system and user) that resides on the HAV-NEMOQ/SQLExpress instance. The results of these checks are stored in a file within the Output directory and emailed to the DBA. This script also purges the number of files that reside in the Output location to prevent unnecessary usage of that directory. A version of this script can be found in the Appendix.

Once that this code was written and tested, the next step was to generate a Windows Scheduled Task that would execute this procedure on a daily basis starting at 10:27 PM. An exported version of that task, called HAV-NEMOQ DBCC CheckDB Job, can also be seen in the Appendix.

It is the DBA’s responsibility to review these results and take corrective action, if needed.

**Appendix**

**PowerShell Script:**

<# DBCC\_CHECKDB.ps1

# Purpose: There are instances of SQLExpress out there. It's not possible to run data integrity

# checks in this case using native SQL. So, the next best thing is to write a

# PS script that can be automated through a Windows Scheduled Task.

# ref: http://sqlmag.com/powershell/run-sql-server-dbccs-check-errorlog-with-powershell

# Date Author Description

#--------- ------- ----------------------------

# 10/21/2021 peb Modified for GLFHC

# 11/14/2022 peb Modified and tested in the HAV-NEMOQ environment

# define date parameters

#>

$today = (get-date).toString()

$all = @()

$startdt = ((get-date).adddays(-1)).ToString() # look back n days from current time

$ThisServer = get-content env:computername

# Define the output file

$FilePath = "C:\Output"

$OutFile = Join-Path -path $FilePath -childPath ("SQLDBCCCheckDBReport\_" + (get-date).toString('yyyyMMdd\_hhmmtt') + ".txt")

$Daysback = "-7"

$Logfile = Join-Path -path $FilePath -childPath ("DeleteLogs\_" + (get-date).toString('yyyyMMdd\_hhmmtt') + ".txt")

$header = "The Files to be deleted are listed here:"

$header | Out-file $Logfile

$CurrentDate = Get-Date

$DatetoDelete = $CurrentDate.AddDays($Daysback)

#let's set up the email stuff

$emailFrom = "SQLCMSAlert@glfhc.org"

$emailTo = "pburkhardt@glfhc.org"

$subject = "DBCC SQL CheckDB Report for $startdt on $ThisServer server..."

$body = "Review the attached file for any inconsistency errors and please research any problems found..."

$smtpServer = "mail.glfhc.org"

#format Header of Report

$Line = "DBCC Data Consistency Databases Checks for $today"

$Line | out-file $OutFile

# Connect to the specified instance

$inst = "HAV-NEMOQ\SQLEXPRESS"

# Load SMO extension

[System.Reflection.Assembly]::LoadWithPartialName('Microsoft.SqlServer.SMO') | out-null

$s = New-Object ('Microsoft.SqlServer.Management.Smo.Server') $inst

# Get the databases for the instance, and iterate through them

# In Server Management Objects (SMO), the Database object has a method called CheckTables().

# Use this with an argument of 'None,' and this has the equivalent of running DBCC CHECKDB WITH NO\_INFOMSGS for each database.

$dbs = $s.Databases

foreach ($db in $dbs)

{

# Check to make sure the database is accessible

if ($db.IsAccessible -eq $True)

{

# Store the database name for reporting

$dbname = $db.Name

# Peform the database check

$db.CheckTables('None')

}

}

#$Line = "'r' n———————————–"

#$Line | out-file $OutFile -append

# Now, if there were problems with the DBCC results, they will appear in the error log

# So, let's read the current error log

$err = $s.ReadErrorLog()

# search the error log for the past 24 hours and look for the DBCC results

$err | where {$\_.LogDate -ge $startdt} | Select-String -inputobject {[string] $\_.LogDate + ' ' + $\_.ProcessInfo + ' ' + $\_.Text} -pattern 'DBCC' -context 0,0 | Out-File $OutFile -append

# Now, list all files in the Output directory to a text file that are older than X days

Get-ChildItem $FilePath | Where-Object { $\_.LastWriteTime -lt $DatetoDelete } | Out-file $Logfile -append

# Next, delete these files

Get-ChildItem $FilePath | Where-Object { $\_.LastWriteTime -lt $DatetoDelete } | Remove-Item

# Finally format and send email

Function sendEmail([string]$emailFrom, [string]$emailTo, [string]$subject,[string]$body,[string]$smtpServer,[string]$OutFile,[string]$Logfile)

{

#initate message

$email = New-Object System.Net.Mail.MailMessage

$email.From = $emailFrom

$email.To.Add($emailTo)

$email.Subject = $subject

$email.Body = $body

# initiate email attachment

$emailAttach = New-Object System.Net.Mail.Attachment $OutFile

$email.Attachments.Add($emailAttach)

$emailAttach2 = New-Object System.Net.Mail.Attachment $Logfile

$email.Attachments.Add($emailAttach2)

#initiate sending email

$smtp = new-object Net.Mail.SmtpClient($smtpServer)

$smtp.Send($email)

}

#Send out the results before existing

sendEmail $emailFrom $emailTo $subject $body $smtpServer $OutFile $Logfile

**HAV-NEMOQ DBCC CheckDB Job:**

<?xml version="1.0" encoding="UTF-16"?>

<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">

<RegistrationInfo>

<Date>2021-11-02T08:33:08.1977516</Date>

<Author>GLFHC\xpburkhardt</Author>

<Description>Runs integrity checks on all database and emails DBA</Description>

<URI>\HAV-NEMOQ DBCC CHeckDB Job</URI>

</RegistrationInfo>

<Triggers>

<CalendarTrigger>

<StartBoundary>2022-11-15T22:27:09</StartBoundary>

<Enabled>true</Enabled>

<ScheduleByDay>

<DaysInterval>1</DaysInterval>

</ScheduleByDay>

</CalendarTrigger>

</Triggers>

<Principals>

<Principal id="Author">

<UserId>S-1-5-21-102828792-201003079-1231754661-34879</UserId>

<LogonType>Password</LogonType>

<RunLevel>HighestAvailable</RunLevel>

</Principal>

</Principals>

<Settings>

<MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>

<DisallowStartIfOnBatteries>true</DisallowStartIfOnBatteries>

<StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>

<AllowHardTerminate>true</AllowHardTerminate>

<StartWhenAvailable>false</StartWhenAvailable>

<RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>

<IdleSettings>

<StopOnIdleEnd>true</StopOnIdleEnd>

<RestartOnIdle>false</RestartOnIdle>

</IdleSettings>

<AllowStartOnDemand>true</AllowStartOnDemand>

<Enabled>true</Enabled>

<Hidden>false</Hidden>

<RunOnlyIfIdle>false</RunOnlyIfIdle>

<WakeToRun>false</WakeToRun>

<ExecutionTimeLimit>PT2H</ExecutionTimeLimit>

<Priority>7</Priority>

</Settings>

<Actions Context="Author">

<Exec>

<Command>C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe</Command>

<Arguments>-ExecutionPolicy Bypass -File "C:\Code\SQLExpressDBCC\_CHECKDB.ps1"</Arguments>

</Exec>

</Actions>

</Task>