DatAvail Script Inventory

Version 1.0

INDEX

[1. Backup Specific Scripts 1](#_Toc276153500)

[1.1. Backup Detail including Backup path, most recent backup date 1](#_Toc276153501)

[1.2. Full Database Backup history 1](#_Toc276153502)

[1.3. Log Backup history 1](#_Toc276153503)

[1.4. Backup all User Databases 1](#_Toc276153504)

[1.5. Backup Missing 3](#_Toc276153505)

[1.6. Script to analyze space required for database if backups are failing due to space issue. 3](#_Toc276153506)

[1.7. Script to Get Backup and Restore Status 4](#_Toc276153507)

[2. Restore Specific Scripts 5](#_Toc276153508)

[2.1. Script to Verify Restore 5](#_Toc276153509)

[2.2. Dynamic Script to restore transaction log file 5](#_Toc276153510)

[2.3. Restore Last Full, Differential, Log Backup 6](#_Toc276153511)

[2.4. Find out Estimated Time, Percentage, Elapsed time of a RESTORE DATABASE/BACKUP 7](#_Toc276153512)

[2.5. Before Restore a Cursor to Kill all user process 7](#_Toc276153513)

[3. SQL Jobs and DTS specific scripts 9](#_Toc276153514)

[3.1. SQL Server Job Details 9](#_Toc276153515)

[3.2. Get Job Schedule detail 9](#_Toc276153516)

[3.3. Get SQL Server Job owner 14](#_Toc276153517)

[3.4. How to change DTS package owner 14](#_Toc276153518)

[3.5. Script to find job in executing mode 15](#_Toc276153519)

[3.6. Find Hung Job 15](#_Toc276153520)

[3.7. Job Status 16](#_Toc276153521)

[4. User and Login Scripts 19](#_Toc276153522)

[4.1. Get all login details 19](#_Toc276153523)

[4.2. List of Users per Role 19](#_Toc276153524)

[4.3. List of special users per database 19](#_Toc276153525)

[4.4. Cursor to fix orphan users 19](#_Toc276153526)

[4.5. Script to Fix orphan User for all existing database 20](#_Toc276153527)

[4.6. Audit SQL Server Logins 21](#_Toc276153528)

[5. Database & Object specific Scripts 24](#_Toc276153529)

[5.1. Query to show free space in %, TotalSpace occupied in Data files 24](#_Toc276153530)

[5.2. Query to get database size, growth report 24](#_Toc276153531)

[5.3. Query to get Log size information 24](#_Toc276153532)

[5.4. Script to get table size information (row count, reserved space & used space) 25](#_Toc276153533)

[5.5. Script to get object property of table 25](#_Toc276153534)

[5.6. Script to Find Trigger status in SQL server 2000 25](#_Toc276153535)

[5.7. Script to get user permission on table 25](#_Toc276153536)

[5.8. Script to identify Hostname on SQL server is running on 26](#_Toc276153537)

[5.9. Permissions at Object Level 26](#_Toc276153538)

[5.10. Store Table Size periodically for Analysis 26](#_Toc276153539)

[5.11. Database Status 27](#_Toc276153540)

[5.12. Delete duplicate records from table 28](#_Toc276153541)

[5.13. Generate INSERT statements 29](#_Toc276153542)

[5.14. Database Status 37](#_Toc276153543)

[5.15. Generate PK Script 38](#_Toc276153544)

[5.16. Generate FK Script 39](#_Toc276153545)

[6. Index Scripts 40](#_Toc276153546)

[6.1. Script to find Missing Indexes from other database server 40](#_Toc276153547)

[6.2. Index Rebuild script for SQL Server 2005 41](#_Toc276153548)

[6.3. Index Rebuild script for SQL Server 2000 & 2005 43](#_Toc276153549)

[6.4. DUPLICATE INDEX SCRIPT FOR 2000/2005 47](#_Toc276153550)

[6.5. Script to get index fragmentation and last update stats status 50](#_Toc276153551)

[6.6. Script to move indexes on different file group 51](#_Toc276153552)

[6.7. Script to find unused Indexes 51](#_Toc276153553)

[6.8. Drop Hypothetical Index 52](#_Toc276153554)

[6.9. Update Statistics on all databases 52](#_Toc276153555)

[7. Wait Stats 54](#_Toc276153556)

[7.1. Script to collect wait stats 54](#_Toc276153557)

[7.2. Script to collect wait stats with delay 54](#_Toc276153558)

[8. Full Text Scripts 56](#_Toc276153559)

[8.1. Full text Catalog Information Query 56](#_Toc276153560)

[8.2. Full Text Catalog Status and Catalog Item count comparison with table count 56](#_Toc276153561)

[9. Performance Scripts 58](#_Toc276153562)

[9.1. Get actual query currently running from sysprocess 58](#_Toc276153563)

[9.2. Shows what individual SQL statements are currently executing 58](#_Toc276153564)

[9.3. SPID waiting for resource to be allocated 58](#_Toc276153565)

[9.4. Get Offending queries 59](#_Toc276153566)

[9.5. Missing or poorly formed indexes 60](#_Toc276153567)

[9.6. Largest IO queries 60](#_Toc276153568)

[9.7. Query plan reuse and DMVs 60](#_Toc276153569)

[9.8. Get top SQL Statement Elapsed Time Stats for top XX Elapsed Time Plans 61](#_Toc276153570)

[9.9. Get TOP SQL Statement CPU Stats for top XX CPU Plans 61](#_Toc276153571)

[9.10. Database Properties (HTML) 62](#_Toc276153572)

[9.11. Unindexed Foreign Keys 67](#_Toc276153573)

[9.12. Non Primary Key Tables 68](#_Toc276153574)

[9.13. Blocking Script 68](#_Toc276153575)

[9.14. Long Running processes 76](#_Toc276153576)

[10. Trace Scripts 78](#_Toc276153577)

[10.1. Script to get bad query with no. of frequency from trace file 78](#_Toc276153578)

[10.2. Query to Load worse offending queries from trace file 78](#_Toc276153579)

[10.3. Get TOP SQL from Trace File 78](#_Toc276153580)

[10.4. Deadlock Trace Analysis 79](#_Toc276153581)

[from ::fn\_trace\_gettable('trace\_file\_path',default) where textdata like '%deadlock-list%' and starttime > ‘what-ever’ 79](#_Toc276153582)

[11. Generic Script-out Object Query 82](#_Toc276153583)

[12. OS related Scripts 90](#_Toc276153584)

[12.1. Get drive space information 90](#_Toc276153585)

[12.2. Directory size with file count 91](#_Toc276153586)

[13. Replication and Mirroring Scripts 92](#_Toc276153587)

[13.1. Command for adding ‘NOT For Replication' for Identity -- without breaking replication 92](#_Toc276153588)

[13.2. Monitor database mirroring 92](#_Toc276153589)

[13.3. Replication synchronization monitoring script 94](#_Toc276153590)

[13.4. Merge Replication Status 97](#_Toc276153591)

[13.5. Start Distribution Agent 98](#_Toc276153592)

[13.6. Start Log Reader Agent 98](#_Toc276153593)

[14. Other Scripts 100](#_Toc276153594)

[14.1. Script to get Server up time 100](#_Toc276153595)

[14.2. Script to get comma Separated values in Row 100](#_Toc276153596)

[14.3. Shrink Data\Log file in chunk with Stop Time Paramaeter 100](#_Toc276153597)

[14.4. Shrink File in Chunk 102](#_Toc276153598)

[14.5. Get SQL Port 103](#_Toc276153599)

[14.6. SQL Server Property Information 103](#_Toc276153600)

[14.7. Script out Triggers using DMO (VB Script) 105](#_Toc276153601)

[14.8. Script out SQL Jobs using DMO (VB Script) 105](#_Toc276153602)

[14.9. Script out Indexes using DMO (VB Script) 106](#_Toc276153603)

[15. HealthCheck Script 106](#_Toc276153604)

[106](#_Toc276153605)

[16. Centralize Monitoring script 106](#_Toc276153606)

[16.1. Script to monitor OPEN RDP SESSION 106](#_Toc276153607)

[16.2. Collector Service Status Report 108](#_Toc276153608)

[108](#_Toc276153609)

[16.3. Sql Server Service Status Report 108](#_Toc276153610)

[16.4. Sql Server Mirroring Status Report 108](#_Toc276153611)

[16.5. Sql Server Job Status Report 108](#_Toc276153612)

# Backup Specific Scripts

## Backup Detail including Backup path, most recent backup date

DECLARE @type char(1)

SET @TYPE='D' --FULL Backup 'D' , 'L' for Log Backup  
  
Select ServerName=@@servername ,

Edition=Serverproperty('Edition'),

ProductLevel=Serverproperty('ProductLevel'),

ServerVersion=Serverproperty('Productversion'),DatabaseName =sd.name,

DBStatus=Databasepropertyex(sd.name,'Status'),

RecoveryModel =Databasepropertyex(sd.name,'Recovery'),

IsBackupDone =Case when Bkp.database\_name is Null then 'No' Else 'Yes' End,

Bkp.physical\_device\_name,Bkp.backup\_finish\_date

from sysdatabases sd

Left Join

(

select

x.database\_name,

z.physical\_device\_name, --CONVERT(char(20), x.backup\_finish\_date, 108) FinishTime,

x.backup\_finish\_date from msdb.dbo.backupset x

JOIN ( SELECT a.database\_name, max(a.backup\_finish\_date) backup\_finish\_date

FROM msdb.dbo.backupset a WHERE type = @TYPE

GROUP BY a.database\_name ) y

ON x.database\_name = y.database\_name

AND x.backup\_finish\_date = y.backup\_finish\_date

JOIN msdb.dbo.backupmediafamily z

ON x.media\_set\_id = z.media\_set\_id

)BKP

ON sd.name=Bkp.database\_name

where sd.name <> 'tempdb'

## Full Database Backup history

SELECT x.database\_name, z.physical\_device\_name,

CONVERT(char(20), x.backup\_finish\_date, 108) FinishTime, x.backup\_finish\_date

from msdb.dbo.backupset x

join ( SELECT a.database\_name, max(a.backup\_finish\_date) backup\_finish\_date

FROM msdb.dbo.backupset a

WHERE type = 'D'

GROUP BY a.database\_name ) y on x.database\_name = y.database\_name

and x.backup\_finish\_date = y.backup\_finish\_date

JOIN msdb.dbo.backupmediafamily z ON x.media\_set\_id = z.media\_set\_id

order by x.backup\_finish\_date desc

## Log Backup history

SELECT x.database\_name, z.physical\_device\_name,

CONVERT(char(20), x.backup\_finish\_date, 108) FinishTime, x.backup\_finish\_date

from msdb.dbo.backupset x

join ( SELECT a.database\_name, max(a.backup\_finish\_date) backup\_finish\_date

FROM msdb.dbo.backupset a

WHERE type = 'L'

GROUP BY a.database\_name ) y on x.database\_name = y.database\_name

and x.backup\_finish\_date = y.backup\_finish\_date

JOIN msdb.dbo.backupmediafamily z ON x.media\_set\_id = z.media\_set\_id

order by x.backup\_finish\_date desc

## Backup all User Databases

DECLARE @DB VARCHAR(300), @Dir VARCHAR(300), @Stmt VARCHAR(1000), @RM sql\_variant,

@Status INT, @DT DateTime, @BakTime varchar(20)

SELECT @DB = '', @Dir = '\\NetworkPath'

SET @DT = GETDATE()

SET @BakTime = dbo.fn\_StringFormatDate(@DT)

IF OBJECT\_ID('tempdb.dbo.#Databases') IS NOT NULL DROP TABLE #Databases

IF OBJECT\_ID('tempdb.dbo.BackUpStatus') IS NULL

CREATE TABLE tempdb.dbo.BackUpStatus

(Row\_Id int IDENTITY NOT NULL PRIMARY KEY,

dbName varchar (50) NULL,

StepName varchar (100) NOT NULL,

Status int NULL,

ActionDateTime datetime NOT NULL)

ELSE

DELETE FROM tempdb.dbo.BackUpStatus WITH (ROWLOCK)

WHERE ActionDateTime < DATEADD(d, -90, GETDATE())

--SELECT @Stmt = 'MKDIR ' + @Dir

--EXEC @Status = master..xp\_cmdshell @Stmt, NO\_OUTPUT

SELECT @Stmt = @Dir + '\' + @DB + ''

--EXEC @Status = master.dbo.xp\_create\_subdir @Stmt

INSERT tempdb.dbo.BackUpStatus (dbName, StepName, Status, ActionDateTime)

SELECT 'Job Requirement', 'Check Backup Directory' + @Dir, @Status, GETDATE()

SELECT Name as dbName, DATABASEPROPERTYEX(name, 'Recovery') RecoveryModel

INTO #Databases

FROM master..SysDatabases WITH (NOLOCK)

WHERE DATABASEPROPERTYEX(name, 'Status') <> 'OFFLINE' AND

DATABASEPROPERTYEX(name, 'Status') <> 'LOADING' AND

DATABASEPROPERTYEX(name, 'Status') <> 'Restoring' AND

DATABASEPROPERTYEX(name, 'Recovery') = 'FULL' AND

Name NOT IN ('master', 'distribution', 'msdb', 'model') AND

(Name NOT LIKE 'a%' AND

Name NOT LIKE 'HMXMeta%' AND

Name NOT LIKE 'xxx%' AND

Name NOT LIKE 'g%' AND

Name NOT LIKE '%temp%' AND

Name NOT LIKE '%tmp%' AND

Name NOT LIKE '%test%' AND

Name NOT LIKE '%train%' AND

Name NOT LIKE '%Secure%' AND

Name NOT LIKE '%LiteSpeedLocal%' OR

Name LIKE '%WCMeta%')

-- AND NAME LIKE @DBname

WHILE 1 = 1 BEGIN

SELECT TOP 1 @DB = ltrim(rtrim(dbName)), @RM = RecoveryModel

FROM #Databases

WHERE dbName > @DB

ORDER BY dbName

IF @@ROWCOUNT <> 1 OR @DB IS NULL BREAK

SET @Stmt = 'master..xp\_cmdshell ''' + 'MKDIR ' + @Dir + '\' + @DB + ''', NO\_OUTPUT'

EXEC(@Stmt)

--SELECT @Stmt = @Dir + '\' + @DB + ''

--EXEC @Status = master.dbo.xp\_create\_subdir @Stmt

SET @Stmt = 'BACKUP DATABASE ['+ @DB + '] ' +

'TO DISK=''' + @Dir + '\' + @DB + '\' + @DB + @BakTime + '\_FULL.BAK'',' +

'@INIT=0'

PRINT @Stmt

--EXEC (@Stmt)

SELECT @Status = @@ERROR

INSERT tempdb.dbo.BackUpStatus (dbName, StepName, Status, ActionDateTime)

SELECT @DB, 'Backup Log ', @Status, GETDATE()

END

## Backup Missing

DECLARE @threshold INT

SET @threshold = 32 -- Hours

Select @@servername AS InstanceName

, d.name AS Database\_Name

, 'Backup did not occurd within ' + CAST(@threshold as VARCHAR) + ' hours.' AS Message

, a.backup\_start\_date AS Last\_Backup

, a.backup\_size

, CASE b.device\_type

when 2 then 'Disk - Temporary'

when 102 then 'Disk - Permanent'

when 5 then 'Tape - Temporary'

when 105 then 'Tape - Permanent'

when 7 then 'Network'

else 'Unknown' END AS Device\_Type

, b.physical\_device\_name AS Physical\_Path

from(

select a.media\_set\_id, a.database\_name, a.backup\_start\_date, a.backup\_size,b.type

from msdb.dbo.backupset a

JOIN (

Select database\_name, max(backup\_finish\_date) backup\_finish\_date, type

from msdb.dbo.backupset

where type='D'

group by database\_name,type

) b

on a.database\_name = b.database\_name

and a.backup\_finish\_date = b.backup\_finish\_date

) a

JOIN msdb..backupmediafamily b ON a.media\_set\_id = b.media\_set\_id

right join master..sysdatabases d on a.database\_name=d.name

where d.name<>'tempdb'

and (DATEDIFF(hh, a.backup\_start\_date, getdate()) > @threshold OR a.backup\_start\_date IS NULL)

order by d.name

## Script to analyze space required for database if backups are failing due to space issue.

Select sum( cast((a.size \* 8.00) / 1024 as numeric(12,2))) as DB\_Size\_in\_MB

from sysaltfiles a

join sysdatabases b on a.dbid = b.dbid

where DATABASEPROPERTYEX(b.name, 'status') = 'ONLINE'

and b.name in (SELECT a.name

FROM master..sysdatabases a

LEFT JOIN (SELECT database\_name, MAX(backup\_finish\_date) backup\_finish\_date

           FROM msdb..backupset

           WHERE Type = 'D'

           GROUP BY database\_name) b ON a.name = b.database\_name

WHERE (DATEDIFF(hh, b.backup\_finish\_date, GETDATE()) > 10 or b.database\_name IS NULL) AND

       (DATABASEPROPERTYEX(a.name, 'Status') = 'ONLINE' AND

        a.Name NOT LIKE '%temp%' AND a.Name NOT LIKE '%tmp%' AND a.Name NOT LIKE '%test%' AND

        a.Name NOT LIKE '%train%'  AND a.Name NOT LIKE '%LiteSpeedLocal%')

)

Group By b.name

Also if native backup is done, you may get the result using backupset table,but for compressed backup files result may differ.

SELECT a.name  ,b.backup\_size

FROM master..sysdatabases a

LEFT JOIN (SELECT database\_name,max(backup\_size)/1024/1024 as backup\_size, MAX(backup\_finish\_date) backup\_finish\_date

           FROM msdb..backupset

           WHERE Type = 'D'

           GROUP BY database\_name) b ON a.name = b.database\_name

WHERE (DATEDIFF(hh, b.backup\_finish\_date, GETDATE()) > 10 or b.database\_name IS NULL) AND

       (DATABASEPROPERTYEX(a.name, 'Status') = 'ONLINE' AND

        --a.Name NOT IN ('master', 'distribution', 'msdb', 'model') AND

        a.Name NOT LIKE '%temp%' AND a.Name NOT LIKE '%tmp%' AND a.Name NOT LIKE '%test%' AND

        a.Name NOT LIKE '%train%'  AND a.Name NOT LIKE '%LiteSpeedLocal%')

## Script to Get Backup and Restore Status

SELECT command,

            s.text,

            start\_time,

            percent\_complete,

            CAST(((DATEDIFF(s,start\_time,GetDate()))/3600) as varchar) + ' hour(s), '

                  + CAST((DATEDIFF(s,start\_time,GetDate())%3600)/60 as varchar) + 'min, '

                  + CAST((DATEDIFF(s,start\_time,GetDate())%60) as varchar) + ' sec' as running\_time,

            CAST((estimated\_completion\_time/3600000) as varchar) + ' hour(s), '

                  + CAST((estimated\_completion\_time %3600000)/60000 as varchar) + 'min, '

                  + CAST((estimated\_completion\_time %60000)/1000 as varchar) + ' sec' as est\_time\_to\_go,

            dateadd(second,estimated\_completion\_time/1000, getdate()) as est\_completion\_time

FROM sys.dm\_exec\_requests r

CROSS APPLY sys.dm\_exec\_sql\_text(r.sql\_handle) s

WHERE r.command in ('RESTORE DATABASE', 'BACKUP DATABASE', 'RESTORE LOG', 'BACKUP LOG')

# Restore Specific Scripts

## Script to Verify Restore

Script to verify restore is done using correct backup; output contains destination database name, date restored and physical device used for restore.

Declare @dbname SYSNAME

SET @dbname = 'DatabaseName'

SELECT top 1

destination\_database\_name as 'Destination Database',

restore\_date as 'Date Restored',

d.physical\_device\_name

FROM msdb..restorehistory a

join msdb..backupset b on a.backup\_set\_id = b.backup\_set\_id

join msdb..backupmediaset c on c.media\_set\_id = b.media\_set\_id

join msdb..backupmediafamily d on d.media\_set\_id = c.media\_set\_id

where destination\_database\_name = CASE  WHEN @dbname IS NOT NULL THEN @dbname

ELSE destination\_database\_name END

order by restore\_date desc

## Dynamic Script to restore transaction log file

Dynamic script to restore transaction log file. Script will generate and script to restore database till last restore backup done.

DECLARE @databaseName sysname

DECLARE @backupStartDate datetime

DECLARE @backup\_set\_id\_start INT

DECLARE @backup\_set\_id\_end INT

-- set database to be used

SET @databaseName = 'enterDatabaseNameHere'

SELECT @backup\_set\_id\_start = MAX(backup\_set\_id)

FROM msdb.dbo.backupset

WHERE database\_name = @databaseName AND type = 'D'

SELECT @backup\_set\_id\_end = MIN(backup\_set\_id)

FROM msdb.dbo.backupset

WHERE database\_name = @databaseName AND type = 'D'

AND backup\_set\_id > @backup\_set\_id\_start

IF @backup\_set\_id\_end IS NULL SET @backup\_set\_id\_end = 999999999

SELECT backup\_set\_id, 'RESTORE DATABASE ' + @databaseName + ' FROM DISK =

'''

+ mf.physical\_device\_name + ''' WITH NORECOVERY'

FROM msdb.dbo.backupset b,

msdb.dbo.backupmediafamily mf

WHERE b.media\_set\_id = mf.media\_set\_id

AND b.database\_name = @databaseName

AND b.backup\_set\_id = @backup\_set\_id\_start

UNION

SELECT backup\_set\_id, 'RESTORE LOG ' + @databaseName + ' FROM DISK = '''

+ mf.physical\_device\_name + ''' WITH NORECOVERY'

FROM msdb.dbo.backupset b,

msdb.dbo.backupmediafamily mf

WHERE b.media\_set\_id = mf.media\_set\_id

AND b.database\_name = @databaseName

AND b.backup\_set\_id >= @backup\_set\_id\_start AND b.backup\_set\_id <

@backup\_set\_id\_end

AND b.type = 'L'

UNION

SELECT 999999999 AS backup\_set\_id, 'RESTORE DATABASE ' + @databaseName + '

WITH RECOVERY'

ORDER BY backup\_set\_id

## Restore Last Full, Differential, Log Backup

create proc usp\_RestoreAllBackups

@DBName varchar(50),

@Days int = 15

as

/\*

This script creates the script to restore your database with the information existing in [msdb] database.

It helps you by finding the last FULL backup, the last DIFFERENTIAL backup and all the TRANSACTION LOG backups needed.

It's quite comfortable when you are doing so many differential or log backups.

The variable @DBName should be set to the name of the database you want to query on.

It is not case sensitive unless your collation is.

The variable @Days should be set to how many days back in the records you want to list backups for.

By default set to 14 (old enought I think)

\*/

-- Important because we're going to 'print' the sql code for the restore

SET NOCOUNT ON

CREATE TABLE #BackupsHistory

(

id INT IDENTITY(1,1),

backup\_start\_date DATETIME,

backup\_type CHAR(1),

physical\_device\_name VARCHAR(2000)

)

INSERT INTO #BackupsHistory (backup\_start\_date, backup\_type, physical\_device\_name)

SELECT S.backup\_start\_date, S.type, M.physical\_device\_name

FROM msdb..backupset S

JOIN msdb..backupmediafamily M ON M.media\_set\_id = S.media\_set\_id

WHERE S.database\_name = @DBName

AND DATEDIFF(DAY,S.backup\_start\_date,GETDATE()) < @Days

ORDER by backup\_start\_date

DECLARE @lastFullBackup INT, @lastFullBackupPath VARCHAR(2000),

@lastDifferentialBackup INT, @lastDifferentialBackupPath VARCHAR(2000)

-- We get the last Full backup done. That where we are going to start the restore process

SET @lastFullBackup = (SELECT TOP 1 id FROM #BackupsHistory WHERE backup\_type='D' ORDER BY backup\_start\_date DESC)

SET @lastFullBackupPath = (SELECT physical\_device\_name FROM #BackupsHistory WHERE id=@lastFullBackup)

-- Restoring the Full backup

PRINT 'RESTORE DATABASE ' + @DBName

PRINT 'FROM DISK=''' + @lastFullBackupPath + ''''

-- IF it's there's no backup (differential or log) after it, we set to recovery

IF (@lastFullBackup = (SELECT MAX(id) FROM #BackupsHistory))

PRINT 'WITH RECOVERY'

ELSE PRINT 'WITH NORECOVERY'

PRINT 'GO'

PRINT ''

-- We get the last Differential backup (it must be done after the last Full backup)

SET @lastDifferentialBackup = (SELECT TOP 1 id FROM #BackupsHistory WHERE backup\_type='I' AND id>@lastFullBackup ORDER BY backup\_start\_date DESC)

SET @lastDifferentialBackupPath = (SELECT physical\_device\_name FROM #BackupsHistory WHERE id=@lastDifferentialBackup)

-- IF there's a differential backup done after the full backup we script it

IF (@lastDifferentialBackup IS NOT NULL)

BEGIN

-- Restoring the Full backup

PRINT 'RESTORE DATABASE ' + @DBName

PRINT 'FROM DISK=''' + @lastDifferentialBackupPath + ''''

-- IF it's there's no backup (differential or log) after it, we set to recovery

IF (@lastDifferentialBackup = (SELECT MAX(id) FROM #BackupsHistory))

PRINT 'WITH RECOVERY'

ELSE PRINT 'WITH NORECOVERY'

PRINT 'GO'

PRINT ''

END

-- For TRANSACTION LOGs

DECLARE @i INT, @logBackupPath VARCHAR(2000)

IF (@lastDifferentialBackup IS NULL)

SET @i = @lastFullBackup + 1

ELSE SET @i = @lastDifferentialBackup + 1

-- Here whe are scripting the restores for the necessary logs

WHILE (@i <= (SELECT MAX(id) FROM #BackupsHistory))

BEGIN

SET @logBackupPath = (SELECT physical\_device\_name FROM #BackupsHistory WHERE id=@i)

PRINT 'RESTORE LOG ' + @DBName

PRINT 'FROM DISK=''' + @logBackupPath + ''''

-- IF it's the last transaction log, we'll say it to recover

IF (@i = (SELECT MAX(id) FROM #BackupsHistory))

PRINT 'WITH RECOVERY'

ELSE PRINT 'WITH NORECOVERY'

PRINT 'GO'

PRINT ''

SET @i = @i + 1

END

DROP TABLE #BackupsHistory

## Find out Estimated Time, Percentage, Elapsed time of a RESTORE DATABASE/BACKUP

SELECT r.session\_id ,r.command,CONVERT(NUMERIC(6,2),r.percent\_complete)AS [Percent Complete],CONVERT(VARCHAR(20),DATEADD(ms,r.estimated\_completion\_time,GetDate()),20) AS [ETA Completion Time],CONVERT(NUMERIC(10,2),r.total\_elapsed\_time/1000.0/60.0) AS [Elapsed Min],CONVERT(NUMERIC(10,2),r.estimated\_completion\_time/1000.0/60.0) AS [ETA Min],CONVERT(NUMERIC(10,2),r.estimated\_completion\_time/1000.0/60.0/60.0) AS [ETA Hours]

,CONVERT(VARCHAR(1000),(SELECT SUBSTRING(text,r.statement\_start\_offset/2

,CASE WHEN r.statement\_end\_offset = -1 THEN 1000 ELSE (r.statement\_end\_offset-r.statement\_start\_offset)/2 END)

FROM sys.dm\_exec\_sql\_text(sql\_handle)))FROM sys.dm\_exec\_requests r

WHERE command IN ('RESTORE DATABASE','BACKUP DATABASE')

## Before Restore a Cursor to Kill all user process

CREATE TABLE #TmpWho

(spid INT, ecid INT, status VARCHAR(150), loginame VARCHAR(150),

hostname VARCHAR(150), blk INT, dbname VARCHAR(150), cmd VARCHAR(150))

INSERT INTO #TmpWho

EXEC sp\_who

DECLARE @spid INT

DECLARE @tString varchar(15)

DECLARE @getspid CURSOR

SET @getspid = CURSOR FOR

SELECT spid

FROM #TmpWho

WHERE dbname = 'mydb'OPEN @getspid

FETCH NEXT FROM @getspid INTO @spid

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @tString = 'KILL ' + CAST(@spid AS VARCHAR(5))

EXEC(@tString)

FETCH NEXT FROM @getspid INTO @spid

END

CLOSE @getspid

DEALLOCATE @getspid

DROP TABLE #TmpWho

# SQL Jobs and DTS specific scripts

## SQL Server Job Details

Exec msdb..sp\_help\_job

## Get Job Schedule detail

Step1 : Create 3 Function fn\_freq\_interval\_desc , fn\_Date2Str, fn\_Date2Str

**Step2** : For SQL 2000 Job run the SQL code wriiten after the Function

**Step3** : For SQL 2005 Job run the SQL code wriiten after the SQL 2000

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

Step1 : Function : fn\_freq\_interval\_desc , fn\_Date2Str, fn\_Date2Str

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

USE master

GO

CREATE FUNCTION fn\_freq\_interval\_desc(@freq\_interval INT)

RETURNS VARCHAR(1000)

AS

BEGIN

DECLARE @result VARCHAR(1000)

SET @result = ''

IF (@freq\_interval & 1 = 1)

SET @result = 'Sunday, '

IF (@freq\_interval & 2 = 2)

SET @result = @result + 'Monday, '

IF (@freq\_interval & 4 = 4)

SET @result = @result + 'Tuesday, '

IF (@freq\_interval & 8 = 8)

SET @result = @result + 'Wednesday, '

IF (@freq\_interval & 16 = 16)

SET @result = @result + 'Thursday, '

IF (@freq\_interval & 32 = 32)

SET @result = @result + 'Friday, '

IF (@freq\_interval & 64 = 64)

SET @result = @result + 'Saturday, '

RETURN(LEFT(@result,LEN(@result)-1))

END

GO

CREATE FUNCTION fn\_Time2Str(@time INT)

RETURNS VARCHAR(10)

AS

BEGIN

DECLARE @strtime CHAR(6)

SET @strtime = RIGHT('000000' + CONVERT(VARCHAR,@time),6)

RETURN LEFT(@strtime,2) + ':' + SUBSTRING(@strtime,3,2) + ':' + RIGHT(@strtime,2)

END

GO

CREATE FUNCTION fn\_Date2Str(@date INT)

RETURNS VARCHAR(10)

AS

BEGIN

DECLARE @strdate CHAR(8)

SET @strdate = LEFT(CONVERT(VARCHAR,@date) + '00000000', 8)

RETURN RIGHT(@strdate,2) + '/' + SUBSTRING(@strdate,5,2) + '/' + LEFT(@strdate,4)

END

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

**Step2** : SQL SERVER 2000

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

/\*

Created by Solihin Ho - http://solihinho.wordpress.com

Usage : Change the value of variable @Filter

'Y' --> display only enabled job

'N' --> display only disabled job

'A' --> display all job

'X' --> display job which is duration already end

\*/

DECLARE @Filter CHAR(1)

SET @Filter = 'A'

DECLARE @sql VARCHAR(8000)

DECLARE @is\_sysadmin INT

DECLARE @job\_owner sysname

IF OBJECT\_ID('tempdb..#xp\_results') IS NOT NULL

BEGIN

DROP TABLE #xp\_results

END

CREATE TABLE #xp\_results (

job\_id UNIQUEIDENTIFIER NOT NULL,

last\_run\_date INT NOT NULL,

last\_run\_time INT NOT NULL,

next\_run\_date INT NOT NULL,

next\_run\_time INT NOT NULL,

next\_run\_schedule\_id INT NOT NULL,

requested\_to\_run INT NOT NULL,

request\_source INT NOT NULL,

request\_source\_id sysname COLLATE database\_default NULL,

running INT NOT NULL,

current\_step INT NOT NULL,

current\_retry\_attempt INT NOT NULL,

job\_state INT NOT NULL

)

SELECT @is\_sysadmin = ISNULL(IS\_SRVROLEMEMBER(N'sysadmin'), 0)

SELECT @job\_owner = SUSER\_SNAME()

INSERT INTO #xp\_results

EXECUTE master.dbo.xp\_sqlagent\_enum\_jobs @is\_sysadmin, @job\_owner

SET @sql = '

SELECT

j.Name AS JobName

, c.Name AS Category

, CASE j.enabled WHEN 1 THEN ''Yes'' else ''No'' END as Enabled

, CASE s.enabled WHEN 1 THEN ''Yes'' else ''No'' END as Scheduled

, j.Description

, CASE s.freq\_type

WHEN 1 THEN ''Once''

