Maintenance Task Baseline and Information

The SQL DBA team in OAB will configuring and monitoring maintenance plans and agent jobs to ensure database optimization. We will create an Agent job to selectively rebuild or reorganize database indexes, an Agent job to check database integrity, a maintenance plan to clear backup, agent log and history records over four weeks and an Agent job that will recycle the SQL Error Log every week – also keeping the maximum amount of error logs. The details of the plan and agent jobs are below.

**Rebuilding/Reorganizing of Indexes**

The DBA team will create an agent job that rebuilds all fragmented indexes on all databases. The following are the job settings:

* Job Type: SQL Agent Job
* Job name: DBA: Performance\_Optimization
* Owner: The SA account
* Description: This agent job provides a fully parameterized T-SQL script which identifies only the fragmented indexes in all databases in a SQL Server instance, and either generates a report with the defragmentation commands for review and controlled execution, or directly reorganizes or rebuilds the fragmented indexes based on Microsoft's recommendations. The script also supports AlwaysOn AG-enabled SQL Server instances. The job runs every Wednesday and Sunday at 4:25 AM <Just copy and pasted this text>
* Steps:
  + Index\_Optmization – Copy the script from Appendix A into the step
  + Update\_Statistics – EXEC sp\_MSforeachdb 'use [?]EXEC sp\_updatestats'
* Schedule:
  + Name: Wed&SunAt4:25AM
* Notification
  + E-Mail – OAB (Operator) When job failes
  + Write to the Windows Application Event Log – when job completes
* Reporting: None at this time

**Integrity check**

The DBA team will create an Agent job that performs a database integrity check on all databases. The following are the job settings.

* Job Type: SQL Agent Job
* Job name: DBA: DBCC\_CHECKDB
* Owner: The SA account
* Description: This agent job will execute the DBCC CheckDB command on all database of the instance. The job will run every Sunday at 9:15 PM <Just copy and pasted this text>
* Steps Insert the below query

EXEC msdb.dbo.sp\_send\_dbmail

@profile\_name = 'OAB\_SQL\_Alert\_HREWWDEVDB1919',

@recipients = 'ayoubj@state.gov;collinstd2@state.gov;',

@body = 'This is a test to send SQL Query results through email! -Tim',

@subject = '(TRA) Neat Right? =)',

@query = 'EXEC sp\_MSforeachdb ''use [?] dbcc checkdb''',

@attach\_query\_result\_as\_file = 1;

* Schedule
  + Name: Sunday\_9:15PM
* Notification
  + E-Mail – OAB (Operator) when job completes
  + Write to the Windows Application Event Log – when job completes
* Reporting: None at this time

**Maintenance plan Log & History Cleanup We need to create a four weeks retention**

The DBA team will create a Maintenance plan that will include removing all history older than four weeks. The follow are the details of the job

* Job Type: Maintenance Plan
* Job Name: DBA: Log and History Cleanup
* Job Description: The maintenance plan will remove agent log and history records over four weeks. The plan will execute every Sunday at 1:00AM
* Job Task: Clean Up History Backup, Job, Maintenance Plan older than 4 weeks
* Reporting: None at this time
* Notification – Configured in the agent job sub plan
  + E-Mail – OAB (Operator) When job fails
  + Write to the Windows Application Event Log – when job completes

**SQL Log Recycle error logs to happen weekly and hold 99 logs**

The DBA team will create a SQL Agent job that will recycle the SQL Error Log week and we will also set the “Maximum number of error log files before they are recycled”

Step One: Configure the SQL Error Logs to keep 99 logs

Step Two: Create the below agent job

* Job Type: SQL Agent Job
* Job name: DBA: Recycle\_SQL\_Error\_Log
* Owner: The SA account
* Description: This agent job will execute the DBCC CheckDB command on all database of the instance. The job will run every Sunday at 1:15AM <Just copy and pasted this text>
* Steps
  + Recycle\_Log - EXEC sp\_cycle\_errorlog
* Schedule
  + Name: Sunday\_1:15AM
* Notification
  + E-Mail – OAB (Operator) When job fails
  + Write to the Windows Application Event Log – when job completes
* Reporting: None at this time

# Appendix A: the Index Optimization Script

The parameters for this script are:

It will scan all databases

It maintains only the indexes that have average fragmentation percentage equal or higher than 5 %;

The fill factor = 90;

Sets the Scanning mode to ='Detailed';

Sorts intermediate index results in TempDB

Sorts intermediate index results in user database's log file SET @sortInTempdb='ON';

----------------------------------------------Script Starts here---------------------------------------------------------------------

----

-- Script that reorganizes or rebuilds all indexes having an average fragmentation

-- percentage above a given threshold. It also works in the case

-- where Availability Groups are enabled as it determines if the

-- relevant databases are the primary replicas.