WHEN 4 THEN ''Daily''

WHEN 8 THEN ''Weekly''

WHEN 16 THEN ''Monthly''

WHEN 32 THEN ''Monthly relative''

WHEN 64 THEN ''When SQL Server Agent starts''

WHEN 128 THEN ''Start whenever the CPU(s) become idle'' END as Occurs

, CASE s.freq\_type

WHEN 1 THEN ''O''

WHEN 4 THEN ''Every ''

+ convert(varchar,s.freq\_interval)

+ '' day(s)''

WHEN 8 THEN ''Every ''

+ convert(varchar,s.freq\_recurrence\_factor)

+ '' weeks(s) on ''

+ master.dbo.fn\_freq\_interval\_desc(s.freq\_interval)

WHEN 16 THEN ''Day '' + convert(varchar,s.freq\_interval)

+ '' of every ''

+ convert(varchar,s.freq\_recurrence\_factor)

+ '' month(s)''

WHEN 32 THEN ''The ''

+ CASE s.freq\_relative\_interval

WHEN 1 THEN ''First''

WHEN 2 THEN ''Second''

WHEN 4 THEN ''Third''

WHEN 8 THEN ''Fourth''

WHEN 16 THEN ''Last'' END

+ CASE s.freq\_interval

WHEN 1 THEN '' Sunday''

WHEN 2 THEN '' Monday''

WHEN 3 THEN '' Tuesday''

WHEN 4 THEN '' Wednesday''

WHEN 5 THEN '' Thursday''

WHEN 6 THEN '' Friday''

WHEN 7 THEN '' Saturday''

WHEN 8 THEN '' Day''

WHEN 9 THEN '' Weekday''

WHEN 10 THEN '' Weekend Day'' END

+ '' of every ''

+ convert(varchar,s.freq\_recurrence\_factor)

+ '' month(s)'' END AS Occurs\_detail

, CASE s.freq\_subday\_type

WHEN 1 THEN ''Occurs once at ''

+ master.dbo.fn\_Time2Str(s.active\_start\_time)

WHEN 2 THEN ''Occurs every ''

+ convert(varchar,s.freq\_subday\_interval)

+ '' Seconds(s) Starting at ''

+ master.dbo.fn\_Time2Str(s.active\_start\_time)

+ '' ending at ''

+ master.dbo.fn\_Time2Str(s.active\_end\_time)

WHEN 4 THEN ''Occurs every ''

+ convert(varchar,s.freq\_subday\_interval)

+ '' Minute(s) Starting at ''

+ master.dbo.fn\_Time2Str(s.active\_start\_time)

+ '' ending at ''

+ master.dbo.fn\_Time2Str(s.active\_end\_time)

WHEN 8 THEN ''Occurs every ''

+ convert(varchar,s.freq\_subday\_interval)

+ '' Hour(s) Starting at ''

+ master.dbo.fn\_Time2Str(s.active\_start\_time)

+ '' ending at ''

+ master.dbo.fn\_Time2Str(s.active\_end\_time) END AS Frequency

, CASE WHEN s.freq\_type = 1 THEN ''On date: ''

+ master.dbo.fn\_Date2Str(active\_start\_date)

+ '' At time: ''

+ master.dbo.fn\_Time2Str(s.active\_start\_time)

WHEN s.freq\_type < 64 THEN ''Start date: ''

+ master.dbo.fn\_Date2Str(s.active\_start\_date)

+ '' end date: ''

+ master.dbo.fn\_Date2Str(s.active\_end\_date) END as Duration

, master.dbo.fn\_Date2Str(xp.next\_run\_date) + '' ''

+ master.dbo.fn\_Time2Str(xp.next\_run\_time) AS Next\_Run\_Date

FROM msdb.dbo.sysjobs j (NOLOCK)

INNER JOIN msdb.dbo.sysjobschedules s (nolock) ON j.job\_id = s.job\_id

INNER JOIN msdb.dbo.syscategories c (NOLOCK) ON j.category\_id = c.category\_id

INNER JOIN #xp\_results xp (NOLOCK) ON j.job\_id = xp.job\_id

WHERE 1 = 1

@Filter

ORDER BY j.name'

IF @Filter = 'Y'

SET @sql = REPLACE(@sql,'@Filter',' AND j.enabled = 1 ')

ELSE

IF @Filter = 'N'

SET @sql = REPLACE(@sql,'@Filter',' AND j.enabled = 0 ')

ELSE

IF @Filter = 'X'

SET @sql = REPLACE(@sql,'@Filter',

'AND s.active\_end\_date < convert(varchar(8),GetDate(),112) ')

ELSE

SET @sql = REPLACE(@sql,'@Filter','')

EXEC(@sql)

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

**Step3** : SQL SERVER 2005

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

/\*

Created by Solihin Ho - http://solihinho.wordpress.com

Usage : Change the value of variable @Filter

'Y' --> display only enabled job

'N' --> display only disabled job

'A' --> display all job

'X' --> display job which is duration already end

\*/

DECLARE @Filter CHAR(1)

SET @Filter = 'A'

DECLARE @sql VARCHAR(8000)

DECLARE @is\_sysadmin INT

DECLARE @job\_owner sysname

IF OBJECT\_ID('tempdb..#xp\_results') IS NOT NULL

BEGIN

DROP TABLE #xp\_results

END

CREATE TABLE #xp\_results (

job\_id UNIQUEIDENTIFIER NOT NULL,

last\_run\_date INT NOT NULL,

last\_run\_time INT NOT NULL,

next\_run\_date INT NOT NULL,

next\_run\_time INT NOT NULL,

next\_run\_schedule\_id INT NOT NULL,

requested\_to\_run INT NOT NULL,

request\_source INT NOT NULL,

request\_source\_id sysname COLLATE database\_default NULL,

running INT NOT NULL,

current\_step INT NOT NULL,

current\_retry\_attempt INT NOT NULL,

job\_state INT NOT NULL)

SELECT @is\_sysadmin = ISNULL(IS\_SRVROLEMEMBER(N'sysadmin'), 0)

SELECT @job\_owner = SUSER\_SNAME()

INSERT INTO #xp\_results

EXECUTE master.dbo.xp\_sqlagent\_enum\_jobs @is\_sysadmin, @job\_owner

SET @sql = '

SELECT

j.Name AS JobName

, c.Name AS Category

, CASE j.enabled WHEN 1 THEN ''Yes'' else ''No'' END as Enabled

, CASE s.enabled WHEN 1 THEN ''Yes'' else ''No'' END as Scheduled

, j.Description

, CASE s.freq\_type

WHEN 1 THEN ''Once''

WHEN 4 THEN ''Daily''

WHEN 8 THEN ''Weekly''

WHEN 16 THEN ''Monthly''

WHEN 32 THEN ''Monthly relative''

WHEN 64 THEN ''When SQL Server Agent starts''

WHEN 128 THEN ''Start whenever the CPU(s) become idle'' END as Occurs

, CASE s.freq\_type

WHEN 1 THEN ''O''

WHEN 4 THEN ''Every ''

+ convert(varchar,s.freq\_interval)

+ '' day(s)''

WHEN 8 THEN ''Every ''

+ convert(varchar,s.freq\_recurrence\_factor)

+ '' weeks(s) on ''

+ master.dbo.fn\_freq\_interval\_desc(s.freq\_interval)

WHEN 16 THEN ''Day '' + convert(varchar,s.freq\_interval)

+ '' of every ''

+ convert(varchar,s.freq\_recurrence\_factor)

+ '' month(s)''

WHEN 32 THEN ''The ''

+ CASE s.freq\_relative\_interval

WHEN 1 THEN ''First''

WHEN 2 THEN ''Second''

WHEN 4 THEN ''Third''

WHEN 8 THEN ''Fourth''

WHEN 16 THEN ''Last'' END

+ CASE s.freq\_interval

WHEN 1 THEN '' Sunday''

WHEN 2 THEN '' Monday''

WHEN 3 THEN '' Tuesday''

WHEN 4 THEN '' Wednesday''

WHEN 5 THEN '' Thursday''

WHEN 6 THEN '' Friday''

WHEN 7 THEN '' Saturday''

WHEN 8 THEN '' Day''

WHEN 9 THEN '' Weekday''

WHEN 10 THEN '' Weekend Day'' END

+ '' of every ''

+ convert(varchar,s.freq\_recurrence\_factor)

+ '' month(s)'' END AS Occurs\_detail

, CASE s.freq\_subday\_type

WHEN 1 THEN ''Occurs once at ''

+ master.dbo.fn\_Time2Str(s.active\_start\_time)

WHEN 2 THEN ''Occurs every ''

+ convert(varchar,s.freq\_subday\_interval)

+ '' Seconds(s) Starting at ''

+ master.dbo.fn\_Time2Str(s.active\_start\_time)

+ '' ending at ''

+ master.dbo.fn\_Time2Str(s.active\_end\_time)

WHEN 4 THEN ''Occurs every ''

+ convert(varchar,s.freq\_subday\_interval)

+ '' Minute(s) Starting at ''

+ master.dbo.fn\_Time2Str(s.active\_start\_time)

+ '' ending at ''

+ master.dbo.fn\_Time2Str(s.active\_end\_time)

WHEN 8 THEN ''Occurs every ''

+ convert(varchar,s.freq\_subday\_interval)

+ '' Hour(s) Starting at ''

+ master.dbo.fn\_Time2Str(s.active\_start\_time)

+ '' ending at ''

+ master.dbo.fn\_Time2Str(s.active\_end\_time) END AS Frequency

, CASE WHEN s.freq\_type = 1 THEN ''On date: ''

+ master.dbo.fn\_Date2Str(s.active\_start\_date)

+ '' At time: ''

+ master.dbo.fn\_Time2Str(s.active\_start\_time)

WHEN s.freq\_type < 64 THEN ''Start date: ''

+ master.dbo.fn\_Date2Str(s.active\_start\_date)

+ '' end date: ''

+ master.dbo.fn\_Date2Str(s.active\_end\_date) END as Duration

, master.dbo.fn\_Date2Str(xp.next\_run\_date) + '' ''

+ master.dbo.fn\_Time2Str(xp.next\_run\_time) AS Next\_Run\_Date

FROM msdb.dbo.sysjobs j (NOLOCK)

INNER JOIN msdb.dbo.sysjobschedules js (NOLOCK) ON j.job\_id = js.job\_id

INNER JOIN msdb.dbo.sysschedules s (NOLOCK) ON js.schedule\_id = s.schedule\_id

INNER JOIN msdb.dbo.syscategories c (NOLOCK) ON j.category\_id = c.category\_id

INNER JOIN #xp\_results xp (NOLOCK) ON j.job\_id = xp.job\_id

WHERE 1 = 1

@Filter

ORDER BY j.name'

IF @Filter = 'Y'

SET @sql = REPLACE(@sql,'@Filter',' AND j.enabled = 1 ')

ELSE

IF @Filter = 'N'

SET @sql = REPLACE(@sql,'@Filter',' AND j.enabled = 0 ')

ELSE

IF @Filter = 'X'

SET @sql = REPLACE(@sql,'@Filter',

'AND s.active\_end\_date < convert(varchar(8),GetDate(),112) ')

ELSE

SET @sql = REPLACE(@sql,'@Filter','')

EXEC(@sql)

## Get SQL Server Job owner

SELECT name, SUSER\_SNAME(owner\_sid) OwnerName,

Enabled = CASE WHEN Enabled = 0 THEN 'No'

ELSE 'Yes'

END

FROM msdb.dbo.sysjobs

ORDER BY name

## How to change DTS package owner

EXEC msdb..sp\_reassign\_dtspackageowner

[@name =] 'name',

[@id =] 'id',

[@newloginname =] 'newloginname'

## Script to find job in executing mode

set nocount on

if object\_id('tempdb..#xp\_results') is not null

          drop table #xp\_results

create table  #xp\_results (job\_id                UNIQUEIDENTIFIER NOT NULL,

                            last\_run\_date         INT              NOT NULL,

                            last\_run\_time         INT              NOT NULL,

                            next\_run\_date         INT              NOT NULL,

                            next\_run\_time         INT              NOT NULL,

                            next\_run\_schedule\_id  INT              NOT NULL,

                            requested\_to\_run      INT              NOT NULL, -- BOOL

                            request\_source        INT              NOT NULL,

                            request\_source\_id     sysname          COLLATE database\_default NULL,

                            running               INT              NOT NULL, -- BOOL

                            current\_step          INT              NOT NULL,

                            current\_retry\_attempt INT              NOT NULL,

                            job\_state             INT              NOT NULL)

Insert into #xp\_results

EXECUTE master.dbo.xp\_sqlagent\_enum\_jobs 1, ''

select b.name

from #xp\_results a

join sysjobs b on a.job\_id = b.job\_id

where running = 1

and b.name = 'test1'

## Find Hung Job

--sp\_helptext DV\_FindHungJob 'Peoplesoft To DW conversion'

CREATE PROCEDURE DV\_FindHungJob

(

@Job\_Name VARCHAR(500),

@Job\_owner VARCHAR(100) = NULL,

@is\_sysadmin INT = 1 ,

--@Jobstatus VARCHAR(30) = null output --Code Commented

@Jobstatus INT OUTPUT --New Code added

)

AS

SET NOCOUNT ON

CREATE TABLE #enum\_job

(

Job\_ID UNIQUEIDENTIFIER,

Last\_Run\_Date INT,

Last\_Run\_Time INT,

Next\_Run\_Date INT,

Next\_Run\_Time INT,

Next\_Run\_Schedule\_ID INT,

Requested\_To\_Run INT,

Request\_Source INT,

Request\_Source\_ID VARCHAR(100),

Running INT,

Current\_Step INT,

Current\_Retry\_Attempt INT,

State INT

)

DECLARE @job\_id UNIQUEIDENTIFIER,

@actualdatetime VARCHAR(100),

@run\_date VARCHAR(100),

@run\_time VARCHAR(100)

SELECT @job\_id = job\_id

FROM msdb..sysjobs

WHERE name = LTRIM(RTRIM(@Job\_Name))

INSERT INTO #enum\_job

EXECUTE master.dbo.xp\_sqlagent\_enum\_jobs @is\_sysadmin, @Job\_owner,@job\_id

SELECT @run\_date = CAST(Next\_Run\_Date AS VARCHAR(50)),

@run\_time = CAST(Next\_Run\_Time AS VARCHAR(50))

FROM #enum\_job

WHERE Job\_ID = @job\_id

SELECT

@actualdatetime = CASE LEN(@run\_time)

WHEN 1 THEN SUBSTRING(@run\_date,1,4) + '-'+SUBSTRING(@run\_date,5,2)+ '-'+SUBSTRING(@run\_date,7,2) + ' '+ '00:00:00'

WHEN 3 THEN SUBSTRING(@run\_date,1,4) + '-'+SUBSTRING(@run\_date,5,2)+ '-'+SUBSTRING(@run\_date,7,2) + ' '+ '00:0'+SUBSTRING(@run\_time,1,1)+':'+SUBSTRING(@run\_time,2,2)

WHEN 4 THEN SUBSTRING(@run\_date,1,4) + '-'+SUBSTRING(@run\_date,5,2)+ '-'+SUBSTRING(@run\_date,7,2) + ' '+ '00:'+SUBSTRING(@run\_time,1,2)+':'+SUBSTRING(@run\_time,3,2)

WHEN 5 THEN SUBSTRING(@run\_date,1,4) + '-'+SUBSTRING(@run\_date,5,2)+ '-'+SUBSTRING(@run\_date,7,2) + ' '+ '0'+SUBSTRING(@run\_time,1,1) + ':'+SUBSTRING(@run\_time,2,2)+ ':'+SUBSTRING(@run\_time,4,2)

ELSE SUBSTRING(@run\_date,1,4) + '-'+SUBSTRING(@run\_date,5,2)+ '-'+SUBSTRING(@run\_date,7,2) + ' '+SUBSTRING(@run\_time,1,2) + ':'+SUBSTRING(@run\_time,3,2)+ ':'+SUBSTRING(@run\_time,5,2)

END

--Code commented

/\*

IF ( SELECT Running FROM #enum\_job ) > 0

BEGIN

set @Jobstatus='Running'

END

\*/

--New Logic Added

IF (SELECT Running FROM #enum\_job) > 0

BEGIN

SET @Jobstatus=1

END

ELSE

BEGIN

SET @Jobstatus=0

END

DROP TABLE #enum\_job

--END OF JOB

## Job Status

set nocount on

declare @job\_name sysname

, @Last\_run\_status varchar(50)

, @Cur\_run\_status varchar(50)

set @job\_name = 'KPIGetData' -- Pass job name

declare @job\_id UNIQUEIDENTIFIER

, @is\_sysadmin INT

, @job\_owner sysname

select @job\_id = job\_id

from msdb..sysjobs\_view

where name = @job\_name

select @is\_sysadmin = ISNULL(IS\_SRVROLEMEMBER(N'sysadmin'), 0)

select @job\_owner = SUSER\_SNAME()

select top 1

@Last\_run\_status = case run\_status

when 0 then 'Failed'

when 1 then 'Succeeded'

when 2 then 'Retry'

when 3 then 'Canceled'

when 4 then 'In progress'

else 'Job History not available'

end

from msdb..sysjobhistory

where job\_id = @job\_id and step\_id = 0

order by instance\_id desc

if @@rowcount = 0

set @Last\_run\_status = 'Job History not available'

CREATE TABLE #xp\_results (job\_id UNIQUEIDENTIFIER NOT NULL,

last\_run\_date INT NOT NULL,

last\_run\_time INT NOT NULL,

next\_run\_date INT NOT NULL,

next\_run\_time INT NOT NULL,

next\_run\_schedule\_id INT NOT NULL,

requested\_to\_run INT NOT NULL, -- BOOL

request\_source INT NOT NULL,

request\_source\_id sysname COLLATE database\_default NULL,

running INT NOT NULL, -- BOOL

current\_step INT NOT NULL,

current\_retry\_attempt INT NOT NULL,

job\_state INT NOT NULL)

IF ((@@microsoftversion / 0x01000000) >= 8) -- SQL Server 8.0 or greater

INSERT INTO #xp\_results

EXECUTE master.dbo.xp\_sqlagent\_enum\_jobs @is\_sysadmin, @job\_owner, @job\_id

ELSE

INSERT INTO #xp\_results

EXECUTE master.dbo.xp\_sqlagent\_enum\_jobs @is\_sysadmin, @job\_owner

--declare @execution\_status int

select @Cur\_run\_status = case job\_state

when 0 then 'Job is not idle or suspended'

when 1 then 'Executing'

when 2 then 'Waiting for thread'

when 3 then 'Between retries'

when 4 then 'Idle'

when 5 then 'Suspended'

when 7 then 'Performing completion actions'

else 'Current Job status not available'

end

from #xp\_results

if @@rowcount = 0

set @Cur\_run\_status = 'Current Job status not available'

/\*

Is the execution status for the jobs.

Value Description

0 Returns only those jobs that are not idle or suspended.

1 Executing.

2 Waiting for thread.

3 Between retries.

4 Idle.

5 Suspended.

7 Performing completion actions

\*/

drop table #xp\_results

select @job\_name as 'job\_name',

@Last\_run\_status as 'Last\_run\_status',

@Cur\_run\_status as 'Current\_status'

set nocount off

**-------------------------------------------------------------------------------------------------------------------------------------------------------**

**Save DTS to file**

**-------------------------------------------------------------------------------------------------------------------------------------------------------**

DECLARE @TARGETDIR varchar(1000)

SET @TARGETDIR = 'C:\Datavail\Scripts\DtsPackages\'

SELECT distinct 'DTSRUN.EXE /S '

+ CONVERT(varchar(200), SERVERPROPERTY('servername'))

+ ' /E '

+ ' /N '

+ '"' + name + '"'

+ ' /F '

+ '"' + @TARGETDIR + replace(replace(name,'\','-'),'/','-') + '.dts"'

+ ' /!X'

FROM msdb.dbo.sysdtspackages P

# User and Login Scripts

## Get all login details

Exec sp\_helplogins

## List of Users per Role

Exec sp\_helpsrvrolemember

## List of special users per database

declare @name sysname,

@SQL nvarchar(600)

if exists (select [id] from tempdb..sysobjects where [id] = OBJECT\_ID ('tempdb..#tmpTable'))

drop table #tmpTable

CREATE TABLE #tmpTable (

[DATABASE\_NAME] sysname NOT NULL ,

[USER\_NAME] sysname NOT NULL,

[ROLE\_NAME] sysname NOT NULL)

declare c1 cursor for

select name from master.dbo.sysdatabases

open c1

fetch c1 into @name

while @@fetch\_status >= 0

begin

select @SQL =

'insert into #tmpTable

select N'''+ @name + ''', a.name, c.name

from ' + QuoteName(@name) + '.dbo.sysusers a

join ' + QuoteName(@name) + '.dbo.sysmembers b on b.memberuid = a.uid

join ' + QuoteName(@name) + '.dbo.sysusers c on c.uid = b.groupuid

where a.name != ''dbo'''

/\* Insert row for each database \*/

execute (@SQL)

fetch c1 into @name

end

close c1

deallocate c1

select \* from #tmpTable

drop table #tmpTable

go

## Cursor to fix orphan users

USE DATABASENAME

DECLARE @username varchar(25)

DECLARE fixusers

CURSOR FOR SELECT UserName = name FROM sysusers

WHERE issqluser = 1 and (sid is not null and sid <> 0x0)

and suser\_sname(sid) is null

ORDER BY name

OPEN fixusers

FETCH NEXT FROM fixusers INTO @username

WHILE @@FETCH\_STATUS = 0

BEGIN

IF @username='dbo'

BEGIN EXEC sp\_changedbowner 'sa'

END

ELSE

BEGIN

EXEC sp\_change\_users\_login 'update\_one', @username, @username

END

FETCH NEXT FROM fixusersINTO @username

END

CLOSE fixusers

DEALLOCATE

fixusers

END

Go

## Script to Fix orphan User for all existing database

DECLARE @DBName nvarchar(255)

DECLARE orphanuser\_curDB cursor for

SELECT name from sysdatabases where name not in ('master','msdb','tempdb','model')

ORDER BY name

OPEN orphanuser\_curDB

FETCH NEXT FROM orphanuser\_curDB INTO @DBName

WHILE (@@fetch\_status = 0)

BEGIN

Declare @sql varchar(5000)

set @sql='Use ['+@DBName+']

DECLARE @UserName nvarchar(255)

DECLARE orphanuser\_cur cursor for

SELECT UserName = name

FROM sysusers

WHERE issqluser = 1 and (sid is not null and sid <> 0x0) and suser\_sname(sid) is null

ORDER BY name

OPEN orphanuser\_cur

FETCH NEXT FROM orphanuser\_cur INTO @UserName

WHILE (@@fetch\_status = 0)

BEGIN

PRINT @UserName + '' user name being resynced''

EXEC sp\_change\_users\_login ''Update\_one'', @UserName, @UserName

FETCH NEXT FROM orphanuser\_cur INTO @UserName

END

CLOSE orphanuser\_cur

DEALLOCATE orphanuser\_cur

go '

print @sql

FETCH NEXT FROM orphanuser\_curDB INTO @DBName

End

CLOSE orphanuser\_curDB

DEALLOCATE orphanuser\_curDB

Go

## Audit SQL Server Logins

/\*

Audit SQL Server user ID

This script will generate an audit of SQL Server logins, as wellas a listing of the database user ID's and the SQL Server login that each DB user ID maps to.

In the database user ID results, [Server Login] = '\*\* Orphaned \*\*' indicates that there is no matching Server login. This script was originally designed for SQL 2000, but works just as

well in SQL 2005.

\*/

IF EXISTS (SELECT \* FROM tempdb.dbo.sysobjects WHERE name = '##Users' AND type in (N'U'))

DROP TABLE ##Users

GO

IF EXISTS (SELECT \* FROM tempdb.dbo.sysobjects WHERE name = '##DBUsers' AND type in (N'U'))

DROP TABLE ##DBUsers

GO

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Always run this from master

USE master

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Declare local variables

DECLARE @DBName VARCHAR(32)

DECLARE @SQLCmd VARCHAR(1024)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Get the SQL Server logins

SELECT sid,

loginname AS [Login Name],

dbname AS [Default Database],

CASE isntname

WHEN 1 THEN 'AD Login'

ELSE 'SQL Login'

END AS [Login Type],

CASE

WHEN isntgroup = 1 THEN 'AD Group'

WHEN isntuser = 1 THEN 'AD User'

ELSE ''

END AS [AD Login Type],

CASE sysadmin

WHEN 1 THEN 'Yes'

ELSE 'No'

END AS [sysadmin],

CASE [securityadmin]

WHEN 1 THEN 'Yes'

ELSE 'No'

END AS [securityadmin],

CASE [serveradmin]

WHEN 1 THEN 'Yes'

ELSE 'No'

END AS [serveradmin],

CASE [setupadmin]

WHEN 1 THEN 'Yes'

ELSE 'No'

END AS [setupadmin],

CASE [processadmin]

WHEN 1 THEN 'Yes'

ELSE 'No'

END AS [processadmin],

CASE [diskadmin]

WHEN 1 THEN 'Yes'

ELSE 'No'

END AS [diskadmin],

CASE [dbcreator]

WHEN 1 THEN 'Yes'

ELSE 'No'

END AS [dbcreator],

CASE [bulkadmin]

WHEN 1 THEN 'Yes'

ELSE 'No'

END AS [bulkadmin]

INTO ##Users

FROM dbo.syslogins

SELECT [Login Name],

[Default Database],

[Login Type],

[AD Login Type],

[sysadmin],

[securityadmin],

[serveradmin],

[setupadmin],

[processadmin],

[diskadmin],

[dbcreator],

[bulkadmin]

FROM ##Users

ORDER BY [Login Type], [AD Login Type], [Login Name]

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Create the output table for the Database User ID's

CREATE TABLE ##DBUsers (

[Database] VARCHAR(64),

[Database User ID] VARCHAR(64),

[Server Login] VARCHAR(64),

[Database Role] VARCHAR(64))

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Declare a cursor to loop through all the databases on the server

DECLARE csrDB CURSOR FOR

SELECT name

FROM sysdatabases

WHERE name NOT IN ('master', 'model', 'msdb', 'tempdb')

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Open the cursor and get the first database name

OPEN csrDB

FETCH NEXT

FROM csrDB

INTO @DBName

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Loop through the cursor

WHILE @@FETCH\_STATUS = 0

BEGIN

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

SELECT @SQLCmd = 'INSERT ##DBUsers ' +

' SELECT ''' + @DBName + ''' AS [Database],' +

' su.[name] AS [Database User ID], ' +

' COALESCE (u.[Login Name], ''\*\* Orphaned \*\*'') AS [Server Login], ' +

' COALESCE (sug.name, ''Public'') AS [Database Role] ' +

' FROM [' + @DBName + '].[dbo].[sysusers] su' +

' LEFT OUTER JOIN ##Users u' +

' ON su.sid = u.sid' +

' LEFT OUTER JOIN ([' + @DBName + '].[dbo].[sysmembers] sm ' +

' INNER JOIN [' + @DBName + '].[dbo].[sysusers] sug ' +

' ON sm.groupuid = sug.uid)' +

' ON su.uid = sm.memberuid ' +

' WHERE su.hasdbaccess = 1' +

' AND su.[name] != ''dbo'' '

EXEC (@SQLCmd)

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Get the next database name

FETCH NEXT

FROM csrDB

INTO @DBName

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- End of the cursor loop

END

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Close and deallocate the CURSOR

CLOSE csrDB

DEALLOCATE csrDB

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Return the Database User data

SELECT \*

FROM ##DBUsers

ORDER BY [Database], [Database User ID]

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GO

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Clean up - delete the Global temp tables

IF EXISTS (SELECT \* FROM tempdb.dbo.sysobjects WHERE name = '##Users' AND type in (N'U'))

DROP TABLE ##Users

GO

IF EXISTS (SELECT \* FROM tempdb.dbo.sysobjects WHERE name = '##DBUsers' AND type in (N'U'))

DROP TABLE ##DBUsers

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GO

## Audit of SQL Server logins, as wellas a listing of the database user ID's

SET NOCOUNT ON

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

/\*

This script will generate an audit of SQL Server logins,

as wellas a listing of the database user ID's and the SQL Server login

that each DB user ID maps to.

In the database user ID results, [Server Login] = '\*\* Orphaned \*\*'

indicates that there is no matching Server login.

This script works both in SQL 2000 SQL 2005.

\*/

IF EXISTS (SELECT \* FROM tempdb.dbo.sysobjects WHERE name = '##Users' AND type in (N'U'))

DROP TABLE ##User

GO

IF EXISTS (SELECT \* FROM tempdb.dbo.sysobjects WHERE name = '##DBUsers' AND type in (N'U'))

DROP TABLE ##DBUsers

GO

-- Always run this from master

USE master

-- Declare local variables

DECLARE @DBName VARCHAR(32)

DECLARE @SQLCmd VARCHAR(1024)

-- Get the SQL Server logins

SELECT sid,

loginname AS [Login Name],

dbname AS [Default Database],

CASE isntname

WHEN 1 THEN 'AD Login'

ELSE 'SQL Login'

END AS [Login Type],

CASE

WHEN isntgroup = 1 THEN 'AD Group'

WHEN isntuser = 1 THEN 'AD User'

ELSE ''

END AS [AD Login Type],

CASE sysadmin

WHEN 1 THEN 'Yes'

ELSE 'No'

END AS [sysadmin],

CASE [securityadmin]

WHEN 1 THEN 'Yes'

ELSE 'No'

END AS [securityadmin],

CASE [serveradmin]

WHEN 1 THEN 'Yes'

ELSE 'No'

END AS [serveradmin],

CASE [setupadmin]

WHEN 1 THEN 'Yes'

ELSE 'No'

END AS [setupadmin],

CASE [processadmin]

WHEN 1 THEN 'Yes'

ELSE 'No'

END AS [processadmin],

CASE [diskadmin]

WHEN 1 THEN 'Yes'

ELSE 'No'

END AS [diskadmin],

CASE [dbcreator]

WHEN 1 THEN 'Yes'

ELSE 'No'

END AS [dbcreator],

CASE [bulkadmin]

WHEN 1 THEN 'Yes'

ELSE 'No'

END AS [bulkadmin]

INTO ##Users

FROM dbo.syslogins

--

--

-- Create the output table for the Database User ID's

CREATE TABLE ##DBUsers (

[Database] VARCHAR(64),

[Database User ID] VARCHAR(64),

[Server Login] VARCHAR(64),

[Database Role] VARCHAR(64))

--

--

-- Declare a cursor to loop through all the databases on the server

DECLARE csrDB CURSOR FOR

SELECT name

FROM sysdatabases

WHERE name NOT IN ('master', 'model', 'msdb', 'tempdb')

AND databasepropertyex(name,'Status')='Online'

-- Open the cursor and get the first database name

OPEN csrDB

FETCH NEXT

FROM csrDB

INTO @DBName

-- Loop through the cursor

WHILE @@FETCH\_STATUS = 0

BEGIN

--

SELECT @SQLCmd = 'INSERT ##DBUsers ' +

' SELECT ''' + @DBName + ''' AS [Database],' +

' su.[name] AS [Database User ID], ' +

' COALESCE (u.[Login Name], ''\*\* Orphaned \*\*'') AS [Server Login], ' +

' COALESCE (sug.name, ''Public'') AS [Database Role] ' +

' FROM [' + @DBName + '].[dbo].[sysusers] su' +

' LEFT OUTER JOIN ##Users u' +

' ON su.sid = u.sid' +

' LEFT OUTER JOIN ([' + @DBName + '].[dbo].[sysmembers] sm ' +

' INNER JOIN [' + @DBName + '].[dbo].[sysusers] sug ' +

' ON sm.groupuid = sug.uid)' +

' ON su.uid = sm.memberuid ' +

' WHERE su.hasdbaccess = 1' +

' AND su.[name] != ''dbo'' '

EXEC (@SQLCmd)

-- Get the next database name

FETCH NEXT

FROM csrDB

INTO @DBName

-- End of the cursor loop

END

-- Close and deallocate the CURSOR

CLOSE csrDB

DEALLOCATE csrDB

-- Return the Database User data

SELECT

CASE WHEN U.[Server Login] is null then L.[Login Name] ELSE U.[Server Login] END AS [Login Name]

,COALESCE (U.[Database User ID], '\*\* NO DB USERS \*\*') AS [Database User ID]

,COALESCE (U.[Database], '\*\*\*\*\*\*') AS [Database]

,COALESCE (U.[Database Role], '\*\*\*\*\*\*') AS [Database Role]

,[Default Database]

, COALESCE ([Login Type],'\*\*\* \*\*\*\*\*\*') AS [Login Type]

, COALESCE ([AD Login Type], '\*\*\* \*\*\*\*\*\*')AS [AD Login Type]

, COALESCE ([sysadmin], '\*\*\*') AS [sysadmin]

, COALESCE ([securityadmin], '\*\*\*')AS [securityadmin]

, COALESCE ([serveradmin], '\*\*\*')AS [serveradmin]

, COALESCE ([setupadmin],'\*\*\*')AS [setupadmin]

, COALESCE ([processadmin], '\*\*\*')AS [processadmin]

, COALESCE ([diskadmin], '\*\*\*')AS [diskadmin]

, COALESCE ([dbcreator], '\*\*\*')AS [dbcreator]

, COALESCE ([bulkadmin],'\*\*\*')AS [bulkadmin]

FROM ##Users L

FuLL JOIN ##DBUsers U

ON L.[Login Name]=U.[Server Login]

ORDER BY U.[Server Login]--, [AD Login Type], [Login Name]

GO

-- Clean up - delete the Global temp tables

IF EXISTS (SELECT \* FROM tempdb.dbo.sysobjects WHERE name = '##Users' AND type in (N'U'))

DROP TABLE ##Users

GO

IF EXISTS (SELECT \* FROM tempdb.dbo.sysobjects WHERE name = '##DBUsers' AND type in (N'U'))

DROP TABLE ##DBUsers

GO

# Database & Object specific Scripts

## Query to show free space in %, TotalSpace occupied in Data files

IF OBJECT\_ID('tempdb.dbo.##filestats') IS NOT NULL DROP TABLE ##filestats

CREATE TABLE ##filestats(Fileid int,FileGroup int,TotalSpace int,UsedSpace int,FreeSpace int,Name varchar(100),FileName varchar(100))

Exec sp\_MSforeachDB

@command1 = 'Use [?];Insert ##filestats (Fileid, FileGroup, TotalSpace,UsedSpace, Name, FileName) exec (''DBCC SHOWFILESTATS WITH TABLERESULTS'')'

UPDATE ##filestats set totalspace = totalspace\*64/1024, usedspace =usedspace\*64/1024

UPDATE ##filestats set freespace = totalspace - usedspace

SELECT

db\_name(dbid) AS DBNAME

,Spaceinperc = (fs.freespace\*100/fs.totalspace)

,fs.TotalSpace AS TotalSpaceinMB

,fs.UsedSpace AS UsedSpaceinMB

,fs.FreeSpace AS FreeSpaceinMB

,fs.Filename

,fs.name AS FileGroup

FROM ##filestats fs

JOIN sysaltfiles sf

ON fs.filename=sf.filename

## Query to get database size, growth report

Select b.name DB\_Name, a.name Logical\_name, a.filename File\_Name,

cast((a.size \* 8.00) / 1024 as numeric(12,2)) as DB\_Size\_in\_MB,

case when a.growth > 100 then 'In MB' else 'In Percentage' end File\_Growth,

cast(case when a.growth > 100 then (a.growth \* 8.00) / 1024

else (((a.size \* a.growth) / 100) \* 8.00) / 1024

end as numeric(12,2)) File\_Growth\_Size\_in\_MB

from sysaltfiles a

join sysdatabases b on a.dbid = b.dbid

where DATABASEPROPERTYEX(b.name, 'status') = 'ONLINE'

order by b.name

## Query to get Log size information

CREATE TABLE #sqlperf

(

DatabaseName VARCHAR(50),

LogSizeMB real,

LogSpaceUsedPercent NUMERIC(5,2),

Status BIT

)

INSERT #sqlperf

EXECUTE ('DBCC sqlperf(logspace)')

SELECT Server\_name= @@servername,

Database\_name= DB\_NAME(dbid),

-- Max\_Size = CAST(@threshold AS VARCHAR) + ' MB',

CurrentLog\_Size = CAST((sf.[size] \* 8 /1024) AS VARCHAR) + ' MB',

CAST(LogSpaceUsedPercent AS VARCHAR) + ' %' LogSpaceUsed,

Log\_Name = sf.[name],

Log\_File = LEFT([filename], 100),

Max\_Growth = CASE sf.[maxsize]

WHEN 0 THEN 'NO GROWTH SET'

WHEN -1 THEN 'NO MAX LIMIT SET'

ELSE STR(((CONVERT(BIGINT,sf.[maxsize]) \* 8) /1024))+ ' MB' END,

File\_Growth=CASE WHEN (sf.status&0x100000) > 0 THEN STR(sf.growth)+' %'

ELSE STR((sf.growth \* 8) /1024)+' MB' END --INTO #tmp\_logspace

FROM master.dbo.sysaltfiles sf (NOLOCK) JOIN #sqlperf perflog

ON sf.dbid = DB\_ID(perflog.DatabaseName)

WHERE lower(FileName) like '%.ldf%'

-- AND (sf.[size] \* 8 /1024) > @threshold

ORDER BY 1

DROP TABLE #sqlperf

## Script to get table size information (row count, reserved space & used space)

select OBJECT\_NAME(object\_id) as objname, Row\_Count

, SUM (reserved\_page\_count) \* 8192/ 1024 as reserved\_kb

, SUM(used\_page\_count) \* 8192 / 1024 as used\_kb

from sys.dm\_db\_partition\_stats

group by OBJECT\_NAME(object\_id),Row\_Count

order by reserved\_kb desc

## Script to get object property of table

Select i.TABLE\_NAME,

Case objectProperty(object\_id(i.TABLE\_NAME), 'TableHasClustIndex')

When 0 then 'No'

When 1 then 'Yes'

End as [Has Clustered Index],

Case objectProperty(object\_id(i.TABLE\_NAME), 'TableHasPrimaryKey')

When 0 then 'No'

When 1 then 'Yes'

End as [Has Primary Key],

Case objectProperty(object\_id(i.TABLE\_NAME), 'TableHasUniqueCnst')

When 0 then 'No'

When 1 then 'Yes'

End as [Has Unique Constraint]

--into #t1

From INFORMATION\_SCHEMA.TABLES i

Where objectProperty(object\_id(i.TABLE\_NAME), 'IsUserTable') = 1 --and

--objectProperty(object\_id(i.TABLE\_NAME), 'TableHasClustIndex') = 0

Order by i.TABLE\_NAME

SELECT \* from #t1 a --where a.Table\_name like '%tmp%'

left join distribution..MSarticles b ON b.Publisher\_db ='LTCabb' AND a.Table\_name =b.article

where b.article is null

order by [HAs primary key],[Has unique constraint],[Has clustered Index]

## Script to Find Trigger status in SQL server 2000

SELECT name, Status= OBJECTPROPERTY (id, 'ExecIsTriggerDisabled')

FROM sysobjects

Where xtype='Tr' and Name like 'TriggerName'

If Status = 0 ---- Trigger is Enable

1 ---- Trigger is Disable

## Script to get user permission on table

DECLARE @TableName Varchar(50)

Set @TableName='PS\_HLD\_FA\_VER\_R2T4'

Select object\_name(id) as Table\_Name, name, status from syspermissions a

join sysusers b on a.grantee = b.uid

where (name like @TableName) and(object\_name (id) = @TableName OR object\_name (id) = @TableName)

## Script to identify Hostname on SQL server is running on

exec xp\_regread 'HKEY\_LOCAL\_MACHINE', 'SYSTEM\CurrentControlSet\Services\Tcpip\Parameters', 'Hostname'

SELECT SERVERPROPERTY('ProductVersion'), SERVERPROPERTY('ProductLevel'), SERVERPROPERTY('Edition'),SERVERPROPERTY('IsClustered') –- IsClustered will return 1 if Server instance is configured for failover cluster.

## Permissions at Object Level

select 'GRANT select ON ' + name + ' to SelectInsertUpdateDeleteExecSP' + 'go' from sysobjects where type = 'u' order by name

go

select 'GRANT insert ON ' + name + ' to SelectInsertUpdateDeleteExecSP' + 'go' from sysobjects where type = 'u' order by name

go

select 'GRANT update ON ' + name + ' to SelectInsertUpdateDeleteExecSP' + 'go' from sysobjects where type = 'u' order by name

go

select 'GRANT delete ON ' + name + ' to SelectInsertUpdateDeleteExecSP' + 'go' from sysobjects where type = 'u' order by name

go

select 'GRANT exec ON ' + name + ' to SelectInsertUpdateDeleteExecSP' + 'go' from sysobjects where type = 'p' order by name

## Store Table Size periodically for Analysis

Create proc usp\_tableSizeAnalysis @dbname varchar (100) as

begin

-- truncate table master..tablesizehistory

set nocount on

declare @sql varchar(3000)

set @sql='use '+ @dbname+'

set nocount on

declare @tblName varchar (100),

@schmName varchar(100),

@crdate datetime,

@sqlstr varchar(255),

@TodayDate datetime

if not exists (select \* from master..sysobjects where id=object\_id(''master..TableSizeHistory''))

begin

create table master..TableSizeHistory

(

dbname varchar(100),

Name varchar (100),

SizeOn datetime,

TableCrDate datetime,

Rows int null,

Reserved int null,

Data int null,

Indexp\_size int null,

Unused int null

)

end

if object\_id(''tempdb..#tbldtls'') is not null

drop table #tbldtls

create table #tbldtls(name varchar (100), rows varchar(15), reserved varchar(15) null, data varchar(15) null, indexp\_size varchar(15) null, unused varchar(15) null)

if object\_id(''tempdb..#tbldtls1'') is not null

drop table #tbldtls1

create table #tbldtls1(name varchar (100), rows varchar(15), reserved varchar(15) null, data varchar(15) null, indexp\_size varchar(15) null, unused varchar(15) null)

declare curtbl cursor for

select u.name,o.name, crdate from '+ @dbname +'..sysobjects o JOIN sys.schemas u

on o.uid=u.schema\_id

where type=''U''

open curtbl

FETCH NEXT FROM curtbl

INTO @schmName, @tblName, @crdate

set @TodayDate = getdate()

WHILE @@FETCH\_STATUS = 0

begin

insert into master..TableSizeHistory values (db\_name(),@schmName+''.''+@tblName ,@TodayDate,@crdate,0,0,0,0,0)

set @sqlstr =''sp\_spaceused ['' + @schmName+''.''+@tblName +'']''

-- print @sqlstr

insert #tbldtls1 (Name,rows,reserved,data,indexp\_size,unused)

exec (@sqlstr)

insert #tbldtls (Name,rows,reserved,data,indexp\_size,unused)

select @schmName+''.''+Name,rows,reserved,data,indexp\_size,unused from #tbldtls1

truncate table #tbldtls1

FETCH NEXT FROM curtbl

INTO @schmName,@tblName, @crdate

end

update master..TableSizeHistory set

master..TableSizeHistory.Rows = cast (replace(#tbldtls.Rows, ''KB'','''') as int),

master..TableSizeHistory.reserved = cast (replace(#tbldtls.reserved, ''KB'','''') as int),

master..TableSizeHistory.data = cast (replace(#tbldtls.data, ''KB'','''') as int),

master..TableSizeHistory.Indexp\_size = cast (replace(#tbldtls.Indexp\_size, ''KB'','''') as int),

master..TableSizeHistory.Unused = cast (replace(#tbldtls.Unused, ''KB'','''') as int)

from #tbldtls inner join master..TableSizeHistory

on master..TableSizeHistory.name =#tbldtls.name

where master..TableSizeHistory.Sizeon=@TodayDate

close curtbl

deallocate curtbl'

--select @sql

exec(@sql)

end

## Database Status

select name as DBNAME,

CASE WHEN (STATUS & 1) = 0 THEN 'FALSE' ELSE 'TRUE' END AS [AUTOCLOSE\_(ALTER\_DATABASE)],

CASE WHEN (STATUS & 4) = 0 THEN 'FALSE' ELSE 'TRUE' END AS [SELECT\_INTO/BULKCOPY\_(ALTER\_DATABASE\_USING\_SET\_RECOVERY)],

CASE WHEN (STATUS & 8) = 0 THEN 'FALSE' ELSE 'TRUE' END AS [TRUNC.\_LOG\_ON\_CHKPT\_(ALTER\_DATABASE\_USING\_SET\_RECOVERY)],

CASE WHEN (STATUS & 16) = 0 THEN 'FALSE' ELSE 'TRUE' END AS [TORN\_PAGE\_DETECTION\_(ALTER\_DATABASE)],

CASE WHEN (STATUS & 32) = 0 THEN 'FALSE' ELSE 'TRUE' END AS [LOADING],

CASE WHEN (STATUS & 64) = 0 THEN 'FALSE' ELSE 'TRUE' END AS [PRE\_RECOVERY],

CASE WHEN (STATUS & 128) = 0 THEN 'FALSE' ELSE 'TRUE' END AS [RECOVERING],

CASE WHEN (STATUS & 256) = 0 THEN 'FALSE' ELSE 'TRUE' END AS [NOT\_RECOVERED],

CASE WHEN (STATUS & 512) = 0 THEN 'FALSE' ELSE 'TRUE' END AS [OFFLINE\_(ALTER\_DATABASE)],

CASE WHEN (STATUS & 1024) = 0 THEN 'FALSE' ELSE 'TRUE' END AS [READ\_ONLY\_(ALTER\_DATABASE)],

CASE WHEN (STATUS & 2048) = 0 THEN 'FALSE' ELSE 'TRUE' END AS [DBO\_USE\_ONLY\_(ALTER\_DATABASE\_USING\_SET\_RESTRICTED\_USER)],

CASE WHEN (STATUS & 4096) = 0 THEN 'FALSE' ELSE 'TRUE' END AS [SINGLE\_USER\_(ALTER\_DATABASE)],

CASE WHEN (STATUS & 32768) = 0 THEN 'FALSE' ELSE 'TRUE' END AS [EMERGENCY\_MODE],

CASE WHEN (STATUS & 4194304) = 0 THEN 'FALSE' ELSE 'TRUE' END AS [AUTOSHRINK\_(ALTER\_DATABASE)],

CASE WHEN (STATUS &1073741824) = 0 THEN 'FALSE' ELSE 'TRUE' END AS [CLEANLY\_SHUTDOWN]

from master.dbo.sysdatabases

## Delete duplicate records from table

The below sp will delete the duplicate records in a table. This sp will take table name and duplicate column name as parameter.

-- exec usp\_Del\_Duplicate\_Rec 'tableName','dupColName'

Create proc usp\_Del\_Duplicate\_Rec @table varchar(50), @pkcol varchar(50) as

begin

declare @SQL varchar(4000)

set @SQL='

-- Declare the variables to store the values returned by FETCH.

DECLARE @accountId varchar(40), @appearances int

-- Get the recordset indicating the AccountId with duplicate entries

DECLARE duplicate\_cursor CURSOR FOR

SELECT ' + @pkcol +', COUNT('+@pkcol+') AS pkcol

FROM ' +@table +'

GROUP BY '+ @pkcol + '

HAVING COUNT('+@pkcol+') > 1

-- Open the recordset

OPEN duplicate\_cursor

-- Perform the first fetch and store the values in variables.

FETCH NEXT FROM duplicate\_cursor

INTO @accountId, @appearances

-- Check @@FETCH\_STATUS to see if there are any more rows to fetch.

WHILE @@FETCH\_STATUS = 0

BEGIN

-- delete all records for this accountId minus 1

-- Determine how many records must be deleted

SET @appearances = @appearances - 1

-- Limit the result of this delete to the above calculated maximum

SET ROWCOUNT @appearances

-- Execute the delete

DELETE '+ @table +'

WHERE ' +@pkcol +'= @accountId

FETCH NEXT FROM duplicate\_cursor

INTO @accountId, @appearances

END

CLOSE duplicate\_cursor

DEALLOCATE duplicate\_cursor

-- Reset the rowcount limits

SET ROWCOUNT 0

'

EXEC (@SQL)

END

## Generate INSERT statements

SET NOCOUNT ON

GO

PRINT 'Checking for the existence of this procedure'

IF (SELECT OBJECT\_ID('sp\_generate\_inserts','P')) IS NOT NULL --means, the procedure already exists

BEGIN

PRINT 'Procedure already exists. So, dropping it'

DROP PROC sp\_generate\_inserts

END

GO

CREATE PROC sp\_generate\_inserts

(

@table\_name varchar(776), -- The table/view for which the INSERT statements will be generated using the existing data

@target\_table varchar(776) = NULL, -- Use this parameter to specify a different table name into which the data will be inserted

@include\_column\_list bit = 1, -- Use this parameter to include/ommit column list in the generated INSERT statement

@from varchar(800) = NULL, -- Use this parameter to filter the rows based on a filter condition (using WHERE)

@include\_timestamp bit = 0, -- Specify 1 for this parameter, if you want to include the TIMESTAMP/ROWVERSION column's data in the INSERT statement

@debug\_mode bit = 0, -- If @debug\_mode is set to 1, the SQL statements constructed by this procedure will be printed for later examination

@owner varchar(64) = NULL, -- Use this parameter if you are not the owner of the table

@ommit\_images bit = 0, -- Use this parameter to generate INSERT statements by omitting the 'image' columns

@ommit\_identity bit = 0, -- Use this parameter to ommit the identity columns

@top int = NULL, -- Use this parameter to generate INSERT statements only for the TOP n rows

@cols\_to\_include varchar(8000) = NULL, -- List of columns to be included in the INSERT statement

@cols\_to\_exclude varchar(8000) = NULL, -- List of columns to be excluded from the INSERT statement

@disable\_constraints bit = 0, -- When 1, disables foreign key constraints and enables them after the INSERT statements

@ommit\_computed\_cols bit = 0 -- When 1, computed columns will not be included in the INSERT statement

)

AS

BEGIN

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Procedure: sp\_generate\_inserts (Build 22)

(Copyright © 2002 Narayana Vyas Kondreddi. All rights reserved.)

Purpose: To generate INSERT statements from existing data.

These INSERTS can be executed to regenerate the data at some other location.

This procedure is also useful to create a database setup, where in you can

script your data along with your table definitions.

Written by: Narayana Vyas Kondreddi

http://vyaskn.tripod.com

Acknowledgements:

Divya Kalra -- For beta testing

Mark Charsley -- For reporting a problem with scripting uniqueidentifier columns with NULL values

Artur Zeygman -- For helping me simplify a bit of code for handling non-dbo owned tables

Joris Laperre -- For reporting a regression bug in handling text/ntext columns

Tested on: SQL Server 7.0 and SQL Server 2000 and SQL Server 2005

Date created: January 17th 2001 21:52 GMT

Date modified: May 1st 2002 19:50 GMT

Email: vyaskn@hotmail.com

NOTE: This procedure may not work with tables with too many columns.

Results can be unpredictable with huge text columns or SQL Server 2000's sql\_variant data types

Whenever possible, Use @include\_column\_list parameter to ommit column list in the INSERT statement, for better results

IMPORTANT: This procedure is not tested with internation data (Extended characters or Unicode). If needed

you might want to convert the datatypes of character variables in this procedure to their respective unicode counterparts

like nchar and nvarchar

ALSO NOTE THAT THIS PROCEDURE IS NOT UPDATED TO WORK WITH NEW DATA TYPES INTRODUCED IN SQL SERVER 2005 / YUKON

Example 1: To generate INSERT statements for table 'titles':

EXEC sp\_generate\_inserts 'titles'

Example 2: To ommit the column list in the INSERT statement: (Column list is included by default)

IMPORTANT: If you have too many columns, you are advised to ommit column list, as shown below,

to avoid erroneous results

EXEC sp\_generate\_inserts 'titles', @include\_column\_list = 0

Example 3: To generate INSERT statements for 'titlesCopy' table from 'titles' table:

EXEC sp\_generate\_inserts 'titles', 'titlesCopy'

Example 4: To generate INSERT statements for 'titles' table for only those titles

which contain the word 'Computer' in them:

NOTE: Do not complicate the FROM or WHERE clause here. It's assumed that you are good with T-SQL if you are using this parameter

EXEC sp\_generate\_inserts 'titles', @from = "from titles where title like '%Computer%'"

Example 5: To specify that you want to include TIMESTAMP column's data as well in the INSERT statement:

(By default TIMESTAMP column's data is not scripted)

EXEC sp\_generate\_inserts 'titles', @include\_timestamp = 1

Example 6: To print the debug information:

EXEC sp\_generate\_inserts 'titles', @debug\_mode = 1

Example 7: If you are not the owner of the table, use @owner parameter to specify the owner name

To use this option, you must have SELECT permissions on that table

EXEC sp\_generate\_inserts Nickstable, @owner = 'Nick'

Example 8: To generate INSERT statements for the rest of the columns excluding images

When using this otion, DO NOT set @include\_column\_list parameter to 0.

EXEC sp\_generate\_inserts imgtable, @ommit\_images = 1

Example 9: To generate INSERT statements excluding (ommiting) IDENTITY columns:

(By default IDENTITY columns are included in the INSERT statement)

EXEC sp\_generate\_inserts mytable, @ommit\_identity = 1

Example 10: To generate INSERT statements for the TOP 10 rows in the table:

EXEC sp\_generate\_inserts mytable, @top = 10

Example 11: To generate INSERT statements with only those columns you want:

EXEC sp\_generate\_inserts titles, @cols\_to\_include = "'title','title\_id','au\_id'"

Example 12: To generate INSERT statements by omitting certain columns:

EXEC sp\_generate\_inserts titles, @cols\_to\_exclude = "'title','title\_id','au\_id'"

Example 13: To avoid checking the foreign key constraints while loading data with INSERT statements:

EXEC sp\_generate\_inserts titles, @disable\_constraints = 1

Example 14: To exclude computed columns from the INSERT statement:

EXEC sp\_generate\_inserts MyTable, @ommit\_computed\_cols = 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SET NOCOUNT ON

--Making sure user only uses either @cols\_to\_include or @cols\_to\_exclude

IF ((@cols\_to\_include IS NOT NULL) AND (@cols\_to\_exclude IS NOT NULL))

BEGIN

RAISERROR('Use either @cols\_to\_include or @cols\_to\_exclude. Do not use both the parameters at once',16,1)

RETURN -1 --Failure. Reason: Both @cols\_to\_include and @cols\_to\_exclude parameters are specified

END

--Making sure the @cols\_to\_include and @cols\_to\_exclude parameters are receiving values in proper format

IF ((@cols\_to\_include IS NOT NULL) AND (PATINDEX('''%''',@cols\_to\_include) = 0))

BEGIN

RAISERROR('Invalid use of @cols\_to\_include property',16,1)

PRINT 'Specify column names surrounded by single quotes and separated by commas'

PRINT 'Eg: EXEC sp\_generate\_inserts titles, @cols\_to\_include = "''title\_id'',''title''"'

RETURN -1 --Failure. Reason: Invalid use of @cols\_to\_include property

END

IF ((@cols\_to\_exclude IS NOT NULL) AND (PATINDEX('''%''',@cols\_to\_exclude) = 0))

BEGIN

RAISERROR('Invalid use of @cols\_to\_exclude property',16,1)

PRINT 'Specify column names surrounded by single quotes and separated by commas'

PRINT 'Eg: EXEC sp\_generate\_inserts titles, @cols\_to\_exclude = "''title\_id'',''title''"'

RETURN -1 --Failure. Reason: Invalid use of @cols\_to\_exclude property

END

--Checking to see if the database name is specified along wih the table name

--Your database context should be local to the table for which you want to generate INSERT statements

--specifying the database name is not allowed

IF (PARSENAME(@table\_name,3)) IS NOT NULL

BEGIN

RAISERROR('Do not specify the database name. Be in the required database and just specify the table name.',16,1)

RETURN -1 --Failure. Reason: Database name is specified along with the table name, which is not allowed

END

--Checking for the existence of 'user table' or 'view'

--This procedure is not written to work on system tables

--To script the data in system tables, just create a view on the system tables and script the view instead

IF @owner IS NULL

BEGIN

IF ((OBJECT\_ID(@table\_name,'U') IS NULL) AND (OBJECT\_ID(@table\_name,'V') IS NULL))

BEGIN

RAISERROR('User table or view not found.',16,1)

PRINT 'You may see this error, if you are not the owner of this table or view. In that case use @owner parameter to specify the owner name.'

PRINT 'Make sure you have SELECT permission on that table or view.'

RETURN -1 --Failure. Reason: There is no user table or view with this name

END

END

ELSE

BEGIN

IF NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = @table\_name AND (TABLE\_TYPE = 'BASE TABLE' OR TABLE\_TYPE = 'VIEW') AND TABLE\_SCHEMA = @owner)

BEGIN

RAISERROR('User table or view not found.',16,1)

PRINT 'You may see this error, if you are not the owner of this table. In that case use @owner parameter to specify the owner name.'

PRINT 'Make sure you have SELECT permission on that table or view.'

RETURN -1 --Failure. Reason: There is no user table or view with this name

END

END

--Variable declarations

DECLARE @Column\_ID int,

@Column\_List varchar(8000),

@Column\_Name varchar(128),

@Start\_Insert varchar(786),

@Data\_Type varchar(128),

@Actual\_Values varchar(8000), --This is the string that will be finally executed to generate INSERT statements

@IDN varchar(128) --Will contain the IDENTITY column's name in the table

--Variable Initialization

SET @IDN = ''

SET @Column\_ID = 0

SET @Column\_Name = ''

SET @Column\_List = ''

SET @Actual\_Values = ''

IF @owner IS NULL

BEGIN

SET @Start\_Insert = 'INSERT INTO ' + '[' + RTRIM(COALESCE(@target\_table,@table\_name)) + ']'

END

ELSE

BEGIN

SET @Start\_Insert = 'INSERT ' + '[' + LTRIM(RTRIM(@owner)) + '].' + '[' + RTRIM(COALESCE(@target\_table,@table\_name)) + ']'

END

--To get the first column's ID

SELECT @Column\_ID = MIN(ORDINAL\_POSITION)

FROM INFORMATION\_SCHEMA.COLUMNS (NOLOCK)

WHERE TABLE\_NAME = @table\_name AND

(@owner IS NULL OR TABLE\_SCHEMA = @owner)

--Loop through all the columns of the table, to get the column names and their data types

WHILE @Column\_ID IS NOT NULL

BEGIN

SELECT @Column\_Name = QUOTENAME(COLUMN\_NAME),

@Data\_Type = DATA\_TYPE

FROM INFORMATION\_SCHEMA.COLUMNS (NOLOCK)

WHERE ORDINAL\_POSITION = @Column\_ID AND

TABLE\_NAME = @table\_name AND

(@owner IS NULL OR TABLE\_SCHEMA = @owner)

IF @cols\_to\_include IS NOT NULL --Selecting only user specified columns

BEGIN

IF CHARINDEX( '''' + SUBSTRING(@Column\_Name,2,LEN(@Column\_Name)-2) + '''',@cols\_to\_include) = 0

BEGIN

GOTO SKIP\_LOOP

END

END

IF @cols\_to\_exclude IS NOT NULL --Selecting only user specified columns

BEGIN

IF CHARINDEX( '''' + SUBSTRING(@Column\_Name,2,LEN(@Column\_Name)-2) + '''',@cols\_to\_exclude) <> 0

BEGIN

GOTO SKIP\_LOOP

END

END

--Making sure to output SET IDENTITY\_INSERT ON/OFF in case the table has an IDENTITY column

IF (SELECT COLUMNPROPERTY( OBJECT\_ID(QUOTENAME(COALESCE(@owner,USER\_NAME())) + '.' + @table\_name),SUBSTRING(@Column\_Name,2,LEN(@Column\_Name) - 2),'IsIdentity')) = 1

BEGIN

IF @ommit\_identity = 0 --Determing whether to include or exclude the IDENTITY column

SET @IDN = @Column\_Name

ELSE

GOTO SKIP\_LOOP

END

--Making sure whether to output computed columns or not

IF @ommit\_computed\_cols = 1

BEGIN

IF (SELECT COLUMNPROPERTY( OBJECT\_ID(QUOTENAME(COALESCE(@owner,USER\_NAME())) + '.' + @table\_name),SUBSTRING(@Column\_Name,2,LEN(@Column\_Name) - 2),'IsComputed')) = 1

BEGIN

GOTO SKIP\_LOOP

END

END

--Tables with columns of IMAGE data type are not supported for obvious reasons

IF(@Data\_Type in ('image'))

BEGIN

IF (@ommit\_images = 0)

BEGIN

RAISERROR('Tables with image columns are not supported.',16,1)

PRINT 'Use @ommit\_images = 1 parameter to generate INSERTs for the rest of the columns.'

PRINT 'DO NOT ommit Column List in the INSERT statements. If you ommit column list using @include\_column\_list=0, the generated INSERTs will fail.'

RETURN -1 --Failure. Reason: There is a column with image data type

END

ELSE

BEGIN

GOTO SKIP\_LOOP

END

END

--Determining the data type of the column and depending on the data type, the VALUES part of

--the INSERT statement is generated. Care is taken to handle columns with NULL values. Also

--making sure, not to lose any data from flot, real, money, smallmomey, datetime columns

SET @Actual\_Values = @Actual\_Values +

CASE

WHEN @Data\_Type IN ('char','varchar','nchar','nvarchar')

THEN

'COALESCE('''''''' + REPLACE(RTRIM(' + @Column\_Name + '),'''''''','''''''''''')+'''''''',''NULL'')'

WHEN @Data\_Type IN ('datetime','smalldatetime')

THEN

'COALESCE('''''''' + RTRIM(CONVERT(char,' + @Column\_Name + ',109))+'''''''',''NULL'')'

WHEN @Data\_Type IN ('uniqueidentifier')

THEN

'COALESCE('''''''' + REPLACE(CONVERT(char(255),RTRIM(' + @Column\_Name + ')),'''''''','''''''''''')+'''''''',''NULL'')'

WHEN @Data\_Type IN ('text','ntext')

THEN

'COALESCE('''''''' + REPLACE(CONVERT(char(8000),' + @Column\_Name + '),'''''''','''''''''''')+'''''''',''NULL'')'

WHEN @Data\_Type IN ('binary','varbinary')

THEN

'COALESCE(RTRIM(CONVERT(char,' + 'CONVERT(int,' + @Column\_Name + '))),''NULL'')'

WHEN @Data\_Type IN ('timestamp','rowversion')

THEN

CASE

WHEN @include\_timestamp = 0

THEN

'''DEFAULT'''

ELSE

'COALESCE(RTRIM(CONVERT(char,' + 'CONVERT(int,' + @Column\_Name + '))),''NULL'')'

END

WHEN @Data\_Type IN ('float','real','money','smallmoney')

THEN

'COALESCE(LTRIM(RTRIM(' + 'CONVERT(char, ' + @Column\_Name + ',2)' + ')),''NULL'')'

ELSE

'COALESCE(LTRIM(RTRIM(' + 'CONVERT(char, ' + @Column\_Name + ')' + ')),''NULL'')'

END + '+' + ''',''' + ' + '

--Generating the column list for the INSERT statement

SET @Column\_List = @Column\_List + @Column\_Name + ','

SKIP\_LOOP: --The label used in GOTO

SELECT @Column\_ID = MIN(ORDINAL\_POSITION)

FROM INFORMATION\_SCHEMA.COLUMNS (NOLOCK)

WHERE TABLE\_NAME = @table\_name AND

ORDINAL\_POSITION > @Column\_ID AND

(@owner IS NULL OR TABLE\_SCHEMA = @owner)

--Loop ends here!