--

-- This script supports only SQL Server 2005 or later.

-- Also, if you execute this script in a SQL Server 2005 instance

-- or later, any databases with compatibility level 2000 (80) or earlier

-- will be automatically excluded from the index reorganization/rebuild process.

----

--Initial check - You must be SysAdmin

DECLARE @isSysAdmin INT

SET @isSysAdmin=(SELECT IS\_SRVROLEMEMBER ('sysadmin'));

--Initial check - You must be using SQL Server 2005 or later

DECLARE @SQLServerVersion INT

SET @SQLServerVersion=(SELECT CAST(LEFT(CAST(SERVERPROPERTY('ProductVersion') AS VARCHAR(50)),CHARINDEX('.',CAST(SERVERPROPERTY('ProductVersion') AS VARCHAR(50)))-1) AS INT));

IF @isSysAdmin=1 AND @SQLServerVersion >= 9

BEGIN

--

-- Variable/parameters Declaration

--

DECLARE @dbname NVARCHAR(128);

DECLARE @ReorganizeOrRebuildCommand NVARCHAR(MAX);

DECLARE @dbid INT;

DECLARE @indexFillFactor VARCHAR(5);

DECLARE @fragmentationThreshold VARCHAR(10);

DECLARE @indexStatisticsScanningMode VARCHAR(20);

DECLARE @verboseMode BIT;

DECLARE @reportOnly BIT;

DECLARE @sortInTempdb VARCHAR(3);

DECLARE @isHadrEnabled BIT;

DECLARE @databaseToCheck VARCHAR(250)

DECLARE @dynamic\_command NVARCHAR(1024);

DECLARE @dynamic\_command\_get\_tables NVARCHAR(MAX);

--Initializations - Do not change

SET @databaseToCheck=NULL;

SET @dynamic\_command = NULL;

SET @dynamic\_command\_get\_tables = NULL;

SET @isHadrEnabled=0;

SET NOCOUNT ON;

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

--Set Parameter Values: You can change these (optional) -

--Note: The script has default parameters set -

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

--if set to 1: it will just generate a report with the index reorganization/rebuild statements

--if set to 0: it will reorganize or rebuild the fragmented indexes

SET @reportOnly = 0;

--optional: if not set (NULL), it will scann all databases

--If name is set (i.e. 'testDB') it will just scan the given database

SET @databaseToCheck = NULL;

--maintains only the indexes that have average fragmentation percentage equal or higher from the given value

SET @fragmentationThreshold = 5;

--fill factor - the percentage of the data page to be filled up with index data

SET @indexFillFactor = 90;

--sets the scanning mode for index statistics

--available values: 'DEFAULT', NULL, 'LIMITED', 'SAMPLED', or 'DETAILED'

SET @indexStatisticsScanningMode='DETAILED';

--if set to ON: sorts intermediate index results in TempDB

--if set to OFF: sorts intermediate index results in user database's log file

SET @sortInTempdb='ON';

--if set to 0: Does not output additional information about the index reorganization/rebuild process

--if set to 1: Outputs additional information about the index reorganization/rebuild process

SET @verboseMode = 1;

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

--End Parameter Values Setup -

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

-- check if given database exists and if compatibility level >= SQL 2005 (90)

IF @verboseMode=1

PRINT 'Checking if database '+@databaseToCheck+' exists and if compatibility level equals or greater 2005 (90)';

-- if given database does not exist, raise error with severity 20

-- in order to terminate script's execution

IF @databaseToCheck IS NOT NULL

BEGIN

DECLARE @checkResult INT

SET @checkResult=(SELECT COUNT(\*) FROM master.sys.databases WHERE [name]=RTRIM(@databaseToCheck));

IF @checkResult<1

RAISERROR('Error executing index reorganization/rebuild script: Database does not exist' , 20, 1) WITH LOG;

DECLARE @checkResult2 INT

SET @checkResult=(SELECT [compatibility\_level] FROM master.sys.databases WHERE [name]=RTRIM(@databaseToCheck));