END

--To get rid of the extra characters that got concatenated during the last run through the loop

SET @Column\_List = LEFT(@Column\_List,len(@Column\_List) - 1)

SET @Actual\_Values = LEFT(@Actual\_Values,len(@Actual\_Values) - 6)

IF LTRIM(@Column\_List) = ''

BEGIN

RAISERROR('No columns to select. There should at least be one column to generate the output',16,1)

RETURN -1 --Failure. Reason: Looks like all the columns are ommitted using the @cols\_to\_exclude parameter

END

--Forming the final string that will be executed, to output the INSERT statements

IF (@include\_column\_list <> 0)

BEGIN

SET @Actual\_Values =

'SELECT ' +

CASE WHEN @top IS NULL OR @top < 0 THEN '' ELSE ' TOP ' + LTRIM(STR(@top)) + ' ' END +

'''' + RTRIM(@Start\_Insert) +

' ''+' + '''(' + RTRIM(@Column\_List) + '''+' + ''')''' +

' +''VALUES(''+ ' + @Actual\_Values + '+'')''' + ' ' +

COALESCE(@from,' FROM ' + CASE WHEN @owner IS NULL THEN '' ELSE '[' + LTRIM(RTRIM(@owner)) + '].' END + '[' + rtrim(@table\_name) + ']' + '(NOLOCK)')

END

ELSE IF (@include\_column\_list = 0)

BEGIN

SET @Actual\_Values =

'SELECT ' +

CASE WHEN @top IS NULL OR @top < 0 THEN '' ELSE ' TOP ' + LTRIM(STR(@top)) + ' ' END +

'''' + RTRIM(@Start\_Insert) +

' '' +''VALUES(''+ ' + @Actual\_Values + '+'')''' + ' ' +

COALESCE(@from,' FROM ' + CASE WHEN @owner IS NULL THEN '' ELSE '[' + LTRIM(RTRIM(@owner)) + '].' END + '[' + rtrim(@table\_name) + ']' + '(NOLOCK)')

END

--Determining whether to ouput any debug information

IF @debug\_mode =1

BEGIN

PRINT '/\*\*\*\*\*START OF DEBUG INFORMATION\*\*\*\*\*'

PRINT 'Beginning of the INSERT statement:'

PRINT @Start\_Insert

PRINT ''

PRINT 'The column list:'

PRINT @Column\_List

PRINT ''

PRINT 'The SELECT statement executed to generate the INSERTs'

PRINT @Actual\_Values

PRINT ''

PRINT '\*\*\*\*\*END OF DEBUG INFORMATION\*\*\*\*\*/'

PRINT ''

END

PRINT '--INSERTs generated by ''sp\_generate\_inserts'' stored procedure '

PRINT ''

PRINT 'SET NOCOUNT ON'

PRINT ''

--Determining whether to print IDENTITY\_INSERT or not

IF (@IDN <> '')

BEGIN

PRINT 'SET IDENTITY\_INSERT ' + QUOTENAME(COALESCE(@owner,USER\_NAME())) + '.' + QUOTENAME(@table\_name) + ' ON'

PRINT 'GO'

PRINT ''

END

IF @disable\_constraints = 1 AND (OBJECT\_ID(QUOTENAME(COALESCE(@owner,USER\_NAME())) + '.' + @table\_name, 'U') IS NOT NULL)

BEGIN

IF @owner IS NULL

BEGIN

SELECT 'ALTER TABLE ' + QUOTENAME(COALESCE(@target\_table, @table\_name)) + ' NOCHECK CONSTRAINT ALL' AS '--Code to disable constraints temporarily'

END

ELSE

BEGIN

SELECT 'ALTER TABLE ' + QUOTENAME(@owner) + '.' + QUOTENAME(COALESCE(@target\_table, @table\_name)) + ' NOCHECK CONSTRAINT ALL' AS '--Code to disable constraints temporarily'

END

PRINT 'GO'

END

PRINT ''

PRINT 'PRINT ''Inserting values into ' + '[' + RTRIM(COALESCE(@target\_table,@table\_name)) + ']' + ''''

--All the hard work pays off here!!! You'll get your INSERT statements, when the next line executes!

EXEC (@Actual\_Values)

PRINT 'PRINT ''Done'''

PRINT ''

IF @disable\_constraints = 1 AND (OBJECT\_ID(QUOTENAME(COALESCE(@owner,USER\_NAME())) + '.' + @table\_name, 'U') IS NOT NULL)

BEGIN

IF @owner IS NULL

BEGIN

SELECT 'ALTER TABLE ' + QUOTENAME(COALESCE(@target\_table, @table\_name)) + ' CHECK CONSTRAINT ALL' AS '--Code to enable the previously disabled constraints'

END

ELSE

BEGIN

SELECT 'ALTER TABLE ' + QUOTENAME(@owner) + '.' + QUOTENAME(COALESCE(@target\_table, @table\_name)) + ' CHECK CONSTRAINT ALL' AS '--Code to enable the previously disabled constraints'

END

PRINT 'GO'

END

PRINT ''

IF (@IDN <> '')

BEGIN

PRINT 'SET IDENTITY\_INSERT ' + QUOTENAME(COALESCE(@owner,USER\_NAME())) + '.' + QUOTENAME(@table\_name) + ' OFF'

PRINT 'GO'

END

PRINT 'SET NOCOUNT OFF'

SET NOCOUNT OFF

RETURN 0 --Success. We are done!

END

GO

PRINT 'Created the procedure'

GO

--Mark procedure as system object

EXEC sys.sp\_MS\_marksystemobject sp\_generate\_inserts

GO

PRINT 'Granting EXECUTE permission on sp\_generate\_inserts to all users'

GRANT EXEC ON sp\_generate\_inserts TO public

SET NOCOUNT OFF

GO

PRINT 'Done'

## Database Status

SET NOCOUNT ON

declare @DB\_Name varchar(100)

set @DB\_Name = 'tempdb' -- status require for the database

DECLARE @db\_status VARCHAR(200), @db\_Updateability VARCHAR(200), @db\_UserAccess VARCHAR(200)

IF OBJECT\_ID('tempdb..#dbInfo') IS NOT NULL

DROP TABLE #dbInfo

CREATE TABLE #dbInfo (KeyId TINYINT, db\_Descriptiion VARCHAR(100))

IF NOT EXISTS(SELECT name FROM master..sysdatabases WHERE name =@DB\_Name)

SET @db\_status = 'Database does not exist'

ELSE

INSERT #dbInfo EXEC('SELECT 1, CAST(DATABASEPROPERTYEX('+ '''' + @DB\_Name + '''' + ','+''''+'STATUS'+''''+') AS VARCHAR(50))')

SELECT @db\_status = CASE db\_Descriptiion

WHEN 'RESTORING' THEN 'Database is being restored'

WHEN 'RECOVERING' THEN 'Database is recovering and not yet ready for queries'

WHEN 'SUSPECT' THEN 'Database is suspected'

WHEN 'OFFLINE' THEN 'Database is offline'

WHEN 'EMERGENCY' THEN 'Database is in an emergency, read-only state'

WHEN NULL THEN 'Database does not exist'

ELSE db\_Descriptiion END

FROM #dbInfo WHERE KeyId = 1

IF @db\_status = 'ONLINE'

BEGIN

INSERT #dbInfo EXEC('SELECT 2, CAST(DATABASEPROPERTYEX('+ '''' + @DB\_Name + '''' + ','+''''+'Updateability'+''''+') AS VARCHAR(50))')

INSERT #dbInfo EXEC('SELECT 3, CAST(DATABASEPROPERTYEX('+ '''' + @DB\_Name + '''' + ','+''''+'UserAccess'+''''+') AS VARCHAR(50))')

SELECT @db\_Updateability = db\_Descriptiion FROM #dbInfo WHERE KeyId = 2

SELECT @db\_UserAccess = db\_Descriptiion FROM #dbInfo WHERE KeyId = 3

END

SELECT @db\_status DB\_Status, @db\_Updateability Updatibility, @db\_UserAccess UserAccess

DROP TABLE #dbInfo

## Generate PK Script

--this script creates an 'alter' script for a table to migrate pks

-- make sure to change the text width to 900

if exists (select name from tempdb.sys.tables where name like '#PKinfo%')

begin

drop table #pkinfo

end;

set nocount on;

With PKs as (

select kc.object\_id PK\_ID,

kc.name PKName,

'['+c.name+'] '+ case when ic.is\_descending\_key = 0 then 'ASC' else 'DESC' end as ColumnNames,

ic.key\_ordinal ColumnOrder

from sys.key\_constraints kc

join sys.index\_columns ic

on kc.parent\_object\_id = ic.object\_id

and kc.unique\_index\_id = ic.index\_id

join sys.columns c

on kc.parent\_object\_id = c.object\_id

and ic.column\_id = c.column\_id

where type = 'PK'

)

select distinct

schema\_name(o.schema\_id) sname,

object\_name(o.object\_id) tname,

object\_name(kc.object\_id) kname,

(select ColumnNames+',' from PKs x

where kc.object\_id = x.PK\_ID

order by x.ColumnOrder

for xml path ('')) cnames,

ds.name fname

----

into #PKInfo

----

from sys.index\_columns ic

join sys.objects o

on o.object\_id = ic.object\_id

join sys.indexes i

on ic.object\_id = i.object\_id

and ic.index\_id= i.index\_id

join sys.data\_spaces ds

on i.data\_space\_id = ds.data\_space\_id

join sys.key\_constraints kc

on o.object\_id = kc.parent\_object\_id

and ic.index\_id = kc.unique\_index\_id

where schema\_name(o.schema\_id) <> 'sys'

and ds.type = 'FG'

Select '--script ['+kname+'] on ['+sname+'].['+tname+']

ALTER TABLE ['+sname+'].['+tname+']

ADD CONSTRAINT ['+kname+'] PRIMARY KEY CLUSTERED

('+LEFT(cnames,len(cnames)-1)+')

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF,

IGNORE\_DUP\_KEY = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON ['+fname+']

GO

'

from #pkinfo

## Generate FK Script

set nocount on;

With FKs as (

select schema\_name(fk.schema\_id) sname,

fk.name fkname,

object\_name(fk.parent\_object\_id) tname,

c1.name fkcname,

object\_name(fc.referenced\_object\_id) pktname,

c2.name pkcname

from sys.foreign\_keys fk

join sys.foreign\_key\_columns fc

on fk.object\_id = fc.constraint\_object\_id

join sys.index\_columns ic

on fk.parent\_object\_id = ic.object\_id

and fk.key\_index\_id = ic.index\_id

join sys.columns c1

on fc.parent\_column\_id = c1.column\_id

and fc.parent\_object\_id = c1.object\_id

join sys.columns c2

on fc.referenced\_column\_id = c2.column\_id

and fc.referenced\_object\_id = c2.object\_id

)

Select

'--script ['+fkname+'] on ['+sname+'].['+tname+']

ALTER TABLE ['+sname+'].['+tname+'] WITH CHECK ADD

CONSTRAINT ['+fkname+'] FOREIGN KEY(['+fkcname+'])

REFERENCES ['+sname+'].['+pktname+'] (['+pkcname+'])

GO

ALTER TABLE ['+sname+'].['+tname+']

CHECK CONSTRAINT ['+fkname+']

GO

'

from FKs

# Index Scripts

## Script to find Missing Indexes from other database server

**-----------------------------------------------------------------------------**

This script will help to compare two database on different server and will generate a dynamic script for creating mising indexes over destination database.Mostly helpful in scenario of replication running with missing indexes over subscriber.Be careful as creating indexes over large reporting\production database will take long time.

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

set nocount on

if object\_id('tempdb.dbo.#t') is not null

drop table #t

if object\_id('tempdb.dbo.#hlpidxresult') is not null

drop table #hlpidxresult

create table #hlpidxresult (indexname varchar(100), indexdesc varchar(200), indexkeys varchar(200))

create table #t (tblname varchar(100), indexname varchar(100), indexdesc varchar(200), indexkeys varchar(200))

declare @tblname varchar(100)

declare cIdx cursor read\_only forward\_only for

select name from sysobjects where xtype in ('U') order by name

open cIdx

FetchPara:

fetch next from cIdx into @tblname

while @@fetch\_status = 0

begin

truncate table #hlpidxresult

insert #hlpidxresult exec sp\_helpindex @tblname

insert #t

select @tblname, \* from #hlpidxresult

goto FetchPara

end

close cIdx

deallocate cIdx

select \* into sv\_newsvridx from #t

-- run above statements on both server and insert data into physical table say sv\_newsvridx, sv\_oldsvridx

-- list all missing indexes

SELECT a.\*

FROM sv\_newsvridx a

left join OPENDATASOURCE(

'SQLOLEDB',

'Data Source=SERVERNAME1;User ID=userID;Password=\*\*\*\*\*\*\*\*\*'

).DataMirror.dbo.sv\_oldsvridx as b

on a.tblname = b.tblname and a.indexkeys = b.indexkeys

where b.tblname is null

order by a.tblname, a.indexname

-- Generate Script

SELECT 'CREATE ' +

case when a.indexdesc like '%unique%' then 'UNIQUE ' else '' end +

case when a.indexdesc like '%nonclustered%' then 'NONCLUSTERED ' else 'CLUSTERED ' end +

' INDEX ' + a.indexname + ' ON ' + a.tblname + '(' + a.indexkeys + ')'

FROM sv\_newsvridx a

left join OPENDATASOURCE(

'SQLOLEDB',

'Data Source=SERVERNAME1;User ID=userID;Password=\*\*\*\*\*\*\*\*'

).DataMirror.dbo.sv\_oldsvridx as b

on a.tblname = b.tblname and a.indexkeys = b.indexkeys

where b.tblname is null

and a.indexdesc not like '%primary key%'

and a.tblname in (select name from sysobjects where xtype = 'U')

order by a.tblname, a.indexname

## Index Rebuild script for SQL Server 2005

CREATE PROCEDURE [dbo].[INDEXREBUILD]

@DBNAME VARCHAR(100), -- Name of database

@MIN\_FRAG INT, -- Minimum fragmentation value

@MAX\_FRAG INT, -- Maximum fragmentation value

@TABSIZE BIGINT, -- Table size in pages

@INDEXTYPE VARCHAR(10), -- REBUILD or REORGANIZE

@REBUILDOPT VARCHAR(10) = 'ONLINE' -- ONLINE REUILD option (available only for Enterprise editions)

AS

SET NOCOUNT ON

BEGIN

DECLARE @INDEXID INT;

DECLARE @PARTITIONNUM BIGINT;

DECLARE @PARTITIONS BIGINT;

DECLARE @FRAG FLOAT;

DECLARE @COMMAND NVARCHAR(4000);

DECLARE @SQLSTRING NVARCHAR(1000);

DECLARE @PARMDEFINITION NVARCHAR(2000);

DECLARE @DBID INT

DECLARE @PARTITIONCOUNT BIGINT;

DECLARE @SCHEMANAME NVARCHAR(130);

DECLARE @OBJECTNAME NVARCHAR(130);

DECLARE @INDEXNAME NVARCHAR(130);

DECLARE @OBJECTID INT;

DECLARE @IS\_BLOB BIT;

CREATE TABLE #INDEXSTATS

(

OBJNAME VARCHAR(130),

IDXNAME VARCHAR(130),

SCHNAME VARCHAR(130),

AVG\_FRAGMENTATION\_IN\_PERCENT FLOAT,

TBLSIZE BIGINT,

IS\_BLOB BIT

)

SET @COMMAND = '

SELECT main.\*, case when blb.[name] is not null then 1 else 0 end IS\_BLOB

FROM (

SELECT O.NAME OBJNAME,

I.NAME IDXNAME,

S.NAME SCHNAME,

AVG\_FRAGMENTATION\_IN\_PERCENT,

SUM(TOTAL\_PAGES) TOTAL\_PAGES

FROM SYS.DM\_DB\_INDEX\_PHYSICAL\_STATS ('+ CAST(DB\_ID(@DBNAME) AS VARCHAR(3)) +', NULL, NULL, NULL, NULL) V

JOIN ['+ @DBNAME +'].SYS.OBJECTS AS O ON V.OBJECT\_ID = O.OBJECT\_ID

JOIN ['+ @DBNAME +'].SYS.SCHEMAS AS S ON S.SCHEMA\_ID = O.SCHEMA\_ID

JOIN ['+ @DBNAME +'].SYS.INDEXES AS I ON I.OBJECT\_ID = O.OBJECT\_ID

AND V.INDEX\_ID = I.INDEX\_ID

JOIN ['+ @DBNAME +'].SYS.PARTITIONS AS P ON P.OBJECT\_ID = O.OBJECT\_ID

JOIN ['+ @DBNAME +'].SYS.ALLOCATION\_UNITS A ON P.PARTITION\_ID = A.CONTAINER\_ID

WHERE AVG\_FRAGMENTATION\_IN\_PERCENT >= '+ CAST(@MIN\_FRAG AS VARCHAR(8)) + '

AND AVG\_FRAGMENTATION\_IN\_PERCENT <= '+ CAST(@MAX\_FRAG AS VARCHAR(8)) + '

AND I.INDEX\_ID > 0

AND O.[NAME] NOT LIKE ''%tmp%''

AND O.[NAME] NOT LIKE ''temp%''

AND O.[NAME] NOT LIKE ''tblTmp%''

AND O.[NAME] NOT LIKE ''%TRACE%''

GROUP BY O.NAME,

I.NAME,

S.NAME,

AVG\_FRAGMENTATION\_IN\_PERCENT

HAVING SUM(TOTAL\_PAGES) >= ' + CAST(@TABSIZE AS VARCHAR(50))

+ '' + ') main

LEFT JOIN (

select distinct obj.[name]

from ['+ @DBNAME +'].sys.columns col

join ['+ @DBNAME +'].sys.types tp on col.system\_type\_id = tp.system\_type\_id

join ['+ @DBNAME +'].sys.objects obj on col.[object\_id] = obj.[object\_id]

where (col.system\_type\_id IN (35,165, 99, 34, 173) or col.max\_length = -1)

) as blb on main.OBJNAME = blb.[name] '

INSERT INTO #INDEXSTATS

(

OBJNAME, IDXNAME SCHNAME, AVG\_FRAGMENTATION\_IN\_PERCENT, TBLSIZE, IS\_BLOB

)

EXEC(@COMMAND)

SET @COMMAND = ''

DECLARE TAB CURSOR FOR

SELECT OBJNAME, IDXNAME, SCHNAME, IS\_BLOB

FROM #INDEXSTATS;

OPEN TAB;

FETCH NEXT

FROM TAB

INTO @OBJECTNAME, @INDEXNAME, @SCHEMANAME, @IS\_BLOB;

WHILE @@FETCH\_STATUS = 0

BEGIN;

IF @INDEXTYPE = 'REBUILD'

BEGIN

IF @REBUILDOPT = 'ONLINE'

SET @COMMAND = N'ALTER INDEX [' + @INDEXNAME + N'] ON [' + @DBNAME + N'].[' + @SCHEMANAME + N'].[' + @OBJECTNAME + N']

REBUILD WITH (ONLINE = ' + CASE WHEN @IS\_BLOB = 1 THEN ' OFF ' ELSE ' ON ' END + ', SORT\_IN\_TEMPDB = ON)';

-- SET @COMMAND = N'ALTER INDEX [' + @INDEXNAME + N'] ON [' + @DBNAME + N'].[' + @SCHEMANAME + N'].[' + @OBJECTNAME + N'] REBUILD WITH (ONLINE = ON, SORT\_IN\_TEMPDB = ON)';

ELSE

SET @COMMAND = N'ALTER INDEX [' + @INDEXNAME + N'] ON [' + @DBNAME + N'].[' + @SCHEMANAME + N'].[' + @OBJECTNAME + N'] REBUILD WITH (ONLINE = OFF, SORT\_IN\_TEMPDB = ON)';

END

ELSE

SET @COMMAND = N'ALTER INDEX [' + @INDEXNAME + N'] ON [' + @DBNAME + N'].[' + @SCHEMANAME + N'].[' + @OBJECTNAME + N'] REORGANIZE';

IF @OBJECTNAME like '%Archive'

SET @COMMAND = REPLACE (@COMMAND,'REBUILD WITH (','REBUILD WITH (FILLFACTOR = 100,' )

-- SELECT'EXECUTING: ' + @COMMAND;

EXEC (@COMMAND);

print @COMMAND

FETCH NEXT

FROM TAB

INTO @OBJECTNAME, @INDEXNAME, @SCHEMANAME, @IS\_BLOB;

END;

CLOSE TAB;

DEALLOCATE TAB;

DROP TABLE #INDEXSTATS

END

## Index Rebuild script for SQL Server 2000 & 2005

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

--[**Create DV\_FindHungJob (Click here to get the scripts)**](#Find_Hung_Job)

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

use msdb

go

ALTER PROCEDURE [dbo].[DV\_INDEX\_MAINTAINENCE]

@DBNAME VARCHAR(100) -- Name of database

,@MIN\_FRAG INT -- Minimum fragmentation value

,@MAX\_FRAG INT -- Maximum fragmentation value

,@TABSIZE BIGINT -- Table size in pages

,@INDEXTYPE VARCHAR(10) -- REBUILD or REORGANIZE

,@FILLFACT INT = 100 -- Fill Factor

,@REBUILDOPT VARCHAR(10) = 'ONLINE' -- ONLINE REBUILD option (available only for 2005 Enterprise editions)

,@SORTINTEMPDB BIT = 1 -- 1 = SORT\_IN\_TEMPDB option is ON, REBUILD INDEXES ONLY

,@LOB\_COMPACTION VARCHAR(3) = 'OFF' -- LOB Compaction, while REORGANIZE

,@TLOG\_FLAG BIT = 0 -- Flag to start TLogs backup job

,@TLOG\_NAME VARCHAR(200) = NULL -- TLOGs backup job name

,@LOGLIMIT INT = 3072

,@PRINT BIT = 0 -- Flag to print command

,@EXECUTE BIT = 1 -- Flag to execute command

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Version Modifier Date Description

-- 1.1 Anup 07/19/2010 The proc was modified to incorporate OFFLINE indexing of LOB Data (text, varbinary(max), varchar(max),etc...)

-- 1.2 Anup 11/09/2010 Uses flags to starting T Log backups, flags to print/execute/both

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

AS

SET NOCOUNT ON

BEGIN

DECLARE @INDEXID INT;

DECLARE @PARTITIONNUM BIGINT;

DECLARE @PARTITIONS BIGINT;

DECLARE @FRAG FLOAT;

DECLARE @COMMAND NVARCHAR(4000);

DECLARE @SQLSTRING NVARCHAR(1000);

DECLARE @PARMDEFINITION NVARCHAR(2000);

DECLARE @DBID INT;

DECLARE @PARTITIONCOUNT BIGINT;

DECLARE @SCHEMANAME NVARCHAR(130);

DECLARE @OBJECTNAME NVARCHAR(130);

DECLARE @INDEXNAME NVARCHAR(130);

DECLARE @INDEXTYPE\_DMV VARCHAR(50);

DECLARE @ALLOCUNITTYPE VARCHAR(50);

DECLARE @OBJECTID INT;

DECLARE @IS\_BLOB BIT;

DECLARE @Exec\_Stmt VARCHAR(1000)

, @TableName VARCHAR (128)

, @CMPTLEVEL INT ;

DECLARE @query VARCHAR(8000) ;

DECLARE @sql VARCHAR(255)

, @Logsize DECIMAL(15,4)

, @jobst VARCHAR(30);

DECLARE @JOB\_ID UNIQUEIDENTIFIER;

DECLARE @REBUILD\_OPT VARCHAR(3);

DECLARE @SORT\_IN\_TEMPDB VARCHAR(3);

DECLARE @ALLOC\_UNIT VARCHAR(20);

DECLARE @LNGDATATYPE INT;

IF (@TLOG\_FLAG = 1 and (@TLOG\_NAME = NULL OR @TLOG\_NAME=''))

BEGIN

RAISERROR('Transaction Log backup job name is not specified. Transaction Log backup job name is required when @TLOG\_FLAG = 1',1001,1001)

RETURN

END

IF (NOT EXISTS(SELECT job\_id FROM msdb..sysjobs WHERE name = @TLOG\_NAME)) AND @TLOG\_FLAG = 1

BEGIN

RAISERROR('No transaction backup job with the specified name found. Please check the job name.',10,100)

RETURN

END

ELSE

SELECT @JOB\_ID = job\_id FROM msdb..sysjobs WHERE name = @TLOG\_NAME

IF OBJECT\_ID('tempdb.dbo.#Frag\_Details') IS NOT NULL DROP TABLE #Frag\_Details

CREATE TABLE #Frag\_Details

(

DBNAME VARCHAR(130),

OBJNAME VARCHAR(130),

IDXNAME VARCHAR(130),

SCHNAME VARCHAR(130),

AVG\_FRAGMENTATION\_IN\_PERCENT FLOAT,

INDEXTYPE\_DMV VARCHAR(50),

ALLOCUNITTYPE VARCHAR(50),

LNGDATATYPE INT,

TBLSIZE BIGINT,

CMPTLEVEL INT

)

/\* DB List to Check Fragmentation \*/

IF OBJECT\_ID('tempdb.dbo.#DBCheck') IS NOT NULL DROP TABLE #DBCheck

CREATE TABLE #DBCheck

(

DBName varchar(100),

DBID INT,

CMPTLEVEL INT

)

Insert Into #DBCheck

SELECT name, dbid, cmptlevel

FROM master.dbo.sysdatabases

WHERE name= CASE WHEN LEN(ISNULL(@DBNAME, '')) = 0 THEN name ELSE @DBNAME END AND

name NOT IN ('Master', 'distribution', 'msdb', 'model','pubs','northwind', 'dba') AND

name NOT LIKE '%temp%' AND name NOT LIKE '%tmp%'--AND name NOT LIKE '%test%'

AND name NOT LIKE '%train%' AND

DATABASEPROPERTYEX(name, 'Status') = 'ONLINE'

/\* ...............................\*/

/\*\*\* FOR COMPATIBILITY LEVEL = 90 and 100 only \*\*\*/

Declare @dbid1 INT

DECLARE DBCheck CURSOR for

Select DBName,DBID,CMPTLEVEL from #DBCheck where CMPTLEVEL >= 90

OPEN DBCheck

FETCH NEXT FROM DBCheck into @DBNAME,@dbid1,@CMPTLEVEL

WHILE(@@FETCH\_STATUS = 0)

BEGIN

SET @COMMAND = 'SELECT ''['+ @DBNAME + ']'' DBNAME

,O.NAME OBJNAME

,I.NAME IDXNAME

,S.NAME SCHNAME

,AVG\_FRAGMENTATION\_IN\_PERCENT

,V.INDEX\_TYPE\_DESC AS INDEXTYPE\_DMV

,ALLOC\_UNIT\_TYPE\_DESC

,ISNULL(SQ.OBJECT\_ID,1)

,SUM(TOTAL\_PAGES)

,' + cast(@CMPTLEVEL as varchar(3)) +' AS CMPTLEVEL

FROM SYS.DM\_DB\_INDEX\_PHYSICAL\_STATS ('+ CAST(DB\_ID(@DBNAME) AS VARCHAR(3)) +', NULL, NULL, NULL, NULL) V

JOIN ['+ @DBNAME +'].SYS.OBJECTS AS O ON V.OBJECT\_ID = O.OBJECT\_ID

JOIN ['+ @DBNAME +'].SYS.SCHEMAS AS S ON S.SCHEMA\_ID = O.SCHEMA\_ID

JOIN ['+ @DBNAME +'].SYS.INDEXES AS I ON I.OBJECT\_ID = O.OBJECT\_ID

AND V.INDEX\_ID = I.INDEX\_ID

JOIN ['+ @DBNAME +'].SYS.PARTITIONS AS P ON P.OBJECT\_ID = O.OBJECT\_ID

JOIN ['+ @DBNAME +'].SYS.ALLOCATION\_UNITS A ON P.PARTITION\_ID = A.CONTAINER\_ID

LEFT JOIN (SELECT DISTINCT A.OBJECT\_ID

FROM ['+ @DBNAME +'].SYS.COLUMNS A

JOIN ['+ @DBNAME +'].SYS.TYPES B ON A.USER\_TYPE\_ID= B.USER\_TYPE\_ID

WHERE (B.NAME IN (''type'', ''text'',''ntext'', ''image'', ''xml'')

OR (B.NAME IN (''varchar'', ''nvarchar'', ''varbinary'') AND A.MAX\_LENGTH=-1))

) SQ ON SQ.OBJECT\_ID = O.OBJECT\_ID

WHERE AVG\_FRAGMENTATION\_IN\_PERCENT >= '+ CAST(@MIN\_FRAG AS VARCHAR(8)) + '

AND AVG\_FRAGMENTATION\_IN\_PERCENT <= '+ CAST(@MAX\_FRAG AS VARCHAR(8)) + '

AND I.INDEX\_ID > 0

AND IS\_DISABLED = 0

AND IS\_HYPOTHETICAL = 0

GROUP BY O.NAME

,I.NAME

,S.NAME

,AVG\_FRAGMENTATION\_IN\_PERCENT

,V.INDEX\_TYPE\_DESC

,ALLOC\_UNIT\_TYPE\_DESC

,ISNULL(SQ.OBJECT\_ID,1)

HAVING SUM(TOTAL\_PAGES) >= ' + CAST(@TABSIZE AS VARCHAR(50)) + ''

IF @PRINT = 1

PRINT @COMMAND

INSERT INTO #Frag\_Details

(DBNAME, OBJNAME, IDXNAME, SCHNAME, AVG\_FRAGMENTATION\_IN\_PERCENT, INDEXTYPE\_DMV, ALLOCUNITTYPE, LNGDATATYPE, TBLSIZE, CMPTLEVEL )

EXEC(@COMMAND)

SET @COMMAND = ''

FETCH NEXT FROM DBCheck into @DBNAME, @dbid1 ,@CMPTLEVEL

END

Close DBCheck

DEALLOCATE DBCheck

/\* .................... \*/

/\*\*\* FOR COMPATIBILITY LEVEL = 80 ONLY \*\*\*/

IF OBJECT\_ID('tempdb.dbo.##TempTables') IS NOT NULL DROP TABLE ##TempTables

SELECT TOP 0\* INTO ##TempTables FROM master.INFORMATION\_SCHEMA.TABLES

IF OBJECT\_ID('tempdb.dbo.##tmpSysObjects') IS NOT NULL DROP TABLE ##tmpSysObjects

SELECT TOP 0\* INTO ##tmpSysObjects FROM master..sysobjects

IF OBJECT\_ID('tempdb..##tmpSysUsers') IS NOT NULL DROP TABLE ##tmpSysUsers

SELECT TOP 0\* INTO ##tmpSysUsers FROM master..sysusers

--TABLE TO STORE FRAGMENTATION DETAILS

IF OBJECT\_ID('tempdb.DBO.##FragList') IS NOT NULL DROP TABLE ##FragList

IF EXISTS (SELECT \* FROM master..syscursors WHERE cursor\_name = 'DefragDBs')

DEALLOCATE DefragDBs

IF EXISTS (SELECT \* FROM master..syscursors WHERE cursor\_name = 'DefragTables')

DEALLOCATE DefragTables

IF EXISTS (SELECT \* FROM master..syscursors WHERE cursor\_name = 'DefragIndexes')

DEALLOCATE DefragIndexes

-- TEMPORARY TABLE TO STROE DBCC SHOWCONTING VALUE

CREATE TABLE ##FragList (

ObjectName CHAR (255), ObjectID INT, IndexName CHAR (255), IndexID INT, lvl INT,

CountPages INT, CountRows INT, MinRecSize INT, MaxRecSize INT, AvgRecSize INT,

ForRecCount INT, Extents INT, ExtentSwitches INT, AvgFreeBytes INT, AvgPageDensity INT,

ScanDensity DECIMAL, BestCount INT, ActualCount INT, LogicalFrag DECIMAL,

ExtentFrag DECIMAL)

-- CURSROR FOR DATABASES

DECLARE DefragDBs CURSOR FOR

Select DBName,DBID from #DBCheck where CMPTLEVEL=80

-- OPEN CURSOR

OPEN DefragDBs

-- LOOP THROUGH ALL DATABASES

FetchNext:

FETCH NEXT FROM DefragDBs INTO @DBNAME, @dbid1

WHILE @@FETCH\_STATUS = 0

BEGIN

TRUNCATE TABLE ##TempTables

SET @Exec\_Stmt = 'INSERT ##TempTables SELECT \* FROM [' + @DBNAME + '].INFORMATION\_SCHEMA.TABLES WHERE TABLE\_TYPE = ''BASE TABLE'''

EXEC (@Exec\_Stmt)

TRUNCATE TABLE ##tmpSysObjects

SET @Exec\_Stmt = 'INSERT ##tmpSysObjects SELECT \* FROM [' + @DBNAME + '].dbo.sysobjects'

EXEC (@Exec\_Stmt)

TRUNCATE TABLE ##tmpSysUsers

SET @Exec\_Stmt = 'INSERT ##tmpSysUsers SELECT \* FROM [' + @DBNAME + '].dbo.sysusers'

EXEC (@Exec\_Stmt)

-- CURSOR FOR BASE TABLES

DECLARE DefragTables CURSOR FOR

SELECT TABLE\_SCHEMA, TABLE\_NAME FROM ##TempTables WHERE TABLE\_NAME NOT LIKE '%tmp%'

TRUNCATE TABLE ##FragList

OPEN DefragTables

FETCH NEXT FROM DefragTables INTO @SCHEMANAME, @TableName

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @Exec\_Stmt = 'USE [' + @DBNAME + '] DBCC SHOWCONTIG (' + '''[' + RTRIM(@SCHEMANAME) + '].[' + RTRIM(@TableName) + ']''' + ') WITH FAST, TABLERESULTS, ALL\_INDEXES, NO\_INFOMSGS'

IF @PRINT = 1

PRINT @Exec\_Stmt

INSERT ##FragList EXEC (@Exec\_Stmt)

FETCH NEXT FROM DefragTables INTO @SCHEMANAME, @TableName

END

SET @Exec\_Stmt = 'DELETE FROM ##FragList WHERE (LogicalFrag <='+ cast(@MIN\_FRAG as varchar(5)) + ' OR LogicalFrag >= ' +

cast(@MAX\_FRAG as varchar(5)) +') OR CountPages < '+ Cast(@TABSIZE as varchar(5))+ ' OR IndexID IN (0,255)'

EXEC (@Exec\_Stmt)

CLOSE DefragTables

DEALLOCATE DefragTables

SET @COMMAND =

'SELECT ''['+ @DBNAME + ']'' DBNAME, ltrim(rtrim(ObjectName)) as OBJNAME,

IDXNAME = (select [name] from ['+ @DBNAME + '].dbo.sysindexes where id=F.ObjectID and indid = F.IndexID),

U.name as SCHNAME,

LogicalFrag ,

F.CountPages as TBLSIZE,

NULL , -- Since this part of code is for complt =80, there is no ONLINE indexing hence set INDEXTYPE\_DMV= NULL, ALLOCUNITTYPE= NULL and IS\_BLOB=0

NULL,

80 CMPTLEVEL

FROM ##FragList F

JOIN ##tmpSysObjects O ON F.ObjectID = O.id

INNER JOIN ##tmpSysUsers U ON O.uid = U.uid

ORDER BY LogicalFrag '

IF @PRINT = 1

PRINT @COMMAND

INSERT INTO #Frag\_Details

(DBNAME, OBJNAME, IDXNAME, SCHNAME, AVG\_FRAGMENTATION\_IN\_PERCENT, TBLSIZE, INDEXTYPE\_DMV, ALLOCUNITTYPE, CMPTLEVEL )

EXEC(@COMMAND)

SET @COMMAND = ''

GOTO FetchNext

END

-- CLOSE AND DEALLOCATE CURSOR FOR DATABASES

CLOSE DefragDBs

DEALLOCATE DefragDBs

-- REINDEXING DATABASES BASED ON COMPATIBILITY LEVEL

DECLARE TAB CURSOR FOR

SELECT DBNAME, OBJNAME, IDXNAME, SCHNAME,INDEXTYPE\_DMV, ALLOCUNITTYPE, LNGDATATYPE, CMPTLEVEL FROM #Frag\_Details;

OPEN TAB;

FETCH NEXT FROM TAB

INTO @DBNAME, @OBJECTNAME, @INDEXNAME, @SCHEMANAME, @INDEXTYPE\_DMV, @ALLOCUNITTYPE, @LNGDATATYPE, @CMPTLEVEL;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF @CMPTLEVEL >= 90 -- FOR COMPATIBILITY LEVEL = 90 or 100

BEGIN

IF @SORTINTEMPDB = 1 -- SORTED IN TEMPDB

SET @SORT\_IN\_TEMPDB = 'ON'

ELSE

SET @SORT\_IN\_TEMPDB = 'OFF'

-- Setting @IS\_BLOB to 1 to do rebuild offline for below condition

IF @PRINT = 1

PRINT 'ObjectName: ' + @OBJECTNAME + ', LongDataType: ' + cast(@LNGDATATYPE as varchar(15))

IF (@INDEXTYPE\_DMV = 'XML INDEX' OR @ALLOCUNITTYPE = 'LOB\_DATA' OR @LNGDATATYPE <> 1)

SET @IS\_BLOB = 1

ELSE

SET @IS\_BLOB = 0

IF @INDEXTYPE = 'REBUILD'

BEGIN

IF @REBUILDOPT = 'ONLINE'

SET @COMMAND = N'ALTER INDEX [' + @INDEXNAME + N'] ON [' + @DBNAME + N'].[' + @SCHEMANAME + N'].[' + @OBJECTNAME + N'] REBUILD WITH (FILLFACTOR = ' + CAST(@FILLFACT AS VARCHAR(3)) + ', ONLINE = ' + CASE WHEN @IS\_BLOB = 1 THEN ' OFF ' ELSE ' ON ' END + ', SORT\_IN\_TEMPDB = ON)';

ELSE

SET @COMMAND = N'ALTER INDEX [' + @INDEXNAME + N'] ON [' + @DBNAME + N'].[' + @SCHEMANAME + N'].[' + @OBJECTNAME + N'] REBUILD WITH (FILLFACTOR = ' + CAST(@FILLFACT AS VARCHAR(3)) + ', ONLINE = OFF, SORT\_IN\_TEMPDB = ON)';

END

ELSE

IF @IS\_BLOB = 1

SET @COMMAND = N'ALTER INDEX [' + @INDEXNAME + N'] ON [' + @DBNAME + N'].[' + @SCHEMANAME + N'].[' + @OBJECTNAME + N'] REORGANIZE WITH (LOB\_COMPACTION = ON)'

ELSE

SET @COMMAND = N'ALTER INDEX [' + @INDEXNAME + N'] ON [' + @DBNAME + N'].[' + @SCHEMANAME + N'].[' + @OBJECTNAME + N'] REORGANIZE';

END

ELSE -- FOR COMPATIBILITY LEVEL = 80

BEGIN

IF @INDEXTYPE = 'REBUILD'

BEGIN

SET @COMMAND = 'DBCC DBREINDEX (''' + @DBNAME +'.' + RTRIM(@SCHEMANAME) + '.' + RTRIM(@OBJECTNAME) + ''', [' + @INDEXNAME+'],' + CAST(@FILLFACT AS VARCHAR(3)) + ')';

END

ELSE

BEGIN

SET @COMMAND = 'DBCC INDEXDEFRAG ([' + @DBNAME +'], ['+ RTRIM(@SCHEMANAME) +'.'+ RTRIM(@OBJECTNAME) +'], [' +@INDEXNAME+'])';

END

END

IF @TLOG\_FLAG = 1 -- Following code will check the log file size and kick-off the log backup job

BEGIN

IF OBJECT\_ID('tempdb.dbo.#DBLogSize') IS NOT NULL DROP TABLE #DBLogSize

CREATE TABLE #DBLogSize (DBName VARCHAR(200), LogSize DECIMAL(15,4), LogSpaceUsedPrcnt DECIMAL(15,4), status INT)

SET @sql= 'DBCC SQLPERF(LOGSPACE) WITH NO\_INFOMSGS'

INSERT #DBLogSize

EXEC(@sql)

SELECT @Logsize= LogSize \* LogSpaceUsedPrcnt/100 FROM #DBLogSize WHERE dbname= @DBNAME

IF @Logsize > @LOGLIMIT -- if log size of the accoutnow database is > 5GB, Log backup job will be kicked-off

BEGIN

EXEC msdb..DV\_FindHungJob @Job\_Name = @TLOG\_NAME, @Job\_owner = 1, @is\_sysadmin = 1, @Jobstatus = @jobst OUTPUT

IF (@jobst != 1 ) -- means (Transaction log job is not running)

BEGIN

EXEC msdb..sp\_start\_job @job\_name = @TLOG\_NAME --'Transaction Log Backup Job for DB Maintenance Plan TLog Backup - Every 2 hours daily'

PRINT @TLOG\_NAME + ' job started.'

WAITFOR DELAY '0:00:30' -- Wait for 30 sec, so that Log backup get completed, before moving ahead.

END

END

END

IF @PRINT = 1

PRINT @COMMAND;

IF @EXECUTE = 1

EXEC (@COMMAND);

FETCH NEXT FROM TAB

INTO @DBNAME, @OBJECTNAME, @INDEXNAME, @SCHEMANAME, @INDEXTYPE\_DMV, @ALLOCUNITTYPE, @LNGDATATYPE, @CMPTLEVEL;

END;

CLOSE TAB;

DEALLOCATE TAB

IF OBJECT\_ID('tempdb.dbo.##TempTables') IS NOT NULL DROP TABLE ##TempTables

IF OBJECT\_ID('tempdb.dbo.##tmpSysObjects') IS NOT NULL DROP TABLE ##tmpSysObjects

IF OBJECT\_ID('tempdb..##tmpSysUsers') IS NOT NULL DROP TABLE ##tmpSysUsers

END

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

--Put the below part in job step for looping through user databases

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

DECLARE @DBNAME VARCHAR(100)

DECLARE DBLIST\_CUR CURSOR FOR

SELECT NAME FROM SYS.DATABASES WHERE DATABASE\_ID > 4 and NAME NOT IN ('ReportServer','ReportServerTempDB')

OPEN DBLIST\_CUR

FETCH NEXT FROM DBLIST\_CUR

INTO @DBNAME

WHILE @@FETCH\_STATUS = 0

BEGIN

-- REORGANIZING THE INDEXES

EXEC DV\_INDEX\_MAINTAINENCE

@DBNAME = @DBNAME

, @MIN\_FRAG = 30

, @MAX\_FRAG = 60

, @TABSIZE = 1000

, @INDEXTYPE = 'REORGANIZE'

, @FILLFACT = 100

, @REBUILDOPT = 'ONLINE'

, @SORTINTEMPDB = 1

, @LOB\_COMPACTION = 'ON'

, @TLOG\_FLAG = 0

, @TLOG\_NAME = NULL

, @LOGLIMIT = 3072

, @PRINT = 1

, @EXECUTE = 0

waitfor delay '0:00:30'

-- REBUILDING THE INDEXES

EXEC DV\_INDEX\_MAINTAINENCE

@DBNAME = @DBNAME

, @MIN\_FRAG = 60

, @MAX\_FRAG = 100

, @TABSIZE = 1000

, @INDEXTYPE = 'REBUILD'

, @FILLFACT = 100

, @REBUILDOPT = 'ONLINE'

, @SORTINTEMPDB = 1

, @LOB\_COMPACTION = 'ON'

, @TLOG\_FLAG = 0

, @TLOG\_NAME = NULL

, @LOGLIMIT = 3072

, @PRINT = 1

, @EXECUTE = 0

FETCH NEXT FROM DBLIST\_CUR

INTO @DBNAME

END

CLOSE DBLIST\_CUR

DEALLOCATE DBLIST\_CUR

## DUPLICATE INDEX SCRIPT FOR 2000/2005

SELECT tbl.[name] AS TableName, idx.[name] AS IndexName,

INDEXPROPERTY( tbl.[id], idx.[name], 'IsClustered') AS IsClustered,

INDEXPROPERTY( tbl.[id], idx.[name], 'IsUnique') AS IsUnique,

CASE WHEN (idx.status & 2048) <> 0 THEN 'Yes' ELSE '' END AS IsPrimaryKey,

ISNULL( '[' + INDEX\_COL( user\_name(tbl.uid)+ '.' + tbl.[name], idx.indid, 1) + ']', '') +

CASE WHEN INDEXKEY\_PROPERTY(tbl.[id], idx.indid, 1, N'isdescending') = 1 THEN ' DESC ' ELSE '' END AS Col1,

ISNULL( '[' + INDEX\_COL( user\_name(tbl.uid)+ '.' + tbl.[name], idx.indid, 2) + ']', '') +

CASE WHEN INDEXKEY\_PROPERTY(tbl.[id], idx.indid, 2, N'isdescending') = 1 THEN ' DESC ' ELSE '' END AS Col2,

ISNULL( '[' + INDEX\_COL( user\_name(tbl.uid)+ '.' + tbl.[name], idx.indid, 3) + ']', '') +

CASE WHEN INDEXKEY\_PROPERTY(tbl.[id], idx.indid, 3, N'isdescending') = 1 THEN ' DESC ' ELSE '' END AS Col3,

ISNULL( '[' + INDEX\_COL( user\_name(tbl.uid)+ '.' + tbl.[name], idx.indid, 4) + ']', '') +

CASE WHEN INDEXKEY\_PROPERTY(tbl.[id], idx.indid, 4, N'isdescending') = 1 THEN ' DESC ' ELSE '' END AS Col4,

ISNULL( '[' + INDEX\_COL( user\_name(tbl.uid)+ '.' + tbl.[name], idx.indid, 5) + ']', '') +

CASE WHEN INDEXKEY\_PROPERTY(tbl.[id], idx.indid, 5, N'isdescending') = 1 THEN ' DESC ' ELSE '' END AS Col5,

ISNULL( '[' + INDEX\_COL( user\_name(tbl.uid)+ '.' + tbl.[name], idx.indid, 6) + ']', '') +

CASE WHEN INDEXKEY\_PROPERTY(tbl.[id], idx.indid, 6, N'isdescending') = 1 THEN ' DESC ' ELSE '' END AS Col6,

ISNULL( '[' + INDEX\_COL( user\_name(tbl.uid)+ '.' + tbl.[name], idx.indid, 7) + ']', '') +

CASE WHEN INDEXKEY\_PROPERTY(tbl.[id], idx.indid, 7, N'isdescending') = 1 THEN ' DESC ' ELSE '' END AS Col7,

ISNULL( '[' + INDEX\_COL( user\_name(tbl.uid)+ '.' + tbl.[name], idx.indid, 8) + ']', '') +

CASE WHEN INDEXKEY\_PROPERTY(tbl.[id], idx.indid, 8, N'isdescending') = 1 THEN ' DESC ' ELSE '' END AS Col8,

ISNULL( '[' + INDEX\_COL( user\_name(tbl.uid)+ '.' + tbl.[name], idx.indid, 9) + ']', '') +

CASE WHEN INDEXKEY\_PROPERTY(tbl.[id], idx.indid, 9, N'isdescending') = 1 THEN ' DESC ' ELSE '' END AS Col9,

ISNULL( '[' + INDEX\_COL( user\_name(tbl.uid)+ '.' + tbl.[name], idx.indid, 10) + ']', '') +

CASE WHEN INDEXKEY\_PROPERTY(tbl.[id], idx.indid, 10, N'isdescending') = 1 THEN ' DESC ' ELSE '' END AS Col10,

ISNULL( '[' + INDEX\_COL( user\_name(tbl.uid)+ '.' + tbl.[name], idx.indid, 11) + ']', '') +

CASE WHEN INDEXKEY\_PROPERTY(tbl.[id], idx.indid, 11, N'isdescending') = 1 THEN ' DESC ' ELSE '' END AS Col11,

ISNULL( '[' + INDEX\_COL( user\_name(tbl.uid)+ '.' + tbl.[name], idx.indid, 12) + ']', '') +

CASE WHEN INDEXKEY\_PROPERTY(tbl.[id], idx.indid, 12, N'isdescending') = 1 THEN ' DESC ' ELSE '' END AS Col12,

ISNULL( '[' + INDEX\_COL( user\_name(tbl.uid)+ '.' + tbl.[name], idx.indid, 13) + ']', '') +

CASE WHEN INDEXKEY\_PROPERTY(tbl.[id], idx.indid, 13, N'isdescending') = 1 THEN ' DESC ' ELSE '' END AS Col13,

ISNULL( '[' + INDEX\_COL( user\_name(tbl.uid)+ '.' + tbl.[name], idx.indid, 14) + ']', '') +

CASE WHEN INDEXKEY\_PROPERTY(tbl.[id], idx.indid, 14, N'isdescending') = 1 THEN ' DESC ' ELSE '' END AS Col14,

ISNULL( '[' + INDEX\_COL( user\_name(tbl.uid)+ '.' + tbl.[name], idx.indid, 15) + ']', '') +

CASE WHEN INDEXKEY\_PROPERTY(tbl.[id], idx.indid, 15, N'isdescending') = 1 THEN ' DESC ' ELSE '' END AS Col15,

ISNULL( '[' + INDEX\_COL( user\_name(tbl.uid)+ '.' + tbl.[name], idx.indid, 16) + ']', '') +

CASE WHEN INDEXKEY\_PROPERTY(tbl.[id], idx.indid, 16, N'isdescending') = 1 THEN ' DESC ' ELSE '' END AS Col16,

INDEXPROPERTY( tbl.[id], idx.[name], 'IsStatistics') AS IsStats,

INDEXPROPERTY( tbl.[id], idx.[name], 'IsAutoStatistics') AS IsAutoStats,

INDEXPROPERTY( tbl.[id], idx.[name], 'IsHypothetical') AS IsHypothetical,

dpages,

used,

rowcnt

INTO #t1

FROM sysindexes idx

INNER JOIN sysobjects tbl ON idx.[id] = tbl.[id]

JOIN sysusers su ON su.[uid] = tbl.[uid]

WHERE indid > 0 AND INDEXPROPERTY( tbl.[id], idx.[name], 'IsStatistics') = 0

SELECT l1.tablename,

l1.indexname,

b.indexname AS overlappingIndex,

l1.IsClustered,

l1.IsUnique,

l1.IsPrimaryKey,

l1.Col1 + CASE WHEN LEN(LTRIM(RTRIM(l1.Col2))) > 0 THEN ', ' ELSE '' END + l1.Col2 +

CASE WHEN LEN(LTRIM(RTRIM(l1.Col3))) > 0 THEN ', ' ELSE '' END + l1.Col3 +

CASE WHEN LEN(LTRIM(RTRIM(l1.Col4))) > 0 THEN ', ' ELSE '' END + l1.Col4 +

CASE WHEN LEN(LTRIM(RTRIM(l1.Col5))) > 0 THEN ', ' ELSE '' END + l1.Col5 +

CASE WHEN LEN(LTRIM(RTRIM(l1.Col6))) > 0 THEN ', ' ELSE '' END + l1.Col6 +

CASE WHEN LEN(LTRIM(RTRIM(l1.Col7))) > 0 THEN ', ' ELSE '' END + l1.Col7 +

CASE WHEN LEN(LTRIM(RTRIM(l1.Col8))) > 0 THEN ', ' ELSE '' END + l1.Col8 +

CASE WHEN LEN(LTRIM(RTRIM(l1.Col9))) > 0 THEN ', ' ELSE '' END + l1.Col9 +

CASE WHEN LEN(LTRIM(RTRIM(l1.Col10))) > 0 THEN ', ' ELSE '' END + l1.Col10 +

CASE WHEN LEN(LTRIM(RTRIM(l1.Col11))) > 0 THEN ', ' ELSE '' END + l1.Col11 +

CASE WHEN LEN(LTRIM(RTRIM(l1.Col12))) > 0 THEN ', ' ELSE '' END + l1.Col12 +

CASE WHEN LEN(LTRIM(RTRIM(l1.Col13))) > 0 THEN ', ' ELSE '' END + l1.Col13 +

CASE WHEN LEN(LTRIM(RTRIM(l1.Col14))) > 0 THEN ', ' ELSE '' END + l1.Col14 +

CASE WHEN LEN(LTRIM(RTRIM(l1.Col15))) > 0 THEN ', ' ELSE '' END + l1.Col15 +

CASE WHEN LEN(LTRIM(RTRIM(l1.Col16))) > 0 THEN ', ' ELSE '' END + l1.Col16 as IndexKey,

l1.dpages,

l1.used,

l1.rowcnt

INTO #dupindex

FROM #t1 l1

INNER JOIN #t1 b ON l1.tablename = b.tablename

AND l1.indexname <> b.indexname

AND l1.col1 = b.col1

AND (LEN(LTRIM(LTRIM(l1.col2))) = 0 OR LEN(LTRIM(LTRIM(b.col2))) = 0 OR l1.col2 = b.col2)

AND (LEN(LTRIM(LTRIM(l1.col3))) = 0 OR LEN(LTRIM(LTRIM(b.col3))) = 0 OR l1.col3 = b.col3)

AND (LEN(LTRIM(LTRIM(l1.col4))) = 0 OR LEN(LTRIM(LTRIM(b.col4))) = 0 OR l1.col4 = b.col4)

AND (LEN(LTRIM(LTRIM(l1.col5))) = 0 OR LEN(LTRIM(LTRIM(b.col5))) = 0 OR l1.col5 = b.col2)

AND (LEN(LTRIM(LTRIM(l1.col6))) = 0 OR LEN(LTRIM(LTRIM(b.col6))) = 0 OR l1.col6 = b.col2)

AND (LEN(LTRIM(LTRIM(l1.col7))) = 0 OR LEN(LTRIM(LTRIM(b.col7))) = 0 OR l1.col7 = b.col2)

AND (LEN(LTRIM(LTRIM(l1.col8))) = 0 OR LEN(LTRIM(LTRIM(b.col8))) = 0 OR l1.col8 = b.col2)

AND (LEN(LTRIM(LTRIM(l1.col9))) = 0 OR LEN(LTRIM(LTRIM(b.col9))) = 0 OR l1.col9 = b.col9)

AND (LEN(LTRIM(LTRIM(l1.col10))) = 0 OR LEN(LTRIM(LTRIM(b.col10))) = 0 OR l1.col10 = b.col10)

AND (LEN(LTRIM(LTRIM(l1.col11))) = 0 OR LEN(LTRIM(LTRIM(b.col11))) = 0 OR l1.col11 = b.col11)

AND (LEN(LTRIM(LTRIM(l1.col12))) = 0 OR LEN(LTRIM(LTRIM(b.col12))) = 0 OR l1.col12 = b.col12)

AND (LEN(LTRIM(LTRIM(l1.col13))) = 0 OR LEN(LTRIM(LTRIM(b.col13))) = 0 OR l1.col13 = b.col13)

AND (LEN(LTRIM(LTRIM(l1.col14))) = 0 OR LEN(LTRIM(LTRIM(b.col14))) = 0 OR l1.col14 = b.col14)

AND (LEN(LTRIM(LTRIM(l1.col15))) = 0 OR LEN(LTRIM(LTRIM(b.col15))) = 0 OR l1.col15 = b.col15)

AND (LEN(LTRIM(LTRIM(l1.col16))) = 0 OR LEN(LTRIM(LTRIM(b.col16))) = 0 OR l1.col16 = b.col16)

WHERE l1.dpages > 100 AND l1.tablename NOT LIKE 'sys%'

ORDER BY l1.tablename, l1.IsClustered DESC, IndexKey

SELECT \* FROM #dupindex

drop table #t1

drop table #dupindex

## Script to get index fragmentation and last update stats status

DECLARE @IndexTable TABLE

(

[Database] sysname, [Table] sysname, [Index Name] sysname NULL, index\_id smallint,

[object\_id] INT, [Index Type] VARCHAR(20), [Alloc Unit Type] VARCHAR(20),

[Avg Frag %] decimal(5,2), [Row Ct] bigint, [Stats Update Dt] datetime

)

DECLARE @dbid smallint --Database id for current database

DECLARE @objectid INT --Object id for table being analyzed

DECLARE @indexid INT --Index id for the target index for the STATS\_DATE() function

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

-- \*\*\*\*\*\*VARIABLE ASSIGNMENTS\*\*\*\*\*\*

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

SELECT @dbid = DB\_ID(DB\_NAME())

--SELECT @objectid = OBJECT\_ID(@table\_name)

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

-- \*\*\*\*\*\*Load @IndexTable with Index Metadata\*\*\*\*\*\*

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

INSERT INTO @IndexTable

(

[Database], [Table], [Index Name], index\_id, [object\_id],

[Index Type], [Alloc Unit Type], [Avg Frag %], [Row Ct], [Stats Update Dt]

)

SELECT

DB\_NAME() AS "Database",

object\_name(IPS.OBJECT\_ID) AS "Table" ,

SI.NAME AS "Index Name",

IPS.index\_id, IPS.OBJECT\_ID, --These fields included for joins only

IPS.index\_type\_desc, --Heap, Non-clustered, or Clustered

IPS.alloc\_unit\_type\_desc, --In-row data or BLOB data

CAST(IPS.avg\_fragmentation\_in\_percent AS decimal(5,2)),

IPS.record\_count,

STATS\_DATE(IPS.OBJECT\_ID, IPS.index\_id)

FROM sys.dm\_db\_index\_physical\_stats (@dbid, NULL, NULL, NULL, 'sampled') IPS

LEFT JOIN sys.sysindexes SI ON IPS.OBJECT\_ID = SI.id AND IPS.index\_id = SI.indid

WHERE IPS.index\_id <> 0

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

-- \*\*\*\*\*\*RETURN RESULTS\*\*\*\*\*\*

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

SELECT I.[Database], I.[Table], I.[Index Name], "Index Type"=

CASE I.[Index Type]

WHEN 'NONCLUSTERED INDEX' THEN 'NCLUST'

WHEN 'CLUSTERED INDEX' THEN 'CLUST'

ELSE 'HEAP'

END,

I.[Avg Frag %], I.[Row Ct],

CONVERT(VARCHAR, I.[Stats Update Dt], 110) AS "Stats Dt"

FROM @IndexTable I

ORDER BY I.[Index Type], I.[index\_id]

## Script to move indexes on different file group

CREATE PROC uspMoveTableToFileGroup

@TblName VARCHAR(200),

@tblFileGroupName VARCHAR(200),

@indFileGroupName VARCHAR(200) AS

SET NOCOUNT ON

create table #indx (index\_name VARCHAR(200), index\_description VARCHAR(200), index\_keys VARCHAR(200))

INSERT #indx (index\_name , index\_description , index\_keys) EXEC sp\_helpindex @TblName

--SELECT 'SELECT ' + '''' + 'Table :' + @TblName + ''''

--SELECT 'SELECT ' + '''' + 'Start Time : ' + '''' + ' + CONVERT(char(25), GETDATE(), 121)' +

-- CHAR(13) + ' GO '

SELECT 'DECLARE @StartDt DATETIME'

SELECT 'SET @StartDt = GETDATE()'

SELECT 'CREATE ' +

CASE WHEN CHARINDEX('UNIQUE', index\_description, 0) > 0 THEN 'UNIQUE ' ELSE '' END +

CASE WHEN SUBSTRING(index\_description,1,3) = 'non' THEN 'NONCLUSTERED ' ELSE 'CLUSTERED ' END +

' INDEX ' +

index\_name + CHAR(13) + ' ON ' + 'dbo.[' + @TblName + ']' +

'(' + index\_keys + ')' + ' WITH DROP\_EXISTING ON '

+ '[' + CASE WHEN SUBSTRING(index\_description,1,3) = 'non' THEN @indFileGroupName ELSE @tblFileGroupName END + ']'

FROM #indx

-- SELECT 'SELECT ' + '''' + 'Table :' + @TblName + '''' + '+' + '''' + ' ' + '''' +

-- + '+' + '''' + 'Start Time : ' + '''' + ' + CONVERT(char(25), @StartDt, 121)'

-- + '+' + '''' + 'End Time : ' + '''' + ' + CONVERT(char(25), GETDATE(), 121)' + CHAR(13) + ' GO '

SELECT 'SELECT ' + '''' + @TblName + ', ' + '''' + ' + CONVERT(char(25), @StartDt, 121)' + '+' +

'''' + ', ' + '''' +

' + CONVERT(char(25), GETDATE(), 121) ' + '+' + '''' + ', ' + '''' + '+' +

' CAST(DATEDIFF(ss, @StartDt, GETDATE()) AS VARCHAR(20))' + CHAR(13) + ' GO '

/\*

uspMoveTableToFileGroup 'departments', 'Primary', 'INDEX'

\*/

## Script to find unused Indexes

DECLARE @dbid INT

SELECT @dbid = DB\_ID(DB\_NAME())

SELECT OBJECTNAME = OBJECT\_NAME(I.OBJECT\_ID),

INDEXNAME = I.NAME,

I.INDEX\_ID

FROM SYS.INDEXES I

JOIN SYS.OBJECTS O

ON I.OBJECT\_ID = O.OBJECT\_ID

WHERE OBJECTPROPERTY(O.OBJECT\_ID,'IsUserTable') = 1

AND I.INDEX\_ID NOT IN (

SELECT S.INDEX\_ID

FROM SYS.DM\_DB\_INDEX\_USAGE\_STATS S

WHERE S.OBJECT\_ID = I.OBJECT\_ID

AND I.INDEX\_ID = S.INDEX\_ID

AND DATABASE\_ID = @dbid)

ORDER BY OBJECTNAME,

I.INDEX\_ID,

INDEXNAME ASC

GO

## Drop Hypothetical Index

DECLARE @strSQL nvarchar(1024)

DECLARE @objid int

DECLARE @indid tinyint

DECLARE ITW\_Stats

CURSOR FOR SELECT id, indid FROM sysindexes

ORDER BY name

OPEN ITW\_Stats

FETCH NEXT FROM ITW\_Stats INTO @objid, @indid

WHILE (@@FETCH\_STATUS <> -1)

BEGIN

SELECT @strSQL = (SELECT case when INDEXPROPERTY(i.id, i.name, 'IsStatistics') = 1

then 'drop statistics [' else 'drop index [' end + OBJECT\_NAME(i.id) + '].[' + i.name + ']'

FROM sysindexes i join sysobjects o on i.id = o.id WHERE i.id = @objid and i.indid = @indid

AND(INDEXPROPERTY(i.id, i.name, 'IsHypothetical') = 1 OR(INDEXPROPERTY(i.id, i.name, 'IsStatistics') = 1

AND INDEXPROPERTY(i.id, i.name, 'IsAutoStatistics') = 0)))

IF LTRIM(RTRIM(@strSQL)) <> ''

PRINT(@strSQL)

FETCH NEXT FROM ITW\_Stats INTO @objid, @indid

END

CLOSE ITW\_Stats

DEALLOCATE ITW\_Stats

## Update Statistics on all databases

set nocount on

declare @db varchar(300), @rm sql\_variant, @script varchar(4000)

if object\_id('tempdb.dbo.#databases') is not null drop table #databases

select name as dbname--, databasepropertyex(name, 'recovery') recoverymodel

into #databases

from master..sysdatabases with (nolock)

where databasepropertyex(name, 'status') <> 'offline' and

databasepropertyex(name, 'status') <> 'loading' and

name not in ('master', 'distribution', 'msdb', 'model', 'tempdb', 'litespeedlocal')

while 1 = 1 begin

select top 1 @db = dbname --, @rm = recoverymodel

from #databases

--where dbname > @db

order by dbname

if @@rowcount <> 1 or @db is null break

-- rebuild indexes

set @script = 'use ['+ @db + '];

print ''update stats for database :'' + db\_name()+ char(13)' + '

declare @table\_name varchar(1000), @sql nvarchar(4000), @owner\_name varchar(100)

declare c1 cursor for

select u.name owner\_name, s.name table\_name

from sysobjects s

JOIN sys.schemas u

on s.uid=u.schema\_id

where xtype = ''U'' and datediff (hh,crdate,getdate())>24

open c1

fetch next from c1 into @owner\_name, @table\_name

while @@fetch\_status = 0

begin

select @sql = ''update statistics '' + @owner\_name + ''.[''+ @table\_name +''] with fullscan''

print @sql

exec sp\_executesql @sql

fetch next from c1 into @owner\_name, @table\_name

end

close c1

deallocate c1'

exec (@script)

delete from #databases where dbname = @db

end

# Wait Stats

## Script to collect wait stats

IF EXISTS (SELECT \* FROM dbo.sysobjects WHERE id = object\_id(N'[dbo].[get\_waitstats]')

AND OBJECTPROPERTY(id, N'IsProcedure') = 1)

DROP procedure [dbo].[get\_waitstats]

GO

CREATE proc get\_waitstats

AS

-- This procedure creates a waitstats report that lists wait types by percentage.

-- You can run the procedure while track\_waitstats is executing.

SET nocount ON

DECLARE @now datetime,@totalwait numeric(20,1)

,@endtime datetime,@begintime datetime

,@hr int,@min int,@sec int

SELECT @now=max(now),@begintime=min(now),@endtime=max(now)

FROM waitstats WHERE [wait type] = 'Total'

-- Subtract waitfor, sleep, and resource\_queue from total.