IF @checkResult<90

RAISERROR('Error executing index reorganization/rebuild script: Only databases with SQL Server 2005 or later compatibility level are supported' , 20, 1) WITH LOG;

END

IF @verboseMode=1

PRINT 'Initial checks completed with no errors.';

-- Temporary table for storing index fragmentation details

IF OBJECT\_ID('tempdb..#tmpFragmentedIndexes') IS NULL

BEGIN

CREATE TABLE #tmpFragmentedIndexes

(

[dbName] sysname,

[tableName] sysname,

[schemaName] sysname,

[indexName] sysname,

[databaseID] SMALLINT ,

[objectID] INT ,

[indexID] INT ,

[AvgFragmentationPercentage] FLOAT,

[reorganizationOrRebuildCommand] NVARCHAR(MAX)

);

END

-- Initialize temporary table

DELETE FROM #tmpFragmentedIndexes;

-- Validate parameters/set defaults

IF @sortInTempdb NOT IN ('ON','OFF')

SET @sortInTempdb='ON';

-- Check if instance has AlwaysOn AGs enabled

SET @isHadrEnabled=CAST((SELECT ISNULL(SERVERPROPERTY('IsHadrEnabled'),0)) AS BIT);

-- if database not specified scan all databases

IF @databaseToCheck IS NULL

BEGIN

DECLARE dbNames\_cursor CURSOR

FOR

SELECT s.[name] AS dbName ,

s.database\_id

FROM master.sys.databases s

WHERE s.state\_desc = 'ONLINE'

AND s.is\_read\_only != 1

AND s.[name] NOT IN ( 'master', 'model', 'tempdb', 'MSDB' )

AND s.[compatibility\_level]>=90

ORDER BY s.database\_id;

END

ELSE

-- if database specified, scan only that database

BEGIN

DECLARE dbNames\_cursor CURSOR

FOR

SELECT s.[name] AS dbName ,

s.database\_id

FROM master.sys.databases s

WHERE s.state\_desc = 'ONLINE'

AND s.is\_read\_only != 1

AND s.[name]=RTRIM(@databaseToCheck)

END

-- if Always On Availability Groups are enabled, check for primary databases

-- (thus exclude secondary databases)

IF @isHadrEnabled=1

BEGIN

DEALLOCATE dbNames\_cursor;

-- if database not specified scan all databases

IF @databaseToCheck IS NULL

BEGIN

DECLARE dbNames\_cursor CURSOR

FOR

SELECT s.[name] AS dbName ,

s.database\_id

FROM master.sys.databases s

LEFT JOIN master.sys.dm\_hadr\_availability\_replica\_states r ON s.replica\_id = r.replica\_id

WHERE s.state\_desc = 'ONLINE'

AND s.is\_read\_only != 1

AND UPPER(ISNULL(r.role\_desc, 'NonHadrEnabled')) NOT LIKE 'SECONDARY'

AND s.[name] NOT IN ( 'master', 'model', 'tempdb' )

AND s.[compatibility\_level]>=90

ORDER BY s.database\_id;

END

ELSE

-- if database specified, scan only that database

BEGIN

DECLARE dbNames\_cursor CURSOR

FOR

SELECT s.[name] AS dbName ,

s.database\_id

FROM master.sys.databases s

LEFT JOIN master.sys.dm\_hadr\_availability\_replica\_states r ON s.replica\_id = r.replica\_id

WHERE s.state\_desc = 'ONLINE'

AND s.is\_read\_only != 1

AND UPPER(ISNULL(r.role\_desc, 'NonHadrEnabled')) NOT LIKE 'SECONDARY'

AND s.[name]=RTRIM(@databaseToCheck);

END

END

--

-- For each database included in the cursor,

-- gather all tables that have indexes with

-- average fragmentation percentage equal or above @fragmentationThreshold

--

OPEN dbNames\_cursor;

FETCH NEXT FROM dbNames\_cursor INTO @dbname, @dbid;

WHILE @@fetch\_status = 0

BEGIN

--If verbose mode is enabled, print logs

IF @verboseMode = 1

BEGIN

PRINT ''

PRINT 'Gathering index fragmentation statistics for database: ['+ @dbname + '] with id: ' + CAST(@dbid AS VARCHAR(10));

END;

SET @dynamic\_command\_get\_tables = N'

USE [' + @dbname+ N'];