SELECT @totalwait = sum([wait time]) + 1 FROM waitstats

WHERE [wait type] NOT IN ('WAITFOR','SLEEP','RESOURCE\_QUEUE', 'Total', '\*\*\*total\*\*\*') AND

now = @now

-- Insert adjusted totals and rank by percentage in descending order.

DELETE waitstats WHERE [wait type] = '\*\*\*total\*\*\*' AND now = @now

INSERT INTO waitstats SELECT '\*\*\*total\*\*\*',0,@totalwait,@totalwait,@now

SELECT [wait type],[wait time],percentage=cast (100\*[wait time]/@totalwait AS numeric(20,1))

FROM waitstats

WHERE [wait type] NOT IN ('WAITFOR','SLEEP','RESOURCE\_QUEUE','Total')

AND now = @now

ORDER BY percentage desc

GO

## Script to collect wait stats with delay

CREATE proc track\_waitstats (@num\_samples int=10,@delaynum int=1,@delaytype

nvarchar(10)='minutes')

AS

-- This stored procedure is provided AS IS with no warranties and confers no rights.

-- Use of included script samples are subject to the terms specified at

-- http://www.microsoft.com/info/cpyright.htm.

-- @num\_samples is the number of times to capture waitstats; default is 10 times.

-- @delaynum is the delay interval; can be in minutes or seconds; default is 1 minute.

-- @delaytype is the time specified. Values are "minutes" or "seconds."

-- Create waitstats table if it doesn't exist; otherwise truncate.

SET nocount ON

IF NOT EXISTS (SELECT 1 FROM sysobjects WHERE name = 'waitstats')

CREATE table waitstats ([wait type] varchar(80),

requests numeric(20,1),

[wait time] numeric (20,1),

[signal wait time] numeric(20,1),

now datetime default getdate())

ELSE truncate table waitstats

dbcc sqlperf (waitstats,clear) -- Clear out waitstats.

DECLARE @i int,@delay varchar(8),@dt varchar(3),@now datetime,@totalwait numeric(20,1)

,@endtime datetime,@begintime datetime,@hr int,@min int,@sec int

SELECT @i = 1

SELECT @dt = case lower(@delaytype)

WHEN 'minutes' THEN 'm'

WHEN 'minute' THEN 'm'

WHEN 'min' THEN 'm'

WHEN 'mm' THEN 'm'

WHEN 'mi' THEN 'm'

WHEN 'm' THEN 'm'

WHEN 'seconds' THEN 's'

WHEN 'second' THEN 's'

WHEN 'sec' THEN 's'

WHEN 'ss' THEN 's'

WHEN 's' THEN 's'

ELSE @delaytype

END

IF @dt NOT IN ('s','m')

BEGIN

PRINT 'please supply delay type e.g. seconds or minutes'

RETURN

END

IF @dt = 's'

BEGIN

SELECT @sec = @delaynum % 60

SELECT @min = cast((@delaynum / 60) AS int)

SELECT @hr = cast((@min / 60) AS int)

SELECT @min = @min % 60

END

IF @dt = 'm'

BEGIN

SELECT @sec = 0

SELECT @min = @delaynum % 60

SELECT @hr = cast((@delaynum / 60) AS int)

END

SELECT @delay= right('0'+ convert(varchar(2),@hr),2) + ':' +

+ right('0'+convert(varchar(2),@min),2) + ':' +

+ right('0'+convert(varchar(2),@sec),2)

IF @hr > 23 or @min > 59 or @sec > 59

BEGIN

SELECT 'hh:mm:ss delay time cannot > 23:59:59'

SELECT 'delay interval and type: ' + convert (varchar(10),@delaynum) + ',' + @delaytype + '

converts to ' + @delay

RETURN

END

WHILE (@i <= @num\_samples)

BEGIN

INSERT INTO waitstats ([wait type], requests, [wait time],[signal wait time])

EXEC ('dbcc sqlperf(waitstats)')

SELECT @i = @i + 1

waitfor delay @delay

END

-- Create report.

EXECUTE get\_waitstats

WITH Waits AS   
 (   
 SELECT    
   wait\_type,    
   wait\_time\_ms / 1000. AS wait\_time\_s,   
   100. \* wait\_time\_ms / SUM(wait\_time\_ms) OVER() AS pct,   
   ROW\_NUMBER() OVER(ORDER BY wait\_time\_ms DESC) AS rn   
 FROM sys.dm\_os\_wait\_stats   
 WHERE wait\_type    
   NOT IN   
     ('CLR\_SEMAPHORE', 'LAZYWRITER\_SLEEP', 'RESOURCE\_QUEUE',   
   'SLEEP\_TASK', 'SLEEP\_SYSTEMTASK', 'SQLTRACE\_BUFFER\_FLUSH', 'WAITFOR',   
   'CLR\_AUTO\_EVENT', 'CLR\_MANUAL\_EVENT')   
   ) -- filter out additional irrelevant waits   
      
SELECT W1.wait\_type,   
 CAST(W1.wait\_time\_s AS DECIMAL(12, 2)) AS wait\_time\_s,   
 CAST(W1.pct AS DECIMAL(12, 2)) AS pct,   
 CAST(SUM(W2.pct) AS DECIMAL(12, 2)) AS running\_pct   
FROM Waits AS W1   
 INNER JOIN Waits AS W2 ON W2.rn <= W1.rn   
GROUP BY W1.rn,    
 W1.wait\_type,    
 W1.wait\_time\_s,    
 W1.pct   
HAVING SUM(W2.pct) - W1.pct < 95; -- percentage threshold;

# Full Text Scripts

## Full text Catalog Information Query

Select

t.name as Table\_name,

fti.change\_tracking\_state\_desc ,

fti.crawl\_type\_desc,

fti.crawl\_start\_date,

fti.crawl\_end\_date,

it.name as internal\_table\_name,

it.object\_id as internal\_table\_id,

it.internal\_type\_desc

from sys.internal\_tables as it

Inner Join sys.fulltext\_indexes as fti

on it.parent\_id=fti.object\_id

Inner join sys.tables t

on t.object\_id=fti.object\_id

where it.internal\_type\_desc like 'FullText%'

Order by t.name

-----

Select

t.name as Table\_name,

fti.change\_tracking\_state\_desc ,

fti.crawl\_type\_desc,

fti.crawl\_start\_date,

fti.crawl\_end\_date,

ftp.status,

it.name as internal\_table\_name,

it.object\_id as internal\_table\_id,

it.internal\_type\_desc

from sys.internal\_tables as it

Inner Join sys.fulltext\_indexes as fti

on it.parent\_id=fti.object\_id

Inner join sys.tables t

on t.object\_id=fti.object\_id

Left Join sys.dm\_fts\_index\_population ftp

on ftp.table\_id=fti.object\_id

where it.internal\_type\_desc like 'FullText%'

Order by t.name

## Full Text Catalog Status and Catalog Item count comparison with table count

Declare @DbName varchar(100)

Declare @cmd varchar(1000)

Drop table #temp

Create table #temp

(

DatabaseName varchar(100),

CatalaogName Varchar(100),

Catalog\_ItemCount Int,

Catalog\_Status Int,

Change\_tracking\_state\_desc Varchar(50),

TableName varchar(100),

TableRowCount int

)

Declare DB\_cur cursor for

select name from sys.databases where Database\_id>4

open DB\_cur fetch next from DB\_cur into @DbName

while (@@fetch\_status = 0)

begin

--Print @cmd1

Set @cmd='USE '+@DbName+'

Insert into #temp

SELECT '''+@DbName+''' as DatbaseName,FT.name as CatalaogName,

fulltextcatalogproperty(FT.name, ''ItemCount'') CataLog\_Itemcount,

fulltextcatalogproperty(FT.name, ''PopulateStatus'')Catalog\_Status,

fti.change\_tracking\_state\_desc,

t.name as Table\_name,I.rows TableRowCount

from sys.fulltext\_indexes as fti

Inner Join sys.fulltext\_catalogs as FT

On FT.fulltext\_catalog\_id=fti.fulltext\_catalog\_id

Inner join sys.tables t

on t.object\_id=fti.object\_id

Join sysindexes I

On t.object\_id=I.id and fti.Unique\_index\_id=I.indid'

Exec(@cmd)

--Print @cmd

fetch next from DB\_cur into @DbName

end

close DB\_cur

deallocate DB\_cur

Select \*,Catalog\_status=Case when Catalog\_status=0 then 'Idle'

when Catalog\_status=1 then 'Full population in progress'

When Catalog\_status=3 then 'Paused' When Catalog\_status=3 then 'Throttled'

when Catalog\_status=5 then 'Shutdown' when Catalog\_status =6 then 'Incremental population in progress'

When Catalog\_status= 7 then 'Building index' when Catalog\_status=8 then 'Disk is full. Paused.'

when Catalog\_status=9 then 'Change tracking' End

from #temp

Order by 1

---

If you are rebuilding a Fulltext catalog verify the ACCENT\_SENSITIVITY

SELECT FULLTEXTCATALOGPROPERTY('ftCatalog', 'accentsensitivity');

--Returned 0, which means the catalog is not accent sensitive

GO

ALTER FULLTEXT CATALOG ftCatalog

REBUILD WITH ACCENT\_SENSITIVITY=ON; -- Accent sensitive

Alter FullText Index on dbo.tblAplSegB -- Full Population

START FULL POPULATION

GO

SELECT FULLTEXTCATALOGPROPERTY('tblAplSegB\_Cat', 'PopulateStatus'); -- Check Repopulation status

SELECT FULLTEXTCATALOGPROPERTY('tblAplSegA\_Cat', 'PopulateStatus');

when Catalog\_status=0 then 'Idle'

when Catalog\_status=1 then 'Full population in progress'

When Catalog\_status=3 then 'Paused'

When Catalog\_status=3 then 'Throttled'

when Catalog\_status=5 then 'Shutdown'

when Catalog\_status =6 then 'Incremental population in progress'

When Catalog\_status=7 then 'Building index'

when Catalog\_status=8 then 'Disk is full. Paused.'

when Catalog\_status=9 then 'Change tracking'

# Performance Scripts

## Get actual query currently running from sysprocess

DECLARE @handle binary (20)

SELECT @handle = sql\_handle FROM master..sysprocesses WHERE spid = SPID -- replace with actual numeric value

SELECT [text] FROM: fn\_get\_sql(@handle)

## Shows what individual SQL statements are currently executing

CREATE PROC [dbo].[dba\_WhatSQLIsExecuting]

AS

/\*--------------------------------------------------------------------

Purpose: Shows what individual SQL statements are currently executing.

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

Parameters: None.

Revision History:

24/07/2008 Ian\_Stirk@yahoo.com Initial version

Example Usage:

1. exec YourServerName.master.dbo.dba\_WhatSQLIsExecuting

---------------------------------------------------------------------\*/

BEGIN

-- Do not lock anything, and do not get held up by any locks.

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

-- What SQL Statements Are Currently Running?

SELECT [Spid] = session\_Id

, ecid

, [Database] = DB\_NAME(sp.dbid)

, [User] = nt\_username

, [Status] = er.status

, [Wait] = wait\_type

, [Individual Query] = SUBSTRING (qt.text,

er.statement\_start\_offset/2,

(CASE WHEN er.statement\_end\_offset = -1

THEN LEN(CONVERT(NVARCHAR(MAX), qt.text)) \* 2

ELSE er.statement\_end\_offset END -

er.statement\_start\_offset)/2)

,[Parent Query] = qt.text

, Program = program\_name

, Hostname

, nt\_domain

, start\_time

FROM sys.dm\_exec\_requests er

INNER JOIN sys.sysprocesses sp ON er.session\_id = sp.spid

CROSS APPLY sys.dm\_exec\_sql\_text(er.sql\_handle) as qt

WHERE session\_Id > 50 -- Ignore system spids.

AND session\_Id NOT IN (@@SPID) -- Ignore this current statement.

ORDER BY 1, 2

END

## SPID waiting for resource to be allocated

Get SPID that has been running for some time and perhaps waiting on some resource to be allocated.

This Requires VIEW\_sSERVER\_STATE permission to work

SELECT

SP.SPID, SP.status Status, SP.cmd Command, A.percent\_complete PercentComplete,

SP.Blocked BlockedBy, DB\_NAME(SP.dbid) AS DatabaseName, SP.hostname Hostname,

SP.physical\_io Physical\_IO, SP.cpu CPU, SP.waittype WaitType, (SP.waittime/1000) WaitTimeSec,

SP.lastwaittype LastWaitType, SP.waitresource WaitResource,

SP.login\_time LoginTime, A.total\_elapsed\_time/1000 BatchElapsedTimeSec,

(SELECT SUBSTRING(C.text,A.statement\_start\_offset/2,(CASE WHEN A.statement\_end\_offset = -1

THEN LEN(CONVERT(nvarchar(max), C.text)) \* 2 ELSE A.statement\_end\_offset END -A.statement\_start\_offset)/2)) SQLBatchText,

C.text SQLStatementText,

B.query\_plan as QueryPlan

FROM sys.sysprocesses AS SP

INNER JOIN sys.dm\_exec\_requests as A

ON SP.SPID = A.session\_id

CROSS APPLY sys.dm\_exec\_query\_plan(A.plan\_handle) as B

CROSS APPLY sys.dm\_exec\_sql\_text(A.sql\_handle) as C

WHERE SP.spid = @spid -- @spid replace spid with actual Numerical Spid

## Get Offending queries

Get Current resource consumption, query batch text, statement text, and XML query plan with high Logical reads, CPUtime, Elaspedtime and rowcount. Very useful to get running Queries when troubleshooting offending queries and sql server is hitting with performance issue.

SELECT

Sessions.session\_id AS SessionID, Requests.request\_id AS RequestID,

Requests.database\_id AS DatabaseID, databases.name AS DatabaseName,

Sessions.login\_name AS LoginName, Sessions.host\_name AS HostName, Sessions.program\_name AS ProgramName,

Sessions.client\_interface\_name AS ClientInterfaceName,

Requests.blocking\_session\_id AS BlockedBySessionID,

ISNULL(BlockRequests.BlockingRequestCount,0) AS BlockingRequestCount,

Requests.wait\_type AS WaitType,

Requests.wait\_time AS WaitTime, Requests.cpu\_time AS CPUTime, Requests.total\_elapsed\_time AS ElapsedTime,

Requests.reads AS Reads, Requests.writes AS Writes, Requests.logical\_reads AS LogicalReads,

dm\_os\_tasks.PendingIOCount, Requests.row\_count AS [RowCount],

Requests.granted\_query\_memory\*8 AS GrantedQueryMemoryKB,

CONVERT(BigInt, (Requests.cpu\_time+1))\*CONVERT(BigInt, (Requests.reads\*10+Requests.writes\*10+Requests.logical\_reads+1)) AS Score,

Statements.text AS BatchText,

LEN(Statements.text) AS BatchTextLength,

Requests.statement\_start\_offset/2 AS StatementStartPos,

CASE

WHEN Requests.statement\_end\_offset = -1 THEN LEN(CONVERT(nvarchar(MAX),Statements.text))\*2

ELSE Requests.statement\_end\_offset

END/2 AS StatementEndPos,

(CASE

WHEN Requests.statement\_end\_offset = -1 THEN LEN(CONVERT(nvarchar(MAX),Statements.text))\*2

ELSE Requests.statement\_end\_offset

END - Requests.statement\_start\_offset)/2 AS StatementTextLength,

CASE

WHEN Requests.sql\_handle IS NULL THEN ' '

ELSE

SubString(

Statements.text,

(Requests.statement\_start\_offset+2)/2,

(CASE

WHEN Requests.statement\_end\_offset = -1 THEN LEN(CONVERT(nvarchar(MAX),Statements.text))\*2

ELSE Requests.statement\_end\_offset

END - Requests.statement\_start\_offset)/2

)

END AS StatementText,

QueryPlans.query\_plan AS QueryPlan

FROM

sys.dm\_exec\_sessions AS Sessions

JOIN sys.dm\_exec\_requests AS Requests ON Sessions.session\_id=Requests.session\_id

LEFT OUTER JOIN sys.databases ON Requests.database\_id=databases.database\_id

LEFT OUTER JOIN (

SELECT blocking\_session\_id, COUNT(\*) AS BlockingRequestCount FROM sys.dm\_exec\_requests GROUP BY blocking\_session\_id

) AS BlockRequests ON Requests.session\_id=BlockRequests.blocking\_session\_id

LEFT OUTER JOIN (

SELECT request\_id, session\_id, SUM(pending\_io\_count) AS PendingIOCount

FROM sys.dm\_os\_tasks WITH (NOLOCK)

GROUP BY request\_id, session\_id

) AS dm\_os\_tasks ON

Requests.request\_id=dm\_os\_tasks.request\_id

AND Requests.session\_id=dm\_os\_tasks.session\_id

CROSS APPLY sys.dm\_exec\_sql\_text(sql\_handle) AS Statements

CROSS APPLY sys.dm\_exec\_query\_plan(plan\_handle) AS QueryPlans

ORDER BY score DESC

GO

## Missing or poorly formed indexes

-- Potentially Useful Indexes

select d.\*

, s.avg\_total\_user\_cost

, s.avg\_user\_impact

, s.last\_user\_seek

,s.unique\_compiles

from sys.dm\_db\_missing\_index\_group\_stats s

,sys.dm\_db\_missing\_index\_groups g

,sys.dm\_db\_missing\_index\_details d

where s.group\_handle = g.index\_group\_handle

and d.index\_handle = g.index\_handle

order by s.avg\_user\_impact desc

go

--- suggested index columns and usage

declare @handle int

select @handle = d.index\_handle

from sys.dm\_db\_missing\_index\_group\_stats s

,sys.dm\_db\_missing\_index\_groups g

,sys.dm\_db\_missing\_index\_details d

where s.group\_handle = g.index\_group\_handle

and d.index\_handle = g.index\_handle

select \*

from sys.dm\_db\_missing\_index\_columns(@handle)

order by column\_id

## Largest IO queries

--- top 20 statements by IO

SELECT TOP 20

(qs.total\_logical\_reads + qs.total\_logical\_writes) /qs.execution\_count as [Avg IO],

substring (qt.text,qs.statement\_start\_offset/2,

(case when qs.statement\_end\_offset = -1

then len(convert(nvarchar(max), qt.text)) \* 2

else qs.statement\_end\_offset end - qs.statement\_start\_offset)/2)

as query\_text,

qt.dbid,

qt.objectid

FROM sys.dm\_exec\_query\_stats qs

cross apply sys.dm\_exec\_sql\_text (qs.sql\_handle) as qt

ORDER BY [Avg IO] DESC

## Query plan reuse and DMVs

--- DMV reports statements with lowest plan reuse

---

SELECT TOP 50

qs.sql\_handle

,qs.plan\_handle

,cp.cacheobjtype

,cp.usecounts

,cp.size\_in\_bytes

,qs.statement\_start\_offset

,qs.statement\_end\_offset

,qt.dbid

,qt.objectid

,qt.text

,SUBSTRING(qt.text,qs.statement\_start\_offset/2,

(case when qs.statement\_end\_offset = -1

then len(convert(nvarchar(max), qt.text)) \* 2

else qs.statement\_end\_offset end -qs.statement\_start\_offset)/2)

as statement

FROM sys.dm\_exec\_query\_stats qs

cross apply sys.dm\_exec\_sql\_text(qs.sql\_handle) as qt

inner join sys.dm\_exec\_cached\_plans as cp on qs.plan\_handle=cp.plan\_handle

where cp.plan\_handle=qs.plan\_handle

and qt.dbid = db\_id() ----- put the database ID here

ORDER BY [Usecounts] ASC

## Get top SQL Statement Elapsed Time Stats for top XX Elapsed Time Plans

SELECT @@SERVERNAME AS [Host Name],

REPLACE(REPLACE(qt.text, CHAR(13), ''), CHAR(10), '') AS [Statement],

ps.plan\_total\_elapsed\_time\_ms AS [Plan Total Elapsed Time (milliseconds)],

ps.plan\_total\_execution\_count AS [Plan Total Executions],

ps.plan\_total\_elapsed\_time\_per\_execution\_ms AS [Plan Total Time Per Execution (milliseconds)],

ps.plan\_number\_of\_statements AS [Plan Number of Statements],

qs.total\_elapsed\_time \* .001 AS [Statement Total Elapsed Time (milliseconds)],

qs.execution\_count AS [Statement Executions] ,

(qs.total\_elapsed\_time \* .001) / qs.execution\_count AS [Statement Total Time Per Execution (milliseconds)],

qs.creation\_time AS [Plan Creation Time], qs.last\_execution\_time AS [Plan Last Execution Time]

-- , qs.\*

FROM sys.dm\_exec\_query\_stats qs

INNER JOIN (

SELECT TOP 100

SUM(total\_elapsed\_time) \* .001 AS plan\_total\_elapsed\_time\_ms,

(SUM(total\_elapsed\_time) \* .001) / SUM(execution\_count) AS plan\_total\_elapsed\_time\_per\_execution\_ms,

SUM(execution\_count) AS plan\_total\_execution\_count,

COUNT(\*) AS plan\_number\_of\_statements,

plan\_handle

FROM sys.dm\_exec\_query\_stats qs

GROUP BY plan\_handle

ORDER BY SUM(total\_worker\_time) DESC) AS ps

ON qs.plan\_handle = ps.plan\_handle

CROSS APPLY sys.dm\_exec\_sql\_text(qs.sql\_handle) AS qt

ORDER BY ps.plan\_total\_elapsed\_time\_ms DESC, qs.total\_elapsed\_time DESC

## Get TOP SQL Statement CPU Stats for top XX CPU Plans

SELECT @@SERVERNAME AS [Host Name],

REPLACE(REPLACE(qt.text, CHAR(13), ''), CHAR(10), '') AS [Statement],

ps.plan\_total\_cpu\_time\_ms AS [Plan Total CPU Time (milliseconds)],

ps.plan\_total\_execution\_count AS [Plan Total Executions],

ps.plan\_total\_cpu\_time\_per\_execution\_ms AS [Plan Total CPU Time Per Execution (milliseconds)],

ps.plan\_number\_of\_statements AS [Plan Number of Statements],

qs.total\_worker\_time \* .001 AS [Statement Total CPU Time (milliseconds)],

qs.execution\_count AS [Statement Executions] ,

(qs.total\_worker\_time \* .001) / qs.execution\_count AS [Statement CPU Time Per Execution (milliseconds)]

-- , \*

FROM sys.dm\_exec\_query\_stats qs

INNER JOIN (

SELECT TOP 100

SUM(total\_worker\_time) \* .001 AS plan\_total\_cpu\_time\_ms,

(SUM(total\_worker\_time) \* .001) / SUM(execution\_count) AS plan\_total\_cpu\_time\_per\_execution\_ms,

SUM(execution\_count) AS plan\_total\_execution\_count,

COUNT(\*) AS plan\_number\_of\_statements,

plan\_handle

FROM sys.dm\_exec\_query\_stats qs

GROUP BY plan\_handle

ORDER BY SUM(total\_worker\_time) DESC) AS ps

ON qs.plan\_handle = ps.plan\_handle

CROSS APPLY sys.dm\_exec\_sql\_text(qs.sql\_handle) AS qt

ORDER BY ps.plan\_total\_cpu\_time\_ms DESC, qs.total\_worker\_time DESC

## Database Properties (HTML)

set nocount on

declare @rcnt int

declare @cbody varchar(max), @sbody VARCHAR(MAX)

set @sbody = ''

set @sbody = @sbody + '<html><head><title> SQL maintenance and Configuration Standards </title></head>'

--print '<br></br>'

print '<body> <table border=1 align="center" cellpadding="1" cellspacing="0" width=98%>'

-- Database Properties

set @sbody = @sbody + '<h2> Database Properties </h2>'

--set @sbody = @sbody + '<font bgcolor="#FFCC99" face="arial" size="4"><b><u>Database Properties</b></u></font>'

set @sbody = @sbody + '<tr>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Database Name</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Recovery</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Compatibility Level</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Auto Close</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Auto Shrink</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Auto Create Statistics</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Auto Update Statistics</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Auto Update Statistics Async</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Cursor Close On Commit</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Local Cursor Default</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Date Correlation</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Parameterization Forced</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Page Verify Option</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Read Only</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">User Access</font></TH>

</tr>'

print @sbody

set @sbody = ''

declare rec CURSOR for

Select '<tr>

<td><font face="arial" size="2">' + a.[name] + '</font></td>

<td><font face="arial" size="2">' +

case when recovery\_model\_desc = 'SIMPLE' THEN '<font color="red"><b>' ELSE '' END

+ case recovery\_model when 1 then 'Full'

when 2 then 'Bulk\_logged'

else 'Simple' end +

case when recovery\_model\_desc = 'SIMPLE' THEN '</b></font>' ELSE '' END

+ '</font></td>

<td><font face="arial" size="2">' +

case when compatibility\_level <> 90 THEN '<font color="red"><b>' ELSE '' END

+ ltrim(str(compatibility\_level)) +

case when compatibility\_level <> 90 THEN '</b></font>' ELSE '' END

+ '</font></td>

<td><font face="arial" size="2">' +

case when Is\_Auto\_Close\_on = 1 THEN '<font color="red"><b>' ELSE '' END

+ ltrim(str(Is\_Auto\_Close\_on)) +

case when Is\_Auto\_Close\_on = 1 THEN '</b></font>' ELSE '' END

+ '</font></td>

<td><font face="arial" size="2">' +

case when Is\_Auto\_Shrink\_on = 1 THEN '<font color="red"><b>' ELSE '' END

+ ltrim(str(Is\_Auto\_Shrink\_on)) +

case when Is\_Auto\_Shrink\_on = 1 THEN '</b></font>' ELSE '' END

+ '</font></td>

<td><font face="arial" size="2">' +

case when is\_auto\_create\_stats\_on = 0 THEN '<font color="red"><b>' ELSE '' END

+ ltrim(str(is\_auto\_create\_stats\_on)) +

case when is\_auto\_create\_stats\_on = 0 THEN '</b></font>' ELSE '' END

+ '</font></td>

<td><font face="arial" size="2">' +

case when is\_auto\_update\_stats\_on = 0 THEN '<font color="red"><b>' ELSE '' END

+ ltrim(str(is\_auto\_update\_stats\_on)) +

case when is\_auto\_update\_stats\_on = 0 THEN '</b></font>' ELSE '' END

+ '</font></td>

<td><font face="arial" size="2">' +

case when is\_auto\_update\_stats\_async\_on = 1 THEN '<font color="red"><b>' ELSE '' END

+ ltrim(str(is\_auto\_update\_stats\_async\_on)) +

case when is\_auto\_update\_stats\_async\_on = 1 THEN '</b></font>' ELSE '' END

+ '</font></td>

<td><font face="arial" size="2">' +

case when is\_cursor\_close\_on\_commit\_on = 1 THEN '<font color="red"><b>' ELSE '' END

+ ltrim(str(is\_cursor\_close\_on\_commit\_on)) +

case when is\_cursor\_close\_on\_commit\_on = 1 THEN '</b></font>' ELSE '' END

+ '</font></td>

<td><font face="arial" size="2">' +

case when is\_local\_cursor\_default = 1 THEN '<font color="red"><b>' ELSE '' END

+ case is\_local\_cursor\_default when 1 then 'Local'

else 'Global' end +

case when is\_local\_cursor\_default = 1 THEN '</b></font>' ELSE '' END

+ '</font></td>

<td><font face="arial" size="2">' +

case when is\_date\_correlation\_on = 1 THEN '<font color="red"><b>' ELSE '' END

+ ltrim(str(is\_date\_correlation\_on)) +

case when is\_date\_correlation\_on = 1 THEN '</b></font>' ELSE '' END

+ '</font></td>

<td><font face="arial" size="2">' +

case when is\_parameterization\_forced = 1 THEN '<font color="red"><b>' ELSE '' END

+ case is\_parameterization\_forced when 1 then 'Forced'

else 'Simple' end +

case when is\_parameterization\_forced = 1 THEN '</b></font>' ELSE '' END

+ '</font></td>

<td><font face="arial" size="2">' +

case when page\_verify\_option <> 2 THEN '<font color="red"><b>' ELSE '' END

+ case page\_verify\_option when 0 then 'None'

when 1 then 'Torn\_page\_detection'

else 'Checksum' end +

case when page\_verify\_option <> 2 THEN '</b></font>' ELSE '' END

+ '</font></td>

<td><font face="arial" size="2">' +

case when is\_read\_only = 1 THEN '<font color="red"><b>' ELSE '' END

+ case is\_read\_only when 1 then 'Read\_Only'

else 'Read\_Write' end +

case when is\_read\_only = 1 THEN '</b></font>' ELSE '' END

+ '</font></td>

<td><font face="arial" size="2">' +

case when user\_access <> 0 THEN '<font color="red"><b>' ELSE '' END

+ case user\_access when 0 then 'Multi\_User'

when 1 then 'Single\_User'

else 'Restricted\_User' end +

case when user\_access <> 0 THEN '</b></font>' ELSE '' END

+ '</font></td>

</tr>'

from sys.databases a

left join syslogins b on a.owner\_sid = b.sid

where (

recovery\_model\_desc = 'SIMPLE' or

Is\_Auto\_Close\_on = 1 or

Is\_Auto\_Shrink\_on = 1 or

is\_auto\_create\_stats\_on = 0 or

is\_auto\_update\_stats\_on = 0 or

is\_auto\_update\_stats\_async\_on = 1 or

is\_cursor\_close\_on\_commit\_on = 1 or

is\_local\_cursor\_default = 1 or

is\_date\_correlation\_on = 1 or

is\_parameterization\_forced = 1 or

page\_verify\_option <> 2)

order by a.[name]

open rec

FETCH NEXT FROM rec into @cbody

WHILE (@@FETCH\_STATUS = 0)

begin

-- set @sbody = @sbody + @cbody

print @cbody

FETCH NEXT FROM rec into @cbody

end

close rec

deallocate rec

print '</table> </font>'

-- Database Properties end

-- Database Sizing

print '<h2> Database Sizing </h2>'

print 'The transaction log file should be initially sized to 10% of the initial data file size.[ Below marked in Red font Log Files size less then 10% of DataFiles]'

print '<br></br>'

--print 'The tempdb database will be initially sized to 20% of total user database volume. [ As Per the starndard TempDB is less than 20% of total DB Size]'

--print '<br></br>'

set @sbody = @sbody + '<table border=1 align="left" cellpadding="1" cellspacing="0" width=100%>'

set @sbody = @sbody + '<tr>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Database Name</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Data File Size (KB)</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Log File Size (KB)</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Percent Log Used</font></TH>

</tr>'

print @sbody

if object\_id('tempdb..#Size') is not null

drop table #Size

select

instance\_name

,max(case when ltrim(rtrim(counter\_name)) = 'Data File(s) Size (KB)' then cntr\_value else 0 end) as 'DataFileSize (KB)'

,max(case when ltrim(rtrim(counter\_name)) = 'Log File(s) Size (KB)' then cntr\_value else 0 end) as 'LogFileSize (KB)'

,max(case when ltrim(rtrim(counter\_name)) = 'Percent Log Used' then cntr\_value else 0 end) as 'PercentLogUsed'

into #Size

from sys.dm\_os\_performance\_counters -- SS2k5\8

where object\_name like '%:Databases%'

and counter\_name

in ('Data File(s) Size (KB)', 'Log File(s) Size (KB)', 'Percent Log Used')

and instance\_name != '\_Total'

and db\_id(instance\_name) is not null

group by instance\_name

order by instance\_name

set @sbody = ''

--declare rec CURSOR for

Select @sbody = @sbody + '<tr>

<td><font face="arial" size="2">' + rtrim(ltrim(instance\_name)) + '</font></td>

<td><font face="arial" size="2">' + ltrim(str([DataFileSize (KB)])) + '</font></td>

<td><font face="arial" size="2">' +

case when [LogFileSize (KB)] < ([DataFileSize (KB)] \* 0.1) THEN '<font color="red"><b>' ELSE '' END

+ ltrim(str([LogFileSize (KB)])) +

case when [LogFileSize (KB)] < ([DataFileSize (KB)] \* 0.1) THEN '</b></font>' ELSE '' END

+ '</font></td>

<td><font face="arial" size="2">' + ltrim(str([PercentLogUsed])) + '</font></td>

</tr>'

from #size

where [LogFileSize (KB)] < ([DataFileSize (KB)] \* 0.1)

set @rcnt = @@rowcount

print @sbody

--open rec

--FETCH NEXT FROM rec into @cbody

--WHILE (@@FETCH\_STATUS = 0)

--begin

-- print @cbody

-- FETCH NEXT FROM rec into @cbody

--end

--close rec

--deallocate rec

print '</table> </font>'

-- Database Sizing end

set @sbody = ''

declare @i int

set @i = 1

while @i <= @rcnt

begin

print '<br>'

set @i = @i + 1

end

print '<br></br><br></br>'

-- Database Growth Pattern

print '<h2> Database Growth Pattern </h2>'

-- pending below lines

print '• The auto-growth parameters for user database data and log files will be set to expand using a static size (MB) rather than by percentages.'

print '<br></br>'

print '• The static growth size will be set to 10% of the original file size rounded up to the nearest 10 MB for databases smaller than 1 GB, nearest 100 MB for databases of multiple GBs and nearest 1000 MB for databases of 1 TB or larger. '

print '<br></br>'

print '• There will not be file growth limits for any database file.'

print '<br></br>'

set @sbody = @sbody + '<table border=1 align="left" cellpadding="1" cellspacing="0" >'

set @sbody = @sbody + '<tr>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Database Name</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Name</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">File Name</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Size (MB)</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Growth (MB)</font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Growth Pattern</font></TH>

</tr>'

print @sbody

if object\_id('tempdb..#growth') is not null

drop table #growth

select db\_name(dbid) DBName, name, filename, ((size \* 8) / 1024) SizeInMB,

case when status & 0x100000 = 0 then ((growth \* 8) / 1024)

else ((size \* 8 \* 10) / (1024 \* 100)) end GrowthInMB,

case when status & 0x100000 = 0 then 'MB' else 'Percentage' end IsPercentage

into #growth

from sysaltfiles

set @sbody = ''

declare rec CURSOR for

--Select @sbody = @sbody + '<tr>

Select '<tr>

<td><font face="arial" size="2">' + rtrim(ltrim(DBName)) + '</font></td>

<td><font face="arial" size="2">' + rtrim(ltrim([Name])) + '</font></td>

<td><font face="arial" size="2">' + rtrim(ltrim(FileName)) + '</font></td>

<td><font face="arial" size="2">' + ltrim(str(SizeInMB)) + '</font></td>

<td><font face="arial" size="2">' +

case when SizeInMB <= 1024 then

case when ABS((SizeInMB \* 0.1) - GrowthInMB) <= 10 then ''

else '<font color="red"><b>' end

when SizeInMB > 1024 and SizeInMB < (1024 \* 1024) then

case when GrowthInMB = 100 then ''

else '<font color="red"><b>' end

when SizeInMB > (1024 \* 1024) then

case when GrowthInMB = 1000 then ''

else '<font color="red"><b>' end

end +

ltrim(str(GrowthInMB)) +

case when SizeInMB <= 1024 then

case when ABS((SizeInMB \* 0.1) - GrowthInMB) <= 10 then ''

else '</b></font>' end

when SizeInMB > 1024 and SizeInMB < (1024 \* 1024) then

case when GrowthInMB = 100 then ''

else '</b></font>' end

when SizeInMB > (1024 \* 1024) then

case when GrowthInMB = 1000 then ''

else '</b></font>' end

end +

'<td><font face="arial" size="2">' +

case when IsPercentage = 'Percentage' THEN '<font color="red"><b>' ELSE '' END

+ ltrim(IsPercentage) +

case when IsPercentage = 'Percentage' THEN '</b></font>' ELSE '' END

+ '</font></td>

</tr>'

from #growth

where

((SizeInMB <= 1024 and ABS((SizeInMB \* 0.1) - GrowthInMB) > 10) or

((SizeInMB > 1024 and SizeInMB < (1024 \* 1024)) and GrowthInMB <> 100) or

(SizeInMB > (1024 \* 1024) and GrowthInMB <> 1000)

) OR

IsPercentage = 'Percentage'

open rec

FETCH NEXT FROM rec into @cbody

WHILE (@@FETCH\_STATUS = 0)

begin

print @cbody

FETCH NEXT FROM rec into @cbody

end

close rec

deallocate rec

--print @sbody

print '</table> </font>'

print '</body></html>'

## Unindexed Foreign Keys

if object\_id('tempdb.dbo.#UnindexedFKs') is not null

drop table #UnindexedFKs

create table #UnindexedFKs

(dbName varchar(50), PK\_TableName varchar(50), FK\_TableName varchar(50), FK\_Name varchar(200),

ColumnName varchar(50), NumberOfRows BIGINT)

exec sp\_msforeachdb '

use [?] SET QUOTED\_IDENTIFIER ON

INSERT #UnindexedFKs (dbName, PK\_TableName, FK\_TableName, FK\_Name, ColumnName, NumberOfRows)

SELECT DISTINCT

DB\_NAME() AS "database\_name",

OBJECT\_NAME(foreign\_keys.referenced\_object\_id) AS "pk\_table\_name",

OBJECT\_NAME(foreign\_keys.parent\_object\_id) AS "fk\_table\_name",

foreign\_keys."name" AS "fk\_name",

syscol.name,

rowcnt

FROM sys.foreign\_keys AS foreign\_keys

JOIN sys.foreign\_key\_columns AS foreign\_key\_columns

ON foreign\_keys."object\_id" = foreign\_key\_columns.constraint\_object\_id

JOIN sys.columns as syscol ON foreign\_keys.parent\_object\_id = syscol."object\_id" AND

foreign\_key\_columns.parent\_column\_id = syscol.column\_id

JOIN sysindexes ON foreign\_keys.parent\_object\_id = sysindexes.id

WHERE NOT EXISTS (

SELECT \*

FROM sys.indexes AS indexes

JOIN sys.index\_columns AS index\_columns

ON indexes."object\_id" = index\_columns."object\_id"

WHERE foreign\_keys.parent\_object\_id = indexes."object\_id"

AND indexes.index\_id = index\_columns.index\_id

AND foreign\_key\_columns.constraint\_column\_id = index\_columns.key\_ordinal

AND foreign\_key\_columns.parent\_column\_id = index\_columns.column\_id

AND OBJECTPROPERTYEX(indexes."object\_id",''IsMSShipped'') = 0

AND indexes.is\_hypothetical = 0

)

AND foreign\_keys.is\_ms\_shipped = 0

and indid in (0,1) and rowcnt > 10000

and DB\_NAME() not in (''master'', ''msdb'', ''model'', ''tempdb'')

order by fk\_name, syscol.name'

select \* from #UnindexedFKs

## Non Primary Key Tables

if object\_id('tempdb.dbo.#t') is not null

drop table #t

create table #t (DBName varchar(50), TableName varchar(50))

exec master..sp\_MSforeachdb '

use [?]

if db\_name() not in (''master'', ''model'', ''msdb'', ''tempdb'')

insert #t

SELECT db\_name(), p.Name

FROM sysobjects p

LEFT JOIN (SELECT c.Id

FROM sysobjects a

JOIN sysconstraints b ON a.Id = b.ConstId

JOIN sysobjects c ON b.Id = c.Id

WHERE a.xtype = ''PK'') q ON q.Id = p.Id

WHERE p.xtype = ''U'' and p.category = 0 AND q.Id IS NULL

ORDER BY p.Name'

select \* from #t

order by 1

## Blocking Script

BEGIN

CREATE TABLE #DBCC(

PARENTOBJECT NVARCHAR(128),

OBJECT NVARCHAR(128),

FIELD NVARCHAR(128),

VALUE NVARCHAR(128)

)

DECLARE @BLOCKED TABLE(

BLOCKER\_SPID SMALLINT,

BLOCKER\_CONTEXT VARCHAR(128),

BLOCKER\_STATUS VARCHAR(18),

BLOCKED\_SPID SMALLINT,

BLOCKED\_CONTEXT VARCHAR(128),

WAITTIME INT,

LOCK\_MODE VARCHAR(7),

LOCK\_TYPE CHAR(3),

DBID SMALLINT,

LOCK\_RESOURCE VARCHAR(30),

BLOCKER\_SQL TEXT,

BLOCKED\_SQL TEXT,

HOSTNAME\_D VARCHAR(100),

APPLICATION\_NAME\_D VARCHAR(500),

LOGIN\_NAME\_D VARCHAR(100),

CPUTIME\_D INT,

DISKIO\_D INT,

HOSTNAME\_R VARCHAR(100),

APPLICATION\_NAME\_R VARCHAR(500),

LOGIN\_NAME\_R VARCHAR(100),

CPUTIME\_R INT,

DISKIO\_R INT

)

DECLARE @BLOCKED2 TABLE(

BLOCKER\_SPID SMALLINT,

BLOCKER\_CONTEXT VARCHAR(128),

BLOCKER\_STATUS VARCHAR(18),

BLOCKED\_SPID SMALLINT,

BLOCKED\_CONTEXT VARCHAR(128),

WAITTIME INT,

LOCK\_MODE VARCHAR(7),

LOCK\_TYPE CHAR(3),

DBNAME CHAR(8),

TABLENAME CHAR(18),

INDEXID INT,

BLOCKER\_SQL TEXT,

BLOCKED\_SQL TEXT,

HOSTNAME\_D VARCHAR(100),

APPLICATION\_NAME\_D VARCHAR(500),

LOGIN\_NAME\_D VARCHAR(100),

CPUTIME\_D INT,

DISKIO\_D INT,

HOSTNAME\_R VARCHAR(100),

APPLICATION\_NAME\_R VARCHAR(500),

LOGIN\_NAME\_R VARCHAR(100),

CPUTIME\_R INT,

DISKIO\_R INT

)

CREATE TABLE #SQLTEXT(

EVENT\_TYPE VARCHAR(100),

PARAMETERS VARCHAR(100),

EVENT\_INFO TEXT

)

SET NOCOUNT ON

DECLARE

@BLOCKER\_SPID SMALLINT,

@BLOCKER\_CONTEXT VARCHAR(128),

@BLOCKER\_STATUS VARCHAR(18),

@BLOCKED\_SPID SMALLINT,

@BLOCKED\_CONTEXT VARCHAR(128),

@WAITTIME INT,

@LOCK\_MODE VARCHAR(7),

@LOCK\_TYPE CHAR(3),

@DBID SMALLINT,

@OBJECTID INT,

@INDEXID INT,

@LOCK\_RESOURCE VARCHAR(30),

@BLOCKER\_HANDLE BINARY(20),

@BLOCKER\_SQL VARCHAR(8000),

@BLOCKED\_HANDLE BINARY(20),

@BLOCKED\_SQL VARCHAR(8000),

@CMD VARCHAR(1000),

@DELIMITER1 TINYINT,

@DELIMITER2 TINYINT,

@DELIMITER3 TINYINT,

@FILEID VARCHAR(10),

@PAGEID VARCHAR(10),

@HOSTNAME\_D VARCHAR(100),

@APPLICATION\_NAME\_D VARCHAR(500),

@LOGIN\_NAME\_D VARCHAR(100),

@CPUTIME\_D INT,

@DISKIO\_D INT,

@HOSTNAME\_R VARCHAR(100),

@APPLICATION\_NAME\_R VARCHAR(500),

@LOGIN\_NAME\_R VARCHAR(100),

@CPUTIME\_R INT,

@DISKIO\_R INT

DECLARE PROCESSES CURSOR FOR

SELECT \* FROM

( SELECT

BLOCKER.spid BLOCKER\_SPID,

RTRIM(convert(varchar(128),BLOCKER.context\_info)) BLOCKER\_CONTEXT,

CASE BLOCKER.blocked

WHEN 0 THEN 'Lead Blocker'

ELSE 'In Blocking Chain'

END BLOCKER\_STATUS,

BLOCKED.spid BLOCKED\_SPID,

RTRIM(convert(varchar(128),BLOCKED.context\_info)) BLOCKED\_CONTEXT,

BLOCKED.waittime,

CASE CONVERT(TINYINT, BLOCKED.waittype)

WHEN 1 THEN 'SCH-ST'

WHEN 2 THEN 'SCH-MOD'

WHEN 3 THEN 'S'

WHEN 4 THEN 'U'

WHEN 5 THEN 'X'

WHEN 6 THEN 'IS'

WHEN 7 THEN 'IU'

WHEN 8 THEN 'IX'

WHEN 9 THEN 'SIU'

WHEN 10 THEN 'SIX'

WHEN 11 THEN 'UIX'

WHEN 12 THEN 'BU'

WHEN 13 THEN 'RangeS-S'

WHEN 14 THEN 'RangeS-U'

WHEN 15 THEN 'RangeIn-Null'

WHEN 16 THEN 'RangeIn-S'

WHEN 17 THEN 'RangeIn-U'

WHEN 18 THEN 'RangeIn-X'

WHEN 19 THEN 'RangeX-S'

WHEN 20 THEN 'RangeX-U'

WHEN 21 THEN 'RangeX-X'

ELSE 'UNKNOWN'

END LOCK\_MODE,

SUBSTRING(BLOCKED.waitresource,1,3) LOCK\_RESOURCE\_TYPE,

BLOCKED.dbid DBID,

SUBSTRING(BLOCKED.waitresource,6,30) LOCK\_RESOURCE,

BLOCKER.sql\_handle sql\_handle1,

BLOCKER.cmd cmd1,

BLOCKED.sql\_handle,

BLOCKED.cmd,

BLOCKER.hostname hostname1,

BLOCKER.program\_name program\_name1,

BLOCKER.loginame loginame1,

BLOCKER.cpu cpu1,

BLOCKER.physical\_io physical\_io1,

BLOCKED.hostname,

BLOCKED.program\_name,

BLOCKED.loginame,

BLOCKED.cpu,

BLOCKED.physical\_io

FROM master..sysprocesses BLOCKER

JOIN master..sysprocesses BLOCKED ON BLOCKER.spid = BLOCKED.blocked

WHERE BLOCKED.blocked <> 0

AND DATEDIFF(mi, BLOCKED.last\_batch,getdate()) > 1 -- show blocking more than 1 minute

) as A

WHERE A.BLOCKER\_SPID <> A.BLOCKED\_SPID

OPEN PROCESSES

FETCH PROCESSES

INTO @BLOCKER\_SPID ,

@BLOCKER\_CONTEXT ,

@BLOCKER\_STATUS ,

@BLOCKED\_SPID ,

@BLOCKED\_CONTEXT ,

@WAITTIME ,

@LOCK\_MODE ,

@LOCK\_TYPE ,

@DBID ,

@LOCK\_RESOURCE ,

@BLOCKER\_HANDLE ,

@BLOCKER\_SQL ,

@BLOCKED\_HANDLE ,

@BLOCKED\_SQL ,

@HOSTNAME\_D ,

@APPLICATION\_NAME\_D ,

@LOGIN\_NAME\_D ,

@CPUTIME\_D ,

@DISKIO\_D ,

@HOSTNAME\_R ,

@APPLICATION\_NAME\_R ,

@LOGIN\_NAME\_R ,

@CPUTIME\_R ,

@DISKIO\_R

WHILE @@FETCH\_STATUS = 0

BEGIN

INSERT #SQLTEXT

EXEC ('dbcc inputbuffer('+@BLOCKED\_SPID+')')

SELECT @BLOCKED\_SQL = EVENT\_INFO FROM #SQLTEXT

TRUNCATE TABLE #SQLTEXT

INSERT #SQLTEXT

EXEC ('dbcc inputbuffer('+@BLOCKER\_SPID+')')

SELECT @BLOCKER\_SQL = EVENT\_INFO FROM #SQLTEXT

INSERT INTO @BLOCKED VALUES(

@BLOCKER\_SPID ,

@BLOCKER\_CONTEXT ,

@BLOCKER\_STATUS ,

@BLOCKED\_SPID ,

@BLOCKED\_CONTEXT ,

@WAITTIME ,

@LOCK\_MODE ,

@LOCK\_TYPE ,

@DBID ,

@LOCK\_RESOURCE ,

@BLOCKER\_SQL ,

@BLOCKED\_SQL ,

@HOSTNAME\_D ,

@APPLICATION\_NAME\_D ,

@LOGIN\_NAME\_D ,

@CPUTIME\_D ,

@DISKIO\_D ,

@HOSTNAME\_R ,

@APPLICATION\_NAME\_R ,

@LOGIN\_NAME\_R ,

@CPUTIME\_R ,

@DISKIO\_R )

FETCH PROCESSES INTO

@BLOCKER\_SPID ,

@BLOCKER\_CONTEXT ,

@BLOCKER\_STATUS ,

@BLOCKED\_SPID ,

@BLOCKED\_CONTEXT ,

@WAITTIME ,

@LOCK\_MODE ,

@LOCK\_TYPE ,

@DBID ,

@LOCK\_RESOURCE ,

@BLOCKER\_HANDLE ,

@BLOCKER\_SQL ,

@BLOCKED\_HANDLE ,

@BLOCKED\_SQL ,

@HOSTNAME\_D ,

@APPLICATION\_NAME\_D ,

@LOGIN\_NAME\_D ,

@CPUTIME\_D ,

@DISKIO\_D ,

@HOSTNAME\_R ,

@APPLICATION\_NAME\_R ,

@LOGIN\_NAME\_R ,

@CPUTIME\_R ,

@DISKIO\_R

END -- @@FETCH\_STATUS = 0

DEALLOCATE PROCESSES

DROP TABLE #SQLTEXT

DECLARE BLOCKED CURSOR FOR

SELECT

BLOCKER\_SPID ,

BLOCKER\_CONTEXT ,

BLOCKER\_STATUS ,

BLOCKED\_SPID ,

BLOCKED\_CONTEXT ,

WAITTIME ,

LOCK\_MODE ,

LOCK\_TYPE ,

DBID ,

LOCK\_RESOURCE ,

BLOCKER\_SQL ,

BLOCKED\_SQL ,

@HOSTNAME\_D ,

APPLICATION\_NAME\_D ,

LOGIN\_NAME\_D ,

CPUTIME\_D ,

DISKIO\_D ,

HOSTNAME\_R ,

APPLICATION\_NAME\_R ,

LOGIN\_NAME\_R ,

CPUTIME\_R ,

DISKIO\_R

FROM @BLOCKED

OPEN BLOCKED

FETCH BLOCKED

INTO @BLOCKER\_SPID ,

@BLOCKER\_CONTEXT ,

@BLOCKER\_STATUS ,

@BLOCKED\_SPID ,

@BLOCKED\_CONTEXT ,

@WAITTIME ,

@LOCK\_MODE ,

@LOCK\_TYPE ,

@DBID ,

@LOCK\_RESOURCE ,

@BLOCKER\_SQL ,

@BLOCKED\_SQL ,

@HOSTNAME\_D ,

@APPLICATION\_NAME\_D ,

@LOGIN\_NAME\_D ,

@CPUTIME\_D ,

@DISKIO\_D ,

@HOSTNAME\_R ,

@APPLICATION\_NAME\_R ,

@LOGIN\_NAME\_R ,

@CPUTIME\_R ,

@DISKIO\_R

WHILE @@FETCH\_STATUS = 0

BEGIN

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

-- Decode the waitresource column from sysprocesses.

-- The 1st 5 bytes have already been trimmed off and stored in LOCK\_TYPE

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

-- Establish position of the delimiters between the fields of the lock

-- resource. In order to establish a uniform delimiter to look for, replace ':' with ' '

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

SET @LOCK\_RESOURCE = REPLACE(@LOCK\_RESOURCE,':',' ')

SET @DELIMITER1 = CHARINDEX(' ',@LOCK\_RESOURCE)

SET @DELIMITER2 = CHARINDEX(' ', @LOCK\_RESOURCE, (@DELIMITER1+1))

SET @DELIMITER3 = CHARINDEX(' ', @LOCK\_RESOURCE, (@DELIMITER2+1))

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

-- Delimiter positions are then used to substring fields from @LOCK\_RESOURCE

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

IF @LOCK\_TYPE IN ('RID','PAG')

BEGIN

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

-- Extract objectid and indexid from file/page for resources of RID or PAG.

-- LOCK\_RESOURCE\_TYPE 'RID': dbid:fileid:pageid:row#

-- Example: 7:3:47912:10

-- LOCK\_RESOURCE\_TYPE 'PAG': dbid:fileid:pageid

-- Example: 7:3:47912

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

SET @FILEID = SUBSTRING(@LOCK\_RESOURCE,@DELIMITER1+1,(@DELIMITER2-@DELIMITER1)-1)

SET @PAGEID = SUBSTRING(@LOCK\_RESOURCE,@DELIMITER2+1,(@DELIMITER3-@DELIMITER2)-1)

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

-- Execute DBCC PAGE to determine the object owner of the page. We use the rowset

-- version of DBCC (WITH TABLERESULTS) to make it easy to retrieve objext id and index id.

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

SET @CMD = 'DBCC PAGE ('+CONVERT(VARCHAR(3),@DBID)+','+@FILEID+','+@PAGEID+') WITH TABLERESULTS, no\_infomsgs'

INSERT INTO #DBCC EXEC(@CMD)

SELECT @OBJECTID = CONVERT(INT,SUBSTRING(VALUE,1,30)) FROM #DBCC WHERE FIELD = 'm\_objId' OPTION(KEEP PLAN)

SELECT @INDEXID = CONVERT(INT,SUBSTRING(VALUE,1,30)) FROM #DBCC WHERE FIELD = 'm\_indexId' OPTION(KEEP PLAN)

TRUNCATE TABLE #DBCC

END

IF @LOCK\_TYPE = 'TAB'

BEGIN

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

-- LOCK\_RESOURCE\_TYPE 'TAB': dbid:objectid

-- Example: 7:1993058136

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

SET @OBJECTID = SUBSTRING(@LOCK\_RESOURCE,@DELIMITER1+1,(@DELIMITER2-@DELIMITER1)-1)

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

-- For table locks, set indexid to '0'

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

SET @INDEXID = 0

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

-- If the waitresource contains the keyword COMPILE, then the object name is actually a

-- stored procedure.

-- Example: 6:834102 [[COMPILE]]

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

IF @LOCK\_RESOURCE LIKE '%COMPILE%'

SET @LOCK\_TYPE = 'PRC'

END

IF @LOCK\_TYPE = 'KEY'

BEGIN

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

-- LOCK\_RESOURCE\_TYPE 'KEY': dbid:objectid:indexid (hash of key value)

-- Example: 7:1993058136:4 (0a0087c006b1)

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

SET @OBJECTID = SUBSTRING(@LOCK\_RESOURCE,@DELIMITER1+1,(@DELIMITER2-@DELIMITER1)-1)

SET @INDEXID = SUBSTRING(@LOCK\_RESOURCE,@DELIMITER2+1,(@DELIMITER3-@DELIMITER2)-1)

END

INSERT INTO @BLOCKED2 VALUES (

@BLOCKER\_SPID ,

@BLOCKER\_CONTEXT ,

@BLOCKER\_STATUS ,

@BLOCKED\_SPID ,

@BLOCKED\_CONTEXT ,

@WAITTIME ,

@LOCK\_MODE ,

@LOCK\_TYPE ,

LEFT(DB\_NAME(@DBID),8) ,

LEFT(OBJECT\_NAME(@OBJECTID),18) ,

@INDEXID ,

@BLOCKER\_SQL ,

@BLOCKED\_SQL ,

@HOSTNAME\_D ,

@APPLICATION\_NAME\_D ,

@LOGIN\_NAME\_D ,

@CPUTIME\_D ,

@DISKIO\_D ,

@HOSTNAME\_R ,

@APPLICATION\_NAME\_R ,

@LOGIN\_NAME\_R ,

@CPUTIME\_R ,

@DISKIO\_R

)

FETCH BLOCKED

INTO @BLOCKER\_SPID ,

@BLOCKER\_CONTEXT ,

@BLOCKER\_STATUS ,

@BLOCKED\_SPID ,

@BLOCKED\_CONTEXT ,

@WAITTIME ,

@LOCK\_MODE ,

@LOCK\_TYPE ,

@DBID ,

@LOCK\_RESOURCE ,

@BLOCKER\_SQL ,

@BLOCKED\_SQL ,

@HOSTNAME\_D ,

@APPLICATION\_NAME\_D ,

@LOGIN\_NAME\_D ,

@CPUTIME\_D ,

@DISKIO\_D ,

@HOSTNAME\_R ,

@APPLICATION\_NAME\_R ,

@LOGIN\_NAME\_R ,

@CPUTIME\_R ,

@DISKIO\_R

END

DEALLOCATE BLOCKED

SELECT

BLOCKED\_SPID ,

BLOCKER\_SPID ,

DBNAME ,

-- TABLENAME ,

-- INDEXID ,

-- LOCK\_TYPE ,

-- LOCK\_MODE ,

BLOCKER\_SQL ,

BLOCKED\_SQL ,

BLOCKED\_HOSTNAME = HOSTNAME\_D ,

BLOCKED\_APPLICATION\_NAME = APPLICATION\_NAME\_D ,

BLOCKED\_LOGIN\_NAME = LOGIN\_NAME\_D ,

BLOCKED\_CPUTIME = CPUTIME\_D ,

BLOCKED\_DISKIO = DISKIO\_D ,

BLOCKER\_HOSTNAME = HOSTNAME\_R ,

BLOCKER\_APPLICATION\_NAME = APPLICATION\_NAME\_R ,

BLOCKER\_LOGIN\_NAME = LOGIN\_NAME\_R ,

BLOCKER\_CPUTIME = CPUTIME\_R ,

BLOCKER\_DISKIO = DISKIO\_R ,

Captured\_Time = GETDATE()

FROM @BLOCKED2 ORDER BY BLOCKER\_STATUS desc

drop table #DBCC

END

## Long Running processes

SET NOCOUNT ON

DECLARE @Threshold INT

SET @Threshold = 5

declare @SPID INT, @CNT INT, @TotalCnt INT

declare @SQL\_Text VARCHAR(4000)

SET @CNT = 1

DECLARE @Process TABLE

(

IDT INT IDENTITY(1,1) ,

Long\_Process\_Threshold INT ,

Current\_Run\_Time INT ,

SPID INT ,

[Program\_Name] VARCHAR(200) ,

Login\_Time DATETIME ,

Last\_Batch DATETIME ,

SQL\_Statements VARCHAR(4000)

)

CREATE TABLE #INPUT

(

EventType VARCHAR(100) ,

Parameters INT ,

SQL\_Text VARCHAR(4000)

)

INSERT INTO @Process

SELECT @Threshold [Long Process Threshold]

, DATEDIFF(mi, last\_batch, getdate()) [Current Run Time]

, SPID [Session ID]

, Program\_Name [Program]

, Login\_TIme

, Last\_Batch [Last Batch]

, '' [SQL]

FROM master.dbo.sysprocesses s

--WHERE STATUS = 'RUNNABLE'

--AND

WHERE DATEDIFF(mi, last\_batch, getdate()) > @Threshold

SELECT @TotalCnt = (SELECT count(\*) from @Process)

WHILE (@CNT <= @TotalCnt)

BEGIN

set @SPID = (select SPID from @Process where IDT = @CNT)

SET @SQL\_Text = 'DBCC INPUTBUFFER(' + cast(@SPID as varchar)+ ')'

Insert Into #INPUT

EXEC (@SQL\_Text)

Update @Process

SET SQL\_Statements = (Select SQL\_Text from #INPUT)

Where IDT = @CNT

Truncate table #INPUT

SET @CNT = @CNT + 1

END

Select Long\_Process\_Threshold, Current\_Run\_Time, SPID, [Program\_Name], Login\_Time, Last\_Batch, SQL\_Statements from @Process

DROP TABLE #INPUT

# Trace Scripts

## Script to get bad query with no. of frequency from trace file

select top 10 cast(substring(textdata,1,50)as varchar(100)),

count(cast(substring(textdata,1,50)as varchar(100))) freq,

LoginName,

Sum(Duration) Duration,

Sum(Reads) Reads,

Sum(writes) Writes,

Sum(CPU) CPU, ServerName, ApplicationName

from ::fn\_trace\_gettable('D:\DVTrace\dv\_sql.trc',default)

group by cast(substring(textdata,1,50)as varchar(100)), LoginName, ServerName, ApplicationName

order by sum(duration) desc, sum(reads) desc, Sum(writes) desc, sum(cpu) desc

## Query to Load worse offending queries from trace file

--This Query assumes you called your trace table dv\_trace

select convert(varchar(25),textdata) Query, avg(reads) AvgReads, avg(duration) AvgDuration, avg(cpu) AvgCPU, count(\*) QueryFrequency, avg(reads) \* count(\*) TotalImpactIO, avg(duration) \* count(\*) TotalImpactDuration, avg(cpu) \* count(\*) TotalImpactCPU

from [dv\_trace]

group by convert(varchar(25),textdata)

having count(\*) > 1

order by 4 desc

--This will strip the first 20 characters off of the query results for 'text data' for statements that begin with 'declare...', etc

select substring(convert(varchar(46),textdata),21,46) Query, avg(reads) AvgReads, avg(duration) AvgDuration, avg(cpu) AvgCPU, count(\*) QueryFrequency, avg(reads) \* count(\*) TotalImpactIO, avg(duration) \* count(\*) TotalImpactDuration, avg(cpu) \* count(\*) TotalImpactCPU

from [dv\_trace]

group by substring(convert(varchar(46),textdata),21,46)

having count(\*) > 1

order by 4 desc

## Get TOP SQL from Trace File

SELECT TextData, Duration, Reads, Writes, CPU,

DatabaseId, DatabaseName, ApplicationName, LoginName,

SPID, StartTime, EndTime, EventClass

INTO Dcom\_TraceData

FROM ::fn\_trace\_gettable('J:\Dv\_Trace\Dcom\_TrcData.trc', default)

select convert(varchar(60),textdata) Query, avg(reads) AvgReads,

avg(duration) AvgDuration, avg(cpu) AvgCPU, count(\*) QueryFrequency,

avg(reads) \* count(\*) TotalImpactIO,

avg(duration) \* count(\*) TotalImpactDuration,

avg(cpu) \* count(\*) TotalImpactCPU

into #t

from Dcom\_TraceData

where duration > 1000

--and convert(varchar(25),textdata) like 'DECLARE%'

group by convert(varchar(60),textdata)

having count(\*) > 1

order by 6 desc

select top 20 (select top 1 TextData

from Dcom\_TraceData

where convert(varchar(60), textdata) = a.Query) Query

,a.AvgReads, AvgDuration, AvgCPU, QueryFrequency,

TotalImpactIO, TotalImpactDuration, TotalImpactCPU

from #t a

order by TotalImpactIO desc

-----------

select convert(varchar(60),textdata) Query, avg(reads) AvgReads, avg(duration) AvgDuration, avg(cpu) AvgCPU, count(\*) QueryFrequency, avg(reads) \* count(\*) TotalImpactIO, avg(duration) \* count(\*) TotalImpactDuration, avg(cpu) \* count(\*) TotalImpactCPU

from [20080617\_trace]

where duration > 1000

--and convert(varchar(25),textdata) like 'DECLARE%'

group by convert(varchar(60),textdata)

having count(\*) > 1

order by 6 desc

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

DROP TABLE #t1

SELECT CAST(TextData AS VARCHAR(20)) TextData, MAX(RowNumber) RowNumber

INTO #t1

FROM --BadSql01

WHERE (ISNULL(Duration, 0) > 1000 OR ISNULL(Reads, 0) > 1000 OR ISNULL(Writes, 0) > 1000 OR ISNULL(CPU, 0) > 1000) AND

TextData NOT LIKE '%sqlagent%' AND TextData NOT LIKE '%DBCC%' AND TextData NOT LIKE '%BackUp%' AND

TextData NOT LIKE '%Restore%' AND TextData NOT LIKE '%RebuildIndex%' AND TextData NOT LIKE '%Defrag%' AND

TextData NOT LIKE '%Populat%' AND TextData NOT LIKE '%Transfer%' AND TextData NOT LIKE '%WeeklyJobs%'

GROUP BY CAST(TextData AS VARCHAR(20))

SELECT top 20 b.TextData, b.Duration, Reads, Writes, CPU,

FROM #t1 a

JOIN BadSql01 b ON a.RowNumber = b.RowNumber

ORDER BY Duration DESC

SELECT top 20 b.TextData, b.Duration, Reads, Writes, CPU

FROM #t1 a

JOIN BadSql01 b ON a.RowNumber = b.RowNumber

ORDER BY Reads DESC

SELECT top 20 b.TextData, b.Duration, Reads, Writes, CPU

FROM #t1 a

JOIN BadSql01 b ON a.RowNumber = b.RowNumber

ORDER BY Writes DESC

SELECT top 20 b.TextData, b.Duration, Reads, Writes, CPU

FROM #t1 a

JOIN BadSql01 b ON a.RowNumber = b.RowNumber

ORDER BY CPU DESC

select \* from #t1

## Deadlock Trace Analysis

Here are the initial steps for getting relevant data to start deadlock analysis.