INSERT INTO #tmpFragmentedIndexes (

[dbName],

[tableName],

[schemaName],

[indexName],

[databaseID],

[objectID],

[indexID],

[AvgFragmentationPercentage],

[reorganizationOrRebuildCommand]

)

SELECT

DB\_NAME() as [dbName],

tbl.name as [tableName],

SCHEMA\_NAME (tbl.schema\_id) as schemaName,

idx.Name as [indexName],

pst.database\_id as [databaseID],

pst.object\_id as [objectID],

pst.index\_id as [indexID],

pst.avg\_fragmentation\_in\_percent as [AvgFragmentationPercentage],

CASE WHEN pst.avg\_fragmentation\_in\_percent > 30 THEN

''ALTER INDEX [''+idx.Name+''] ON [''+DB\_NAME()+''].[''+SCHEMA\_NAME (tbl.schema\_id)+''].[''+tbl.name+''] REBUILD WITH (FILLFACTOR = '+@indexFillFactor+', SORT\_IN\_TEMPDB = '+@sortInTempdb+', STATISTICS\_NORECOMPUTE = OFF);''

WHEN pst.avg\_fragmentation\_in\_percent > 5 AND pst.avg\_fragmentation\_in\_percent <= 30 THEN

''ALTER INDEX [''+idx.Name+''] ON [''+DB\_NAME()+''].[''+SCHEMA\_NAME (tbl.schema\_id)+''].[''+tbl.name+''] REORGANIZE;''

ELSE

NULL

END

FROM sys.dm\_db\_index\_physical\_stats(DB\_ID(), NULL, NULL, NULL , '''+@indexStatisticsScanningMode+''') as pst

INNER JOIN sys.tables as tbl ON pst.object\_id = tbl.object\_id

INNER JOIN sys.indexes idx ON pst.object\_id = idx.object\_id AND pst.index\_id = idx.index\_id

WHERE pst.index\_id != 0

AND pst.alloc\_unit\_type\_desc IN ( N''IN\_ROW\_DATA'', N''ROW\_OVERFLOW\_DATA'')

AND pst.avg\_fragmentation\_in\_percent >= '+ @fragmentationThreshold + '

AND pst.page\_count >= 1000;'

-- if verbose mode is enabled, print logs

IF @verboseMode=1

BEGIN

PRINT 'Index fragmentation statistics script: ';

PRINT @dynamic\_command\_get\_tables;

END

-- gather index fragmentation statistics

EXEC (@dynamic\_command\_get\_tables);

-- bring next record from the cursor

FETCH NEXT FROM dbNames\_cursor INTO @dbname, @dbid;

END;

CLOSE dbNames\_cursor;

DEALLOCATE dbNames\_cursor;

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

-- if 'report only' mode is enabled

IF @reportOnly=1

BEGIN

SELECT dbName ,

tableName ,

schemaName ,

indexName ,

AvgFragmentationPercentage ,

reorganizationOrRebuildCommand

FROM #tmpFragmentedIndexes

ORDER BY AvgFragmentationPercentage DESC;

END

ELSE

-- if 'report only' mode is disabled, then execute

-- index reorganize/rebuild statements

BEGIN

DECLARE reorganizeOrRebuildCommands\_cursor CURSOR

FOR

SELECT reorganizationOrRebuildCommand

FROM #tmpFragmentedIndexes

WHERE reorganizationOrRebuildCommand IS NOT NULL

ORDER BY AvgFragmentationPercentage DESC;

OPEN reorganizeOrRebuildCommands\_cursor;

FETCH NEXT FROM reorganizeOrRebuildCommands\_cursor INTO @ReorganizeOrRebuildCommand;

WHILE @@fetch\_status = 0

BEGIN

IF @verboseMode = 1

BEGIN

PRINT ''

PRINT 'Executing script:'

PRINT @ReorganizeOrRebuildCommand

END

EXEC (@ReorganizeOrRebuildCommand);

FETCH NEXT FROM reorganizeOrRebuildCommands\_cursor INTO @ReorganizeOrRebuildCommand;

END;

CLOSE reorganizeOrRebuildCommands\_cursor;

DEALLOCATE reorganizeOrRebuildCommands\_cursor;

PRINT ''

PRINT 'All fragmented indexes have been reorganized/rebuilt.'

PRINT ''

END

END

ELSE

BEGIN

PRINT '';

PRINT 'Error: You need to be SysAdmin and use SQL Server 2005 or later in order to use this script.';

PRINT '';

END

--End of Script