After the collecting the trace data, load the **textdata** to table **dv\_deadlock**; below is the query;

Select IDENTITY(INT, 1, 1) Row\_Number, textdata into dv\_deadlock

from ::fn\_trace\_gettable('trace\_file\_path',default) where textdata like '%deadlock-list%' and starttime > ‘what-ever’

Then run the below query, it will create a table **DV\_All\_Deadlock\_Analysis** and use a cursor to loop through the XML data in **dv\_deadlock** (The deadlock graph stores data in XML format in textdata colum) and populate the table with required information. Then using this information we need to further analyse to minimize/eliminate/find root cause of deadlocks

if (select object\_id('master..DV\_All\_Deadlock\_Analysis'))is not null

      drop table [DV\_All\_Deadlock\_Analysis]

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[DV\_All\_Deadlock\_Analysis](

      [Counter] [int] NULL,

      [ProcessStatus] [varchar](6) COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS NOT NULL,

      [id] [varchar](25) COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS NULL,

      [Victim] [varchar](25) COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS NULL,

      [inputbuf] [varchar](max) COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS NULL,

      [ProcName] [varchar](100) COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS NULL,

      [transactionname] [varchar](20) COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS NULL,

      [taskpriority] [int] NULL,

      [logused] [int] NULL,

      [waitresource] [varchar](50) COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS NULL,

      [waittime] [int] NULL,

      [status] [varchar](20) COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS NULL,

      [spid] [int] NULL,

      [priority] [int] NULL,

      [trancount] [int] NULL,

      [lastbatchstarted] [varchar](30) COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS NULL,

      [clientapp] [varchar](50) COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS NULL,

      [hostname] [varchar](50) COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS NULL,

      [isolationlevel] [varchar](20) COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS NULL,

      [currentdb] [int] NULL,

      [lockTimeout] [bigint] NULL

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

DECLARE @xmlDoc VARCHAR(max)

DECLARE @handle INT, @cnt int, @rowNo int

set @cnt = 1

declare cur\_deadlock cursor

for select Row\_Number, textdata from dv\_deadlock –-where Row\_Number not in (3, 6)

OPEN cur\_deadlock

FETCH NEXT FROM cur\_deadlock INTO @rowNo ,@xmlDoc

WHILE @@FETCH\_STATUS = 0

BEGIN

      EXEC sp\_xml\_preparedocument @handle OUTPUT, @xmlDoc

      INSERT INTO DV\_All\_Deadlock\_Analysis

      SELECT @cnt Counter,CASE WHEN id=victim THEN 'Victim' ELSE 'Winner' END ProcessStatus, \*

      FROM OPENXML (@handle, '/deadlock-list/deadlock/process-list/process', 2)

      WITH

      (     id varchar(25)    '@id',

            Victim varchar(25)      '../../@victim',

            inputbuf VARCHAR(max),

            ProcName varchar(100)   'executionStack/frame/@procname',

            transactionname varchar(20) '@transactionname',

            taskpriority int  '@taskpriority',

            logused int '@logused',

            waitresource varchar(50)      '@waitresource',

            waittime int      '@waittime',

            status varchar(20)      '@status',

            spid int    '@spid',

            priority int      '@priority',

            trancount int     '@trancount',

            lastbatchstarted varchar(30)  '@lastbatchstarted',

            clientapp varchar(50)   '@clientapp',

            hostname varchar(50)    '@hostname',

            isolationlevel varchar(20)    '@isolationlevel',

            currentdb int     '@currentdb',

            lockTimeout bigint      '@lockTimeout')

      EXEC sp\_xml\_removedocument @handle

set @cnt = @cnt + 1

FETCH NEXT FROM cur\_deadlock INTO @rowNo ,@xmlDoc

END

CLOSE cur\_deadlock

DEALLOCATE cur\_deadlock

SELECT \* FROM DV\_All\_Deadlock\_Analysis

**Note: Some time the XML data would be not in proper format which may result in failure (you can skip them by putting the Row\_number in where condition of cursor). This improperly formatted data needs to be manually looked into.**

# Generic Script-out Object Query

* 1. Script-out Sql Server Objects using SQLDMO.SQLServer

/\*

exec s\_ScriptObjects

@SourceDB = 'AdventureWorks' ,

@SourceObject = null , -- null for all objects

@SourceUID = 'sa' ,

@SourcePWD = 'sa123' ,

@OutFilePath = 'C:\temp\' ,

@OutFileName = 'StoredProcedures.sql' , -- null for separate file per object script

@ObjectType = 'JOBS' ,

@WorkPath = 'C:\temp' ,

@SourceSVR = 'RAM'

\*/

if exists (select \* from dbo.sysobjects where id = object\_id(N'[dbo].[s\_ScriptAllDatabases]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)

drop procedure [dbo].[s\_ScriptAllDatabases]

GO

CREATE procedure s\_ScriptAllDatabases

@SourceUID varchar(128) , -- null for trusted connection

@SourcePWD varchar(128) ,

@OutFilePath varchar(256) , -- Root path - will add directory for object types

@OutFileName varchar(128) , -- null for separate file per object script

@WorkPath varchar(256) ,

@SourceSVR varchar(128) = null , -- to script remote server

@Database varchar(128) = null -- to script single database / jobs / DTS

as

/\*

exec s\_ScriptAllDatabases

@SourceUID = null ,

@SourcePWD = null ,

@OutFilePath = 'c:\a\' ,

@OutFileName = null ,

@WorkPath = 'c:\temp\' , -- no spaces

@SourceSVR = null

exec s\_ScriptAllDatabases

@SourceUID = null ,

@SourcePWD = null ,

@OutFilePath = 'c:\a\' ,

@OutFileName = null ,

@WorkPath = 'c:\temp\' , -- no spaces

@SourceSVR = null ,

@Database = 'JOBS'

exec s\_ScriptAllDatabases

@SourceUID = null ,

@SourcePWD = null ,

@OutFilePath = 'c:\a\' ,

@OutFileName = null ,

@WorkPath = 'c:\temp\' , -- no spaces

@SourceSVR = null ,

@Database = 'DTS'

exec s\_ScriptAllDatabases

@SourceUID = null ,

@SourcePWD = null ,

@OutFilePath = 'c:\a\' ,

@OutFileName = null ,

@WorkPath = 'c:\temp\' , -- no spaces

@SourceSVR = null ,

@Database = 'mydb'

\*/

declare @sql varchar(1000) ,

@cmd varchar(1000)

if @SourceSVR is null

begin

select @SourceSVR = @@servername

end

if right(@OutFilePath,1) <> '\'

begin

select @OutFilePath = @OutFilePath + '\'

end

if right(@WorkPath,1) <> '\'

begin

select @WorkPath = @WorkPath + '\'

end

select @OutFilePath = @OutFilePath + '"' + @SourceSVR + '"'

exec master..xp\_cmdshell @cmd

select @OutFilePath = @OutFilePath + '\'

select @sql =

'select name

from [' + @SourceSVR + '].master.dbo.sysdatabases

where name <> ''tempdb'''

if @Database is not null

begin

select @sql = @sql + ' and name = ''' + @Database + ''''

end

create table #tblDatabases (name varchar(128))

insert #tblDatabases

(name)

exec (@sql)

declare @FilePath varchar(256)

declare @name varchar(128) ,

@maxname varchar(128)

select @name = '' ,

@maxname = max(name)

from #tblDatabases

while @name < @maxname

begin

select @name = min(name) from #tblDatabases where name > @name

select @FilePath = @OutFilePath + '"' + @name + '"'

-- output current database name

select CurrentDatabase = @name

-- create output directory - will fail if already exists but ...

select @cmd = 'mkdir ' + @FilePath

exec master..xp\_cmdshell @cmd, no\_output

exec s\_ScriptAllObjectsInDatabase

@SourceDB = @name ,

@SourceUID = @SourceUID ,

@SourcePWD = @SourcePWD ,

@OutFilePath = @FilePath ,

@OutFileName = @OutFileName , -- null for separate file per object script

@WorkPath = @WorkPath ,

@SourceSVR = @SourceSVR

end

if coalesce(@Database, 'JOBS') = 'JOBS'

begin

select @FilePath = @OutFilePath + 'JOBS'

-- create output directory - will fail if already exists but ...

select @cmd = 'mkdir ' + @FilePath

exec master..xp\_cmdshell @cmd, no\_output

exec s\_ScriptObjects

@SourceDB = 'msdb' ,

@SourceObject = null , -- null for all objects

@SourceUID = @SourceUID ,

@SourcePWD = @SourcePWD ,

@OutFilePath = @FilePath ,

@OutFileName = @OutFileName , -- null for separate file per object script

@ObjectType = 'JOBS' ,

@WorkPath = @WorkPath ,

@SourceSVR = @SourceSVR

end

if coalesce(@Database, 'DTS') = 'DTS'

begin

select @FilePath = @OutFilePath + 'DTS'

-- create output directory - will fail if already exists but ...

select @cmd = 'mkdir ' + @FilePath

exec master..xp\_cmdshell @cmd, no\_output

exec s\_ScriptObjects

@SourceDB = 'msdb' ,

@SourceObject = null , -- null for all objects

@SourceUID = @SourceUID ,

@SourcePWD = @SourcePWD ,

@OutFilePath = @FilePath ,

@OutFileName = @OutFileName , -- null for separate file per object script

@ObjectType = 'DTS' ,

@WorkPath = @WorkPath ,

@SourceSVR = @SourceSVR

end

GO

if exists (select \* from dbo.sysobjects where id = object\_id(N'[dbo].[s\_ScriptAllObjectsInDatabase]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)

drop procedure [dbo].[s\_ScriptAllObjectsInDatabase]

GO

Create procedure s\_ScriptAllObjectsInDatabase

@SourceDB varchar(128) ,

@SourceUID varchar(128) , -- null for trusted connection

@SourcePWD varchar(128) ,

@OutFilePath varchar(256) , -- Root path - will add directory for object types

@OutFileName varchar(128) , -- null for separate file per object script

@WorkPath varchar(256) ,

@SourceSVR varchar(128)

as

if right(@OutFilePath,1) <> '\'

begin

select @OutFilePath = @OutFilePath + '\'

end

if right(@WorkPath,1) <> '\'

begin

select @WorkPath = @WorkPath + '\'

end

set nocount on

declare @tblObjectType table (ObjectType varchar(50))

insert @tblObjectType select 'PROCEDURES'

insert @tblObjectType select 'FUNCTIONS'

insert @tblObjectType select 'TABLES'

insert @tblObjectType select 'VIEWS'

insert @tblObjectType select 'INDEXES'

insert @tblObjectType select 'TRIGGERS'

insert @tblObjectType select 'DEFAULTS'

insert @tblObjectType select 'RULES'

declare @FilePath varchar(256) ,

@cmd varchar(1000)

declare @ObjectType varchar(50) ,

@maxObjectType varchar(50)

select @ObjectType = '' ,

@maxObjectType = max(ObjectType)

from @tblObjectType

while @ObjectType < @maxObjectType

begin

select @ObjectType = min(ObjectType) from @tblObjectType where ObjectType > @ObjectType

select @FilePath = @OutFilePath + @ObjectType

-- create output directory - will fail if already exists but ...

select @cmd = 'mkdir ' + @FilePath

exec master..xp\_cmdshell @cmd, no\_output

exec s\_ScriptObjects

@SourceDB = @SourceDB ,

@SourceObject = null ,

@SourceUID = @SourceUID ,

@SourcePWD = @SourcePWD ,

@OutFilePath = @FilePath ,

@OutFileName = null , -- null for separate file per object script

@ObjectType = @ObjectType ,

@WorkPath = @WorkPath ,

@SourceSVR = @SourceSVR

end

GO

if exists (select \* from dbo.sysobjects where id = object\_id(N'[dbo].[s\_ScriptObjects]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)

drop procedure [dbo].[s\_ScriptObjects]

GO

CREATE procedure s\_ScriptObjects

@SourceDB varchar(128) ,

@SourceObject varchar(128) , -- null for all objects

@SourceUID varchar(128) , -- null for trusted connection

@SourcePWD varchar(128) ,

@OutFilePath varchar(256) ,

@OutFileName varchar(128) , -- null for separate file per object script

@ObjectType varchar(50) , -- PROCS, FUNCTIONS, TABLES, VIEWS, INDEXES

@WorkPath varchar(256) ,

@SourceSVR varchar(128)

as

/\*

exec s\_ScriptObjects

@SourceDB = 'TradarBe' ,

@SourceObject = 'tbl\_CQS\_Pricing\_BloombergData' , -- null for all objects

@SourceUID = null , -- null for trusted connection

@SourcePWD = null ,

@OutFilePath = 'c:\a\' ,

@OutFileName = null , -- null for separate file per object script

@ObjectType = 'TABLES' , -- PROCS, FUNCTIONS, TABLES, VIEWS, INDEXES

@WorkPath = 'c:\temp\' ,

@SourceSVR = 'SVR01'

\*/

set nocount on

declare @ScriptType int ,

@FileName varchar(256) ,

@tmpFileName varchar(256) ,

@buffer varchar(8000) ,

@Collection varchar(128) ,

@id int ,

@name varchar(128) ,

@subname varchar(128)

declare @context varchar(255) ,

@sql varchar(1000) ,

@rc int

if right(@OutFilePath,1) <> '\'

begin

select @OutFilePath = @OutFilePath + '\'

end

if right(@WorkPath,1) <> '\'

begin

select @WorkPath = @WorkPath + '\'

end

select @SourceDB = replace(replace(@SourceDB,'[',''),'[','')

select @ScriptType = 4 | 1 | 64 ,

@FileName = @OutFilePath + @OutFileName ,

@tmpFileName = @WorkPath + 'ScriptTmp.txt'

declare @objServer int ,

@objTransfer int ,

@strResult varchar(255) ,

@strCommand varchar(255)

-- get objects to script and object type

create table #Objects (name varchar(128), subname varchar(128) default null, id int identity(1,1))

if @SourceObject is not null

begin

insert #Objects

(name)

select @SourceObject

end

if @ObjectType = 'TABLES'

begin

if @SourceObject is null

begin

select @sql = 'select TABLE\_NAME, null '

select @sql = @sql + 'from [' + @SourceDB + '].INFORMATION\_SCHEMA.TABLES '

select @sql = @sql + 'where TABLE\_TYPE = ''BASE TABLE'''

end

select @Collection = 'tables'

end

else if @ObjectType in ('PROCS', 'PROCEDURES')

begin

if @SourceObject is null

begin

select @sql = 'select ROUTINE\_NAME, null '

select @sql = @sql + 'from [' + @SourceDB + '].INFORMATION\_SCHEMA.ROUTINES '

select @sql = @sql + 'where ROUTINE\_TYPE = ''PROCEDURE'''

end

select @Collection = 'storedprocedures'

end

else if @ObjectType = 'FUNCTIONS'

begin

if @SourceObject is null

begin

select @sql = 'select ROUTINE\_NAME, null '

select @sql = @sql + 'from [' + @SourceDB + '].INFORMATION\_SCHEMA.ROUTINES '

select @sql = @sql + 'where ROUTINE\_TYPE = ''FUNCTION'''

end

select @Collection = 'userdefinedfunctions'

end

else if @ObjectType = 'VIEWS'

begin

if @SourceObject is null

begin

select @sql = 'select TABLE\_NAME, null '

select @sql = @sql + 'from [' + @SourceDB + '].INFORMATION\_SCHEMA.VIEWS '

select @sql = @sql + 'where TABLE\_NAME not like ''sys%'''

end

select @Collection = 'views'

end

else if @ObjectType = 'INDEXES'

begin

if @SourceObject is null

begin

select @sql = 'select o.name, i.name '

select @sql = @sql + 'from [' + @SourceDB + ']..sysobjects o, [' + @SourceDB + ']..sysindexes i '

select @sql = @sql + 'where o.type = ''U'' '

select @sql = @sql + 'and i.id = o.id and i.indid <> 0 '

select @sql = @sql + 'and i.name not like ''\_WA\_%'''

select @sql = @sql + 'and o.name not like ''dtprop%'''

select @sql = @sql + 'and i.name not in (select name from [' + @SourceDB + ']..sysobjects)'

end

select @Collection = 'tables'

end

else if @ObjectType = 'TRIGGERS'

begin

if @SourceObject is null

begin

select @sql = 'select o2.name, o.name '

select @sql = @sql + 'from [' + @SourceDB + ']..sysobjects o, [' + @SourceDB + ']..sysobjects o2 '

select @sql = @sql + 'where o.xtype = ''TR'' '

select @sql = @sql + 'and o.parent\_obj = o2.id '

end

select @Collection = 'tables'

end

else if @ObjectType = 'DEFAULTS'

begin

if @SourceObject is null

begin

select @sql = 'select o.name, null '

select @sql = @sql + 'from [' + @SourceDB + ']..sysobjects o '

select @sql = @sql + 'where o.type = ''D'' and o.parent\_obj = ''0'''

end

select @Collection = 'Defaults'

end

else if @ObjectType = 'RULES'

begin

if @SourceObject is null

begin

select @sql = 'select o.name, null '

select @sql = @sql + 'from [' + @SourceDB + ']..sysobjects o '

select @sql = @sql + 'where type = ''R'''

end

select @Collection = 'Rules'

end

else if @ObjectType = 'JOBS'

begin

if @SourceObject is null

begin

select @sql = 'select j.name, null '

select @sql = @sql + 'from msdb..sysjobs j '

end

select @Collection = 'jobs'

end

else if @ObjectType = 'DTS'

begin

select @sql = 'dtsrun /NScript\_DTS\_Packages /S(local) /E '

+ '/A"ServerName":8="' + @SourceSVR + '" '

+ '/A"Path":8="' + @OutFilePath + '" '

+ '/A"UserName":8="' + coalesce(@SourceUID,'') + '" '

+ '/A"Password":8="' + coalesce(@SourcePWD,'') + '" '

exec master..xp\_cmdshell @sql

return

end

else

begin

select 'invalid @ObjectType'

return

end

if @SourceSVR <> @@servername

begin

select @sql = replace(@sql,'''','''''')

insert #Objects (name, subname) exec ('select \* from openquery(' + @SourceSVR + ',''' + @sql + ''')')

end

else

begin

insert #Objects (name, subname) exec (@sql)

end

-- create empty output file

if @OutFileName is not null

begin

select @sql = 'echo. > ' + @FileName

exec master..xp\_cmdshell @sql

end

-- prepare scripting object

select @context = 'create dmo object'

exec @rc = sp\_OACreate 'SQLDMO.SQLServer', @objServer OUT

if @rc <> 0 or @@error <> 0 goto ErrorHnd

if @SourceUID is null

begin

select @context = 'set integrated security ' + @SourceSVR

exec @rc = sp\_OASetProperty @objServer, LoginSecure, 1

if @rc <> 0 or @@error <> 0 goto ErrorHnd

end

select @context = 'connect to server ' + @SourceSVR

exec @rc = sp\_OAMethod @objServer , 'Connect', NULL, @SourceSVR , @SourceUID , @SourcePWD

if @rc <> 0 or @@error <> 0 goto ErrorHnd

select @context = 'scripting'

-- Script all the objects

select @id = 0

while exists (select \* from #Objects where id > @id)

begin

select @id = min(id) from #Objects where id > @id

select @name = name, @subname = subname from #Objects where id = @id

if @OutFileName is null

begin

select @FileName = @OutFilePath + 'dbo."' + @name + coalesce('[' + @subname + ']','') + '.sql"'

select @sql = 'echo. > ' + @FileName

exec master..xp\_cmdshell @sql

end

--select @sql = 'echo print ''Create = dbo.[' + @name + ']'+ coalesce('[' + @subname + ']','') + ''' >> ' + @FileName

--exec master..xp\_cmdshell @sql

if @ObjectType = 'INDEXES'

begin

Set @sql = 'databases("' + @SourceDB + '").' + @Collection + '("' + @name + '").indexes("' + @subname + '").script'

end

else if @ObjectType = 'TRIGGERS'

begin

Set @sql = 'databases("' + @SourceDB + '").' + @Collection + '("' + @name + '").triggers("' + @subname + '").script'

end

else if @ObjectType = 'JOBS'

begin

Set @sql = 'Jobserver.Jobs("' + @name + '").Script'

end

else

begin

Set @sql = 'databases("' + @SourceDB + '").' + @Collection + '("' + @name + '").script'

end

exec @rc = sp\_OAMethod @objServer, @sql , @buffer OUTPUT, @ScriptType , @tmpFileName

select @sql = 'type ' + @tmpFileName + ' >> ' + @FileName

exec master..xp\_cmdshell @sql

end

-- delete tmp file

select @sql = 'del ' + @tmpFileName

exec master..xp\_cmdshell @sql, no\_output

-- clear up dmo

exec @rc = sp\_OAMethod @objServer, 'Disconnect'

if @rc <> 0 or @@error <> 0 goto ErrorHnd

exec @rc = sp\_OADestroy @objServer

if @rc <> 0 or @@error <> 0 goto ErrorHnd

-- clear up temp table

drop table #Objects

return

ErrorHnd:

select 'fail', @context

GO

# OS related Scripts

## Get drive space information

SET NOCOUNT ON

-- Description: This script helps in obtaining drive space related information from your server.

-- It provides you with:

-- 1) Drive Letter

-- 2) Total Size in MB

-- 3) Used Space in MB

-- 4) Free Space in MB

-- 5) Total Percent of Free disk space

-- 6) Total percent of disk being used

-- 7) Total space occupied by SQL Server on that drive

-- 8) Current date and time when this information was generated (Good for maintaining history)

-- 9) Drive Type (Fixed Drives / CD-Rom / DVDROM

-- Note: If you are using this script on SQL Server 2005, you must enable "OLE Automation" feature from SAC (Surface area configuration)

IF (LEFT(CONVERT(VARCHAR(10),SERVERPROPERTY('ProductVersion')),1) = '9')

BEGIN

EXEC sp\_configure 'OLE Automation',1

RECONFIGURE WITH OVERRIDE

END

Create Table #tmp (DBName Varchar(25),Location Varchar(60),Size Varchar(8),Device Varchar(30))

If Exists (Select Name from SysObjects where Name = '#SQL\_DiskINFO')

Begin

Drop Table #SQL\_DiskINFO

End

Else

Begin

CREATE TABLE [#SQL\_DiskINFO] (

[Drive] [varchar] (3) NULL ,

[TotalSize\_MB] [int] NULL ,

[UsedSpace\_MB] [int] NULL ,

[Freespace\_MB] [int] NULL ,

[TotalPercentFree] Varchar(20) NULL ,

[TotalPercentUsed] Varchar(20) NULL,

[SQLOccupied\_MB] Int Null,

[DateRecorded] Varchar(20),

[DriveType] Varchar(20)

)

End

Exec SP\_MSForEachDB 'Use ? Insert into #Tmp Select Convert(Varchar(25),DB\_Name())''Database'',Convert(Varchar(60),FileName),Convert(Varchar(8),Size/128)''Size in MB'',Convert(Varchar(30),Name) from SysFiles'

DECLARE @hr int,@fso int,@mbtotal BIGINT,@TotalSpace int,@MBFree int,@Percentage int,@SQLDriveSize int

Declare @size bigint, @dtype int, @drivetype varchar(20)

DECLARE @drive Varchar(1),@fso\_Method varchar(255)

SET @mbTotal = 0

EXEC @hr = master.dbo.sp\_OACreate 'Scripting.FilesystemObject', @fso OUTPUT

CREATE TABLE #space (drive char(1), mbfree int)

INSERT INTO #space EXEC master.dbo.xp\_fixeddrives 4

INSERT INTO #space EXEC master.dbo.xp\_fixeddrives 3

Declare CheckDrives Cursor For Select drive,MBfree From #space

Open CheckDrives

Fetch Next from CheckDrives into @Drive,@MBFree

While(@@FETCH\_STATUS=0)

Begin

SET @fso\_Method = 'Drives("' + @drive + ':").TotalSize'

Select @SQLDriveSize=sum(Convert(Int,Size)) from #Tmp where Substring(Location,1,1)=@drive

EXEC @hr = sp\_OAMethod @fso, @fso\_method, @size OUTPUT

SET @fso\_Method = 'Drives("' + @drive + ':").DriveType'

EXEC @hr = sp\_OAMethod @fso, @fso\_method, @dtype OUTPUT

Select @drivetype = case @dtype when 2 then 'Fixed drives' when 4 then 'CD-Rom / DVD' else 'Unknown' end

set @mbtotal = 0

SET @mbtotal = @mbtotal + @size / (1024.0 \* 1024.0)

Insert into #SQL\_DiskINFO Values

(

@Drive+':\',

@MBTotal,

@MBTotal-@MBFree,

@MBFree,

Convert(Varchar,100 \* round(@MBFree,2) / round(@MBTotal,2))+'%',

Convert(Varchar,100 - 100 \* round(@MBFree,2) / round(@MBTotal,2))+'%',

@SQLDriveSize, DATENAME(month, getdate())+' '+DATENAME(dd, getdate())+' '+DateName(yy,getdate())+' '+DateName(hh,getdate())+':'+DateName(Mi,getdate()),

@drivetype

)

FETCH NEXT FROM CheckDrives INTO @drive,@mbFree

END

Select \* from #SQL\_DiskINFO

drop table #tmp

drop table #space

drop table #SQL\_DiskINFO

close CheckDrives

deallocate CheckDrives

GO

## Directory size with file count

if object\_id('tempdb..#t') is not null drop table #t

if object\_id('tempdb..#t1') is not null drop table #t1

if object\_id('tempdb..#t2') is not null drop table #t2

create table #t (keyid int IDENTITY(1,1) , col1 varchar(500))

insert #t exec xp\_cmdshell 'dir "C:\" /s /-c' -- provide the derictory path

--insert #t exec xp\_cmdshell 'dir "c:\kailash\\*.sql" /s /-c'

select \*

into #t1

from #t

where col1 like '%directory of%' or col1 like '%file(s)%'

select Keyid, col1, (select top 1 col1 from #t1

where keyid > a.keyid

order by keyid) col2

into #t2

from #t1 a

where col1 like '%directory of%'

select col1, substring(col2, 1, charindex(')', col2)) NoOfFiles,

str(cast(substring(rtrim(replace(col2, 'bytes', '')), charindex(')', col2)+1, len(col2)) as bigint) \* 1.00 / 1024 / 1024, 12, 2) SizeInMB

from #t2

order by 3 desc

# Replication and Mirroring Scripts

## Command for adding ‘NOT For Replication' for Identity -- without breaking replication

-- For 2005 onwards

EXEC sp\_msforeachtable @command1 = '  
declare @int int  
set @int =object\_id("?")  
EXEC sys.sp\_identitycolumnforreplication @int, 1'

-- For SQL 2000

update syscolumns

set colstat = colstat | 0x0008

where colstat & 0x0008 = 0 -- ie not already "not for replication"

and colstat & 0x0001 <> 0 -– identity columns

## Monitor database mirroring

CREATE PROCEDURE usp\_Mirror\_Sync\_Report

@sendmail bit = 0 OUTPUT

AS

BEGIN

SET NOCOUNT ON

Declare @dbname varchar(200), @str nvarchar(2000),@htmlcode varchar(4000)

set @sendmail=0

Create Table #Mirror\_Principal

(

database\_name varchar(50), role int, mirroring\_state int, witness\_status int,

log\_generation\_rate int, unsent\_log int, send\_rage int, unrestored\_log int,

recovery\_rate int, transaction\_delay int, transaction\_per\_sec int, average\_delay int,

time\_recorded datetime, time\_behind datetime, local\_time datetime)

DECLARE mirror CURSOR local for

select d.name from sys.database\_mirroring dm

join sys.databases d on (dm.database\_id=d.database\_id)

where mirroring\_guid is not null order by d.name

open mirror

fetch next from mirror into @dbname

while (@@fetch\_status = 0)

BEGIN

select @str = 'msdb.sys.sp\_dbmmonitorresults @database\_name = ' + LTRIM(RTRIM(@dbname)) + ', @mode = 0, @update\_table = 0'

--print @str

Insert #Mirror\_Principal

EXEC sp\_executesql @str

fetch next from mirror into @dbname

END

Delete from #Mirror\_Principal Where mirroring\_state = 4

IF EXISTS ( Select mirroring\_state from #Mirror\_Principal

where datediff(mi,time\_behind,time\_recorded) > 30 )

BEGIN

set @sendmail=1

PRINT '<HTML><HEAD> <TITLE> Sync. Status </TITLE> </HEAD> <BODY>

<h2 align="center"> <font face="arial" size="3"> Following Mirrored Databases are out of Sync. </font> </h2>

<TABLE border="1" align="center" cellpadding="2" cellspacing="0">

<TR> <TH bgcolor="#FFCC99"> <font face="arial" size="2"> Principal Server </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Prinicipal DB </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2"> Mirroring Role </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2"> Mirrored Server </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2"> Witness Server </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2"> Witness Server State </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2"> Unsent Log (KB) </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2"> Latency (Minutes) </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2"> Server Time </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2"> Mirroring State </font></TH> </TR>'

select '

<TR> <TD> <font face="arial" size="2">' + @@servername + '</font></TD>

<TD> <font face="arial" size="2">' + B.name + '</font></TD>

<TD> <font face="arial" size="2">' +

Case B.mirroring\_role When 1 then 'Principal' when 1 then 'Disconnected'

When 2 then 'Mirror' else 'Database is not online' END

+ '</font></TD>

<TD> <font face="arial" size="2">' + B.Mirroring\_Partner\_instance + '</font></TD>

<TH> <font face="arial" size="2">' +

Case When B.mirroring\_witness\_name='' then 'No Witness Exist' else B.mirroring\_witness\_name End

+ '</font></TH>

<TH> <font face="arial" size="2">' +

Case B.mirroring\_witness\_state when 0 then 'Unknown' when 1 then 'Connected' when 2 then 'Disconnected'

else 'No witness exists or the database is not online' end

+ '</font></TH>

<TD> <font face="arial" size="2">' + cast(A.unsent\_log as varchar) + '</font></TD>

<TD> <font face="arial" size="2">' + cast(datediff(minute,time\_behind,time\_recorded) as varchar) + '</font></TD>

<TD> <font face="arial" size="2">' + cast(A.local\_time as varchar) + '</font></TD>

<TD> <font face="arial" size="2">' +

Case A.mirroring\_state When 0 then 'Suspended' when 1 then 'Disconnected'

when 2 then 'Synchronizing' when 3 then 'Pending Failover'

when 4 then 'Synchronized' else 'Database is not online' END

+ '</font></TD></TR>' as HTML\_Code into #disp

from #Mirror\_Principal A join

(

select d.name, dm.mirroring\_partner\_instance, dm.mirroring\_role ,dm.mirroring\_state,

dm.mirroring\_witness\_state,dm.mirroring\_safety\_level,

dm.mirroring\_witness\_name,dm.mirroring\_state\_desc from sys.database\_mirroring dm

join sys.databases d on (dm.database\_id=d.database\_id)

where mirroring\_guid is not null

) as B

ON A.database\_name = B.name

where datediff(mi,A.time\_behind,A.time\_recorded) > 30

declare display CURSOR for

select HTML\_code from #disp

open display

fetch next from display into @htmlcode

while (@@fetch\_status = 0)

BEGIN

PRINT @htmlcode

fetch next from display into @htmlcode

END

PRINT '</TABLE> </font></BODY> </HTML>'

drop table #disp

CLOSE display

deallocate display

END

drop table #Mirror\_Principal

close mirror

deallocate mirror

END

## Replication synchronization monitoring script

EXEC xp\_cmdshell 'osql -S InstanceName -U sa -P password -d msdb -Q "Exec msdb.dbo.usp\_Tran\_Sync\_Report" -o N:\Syncreport\SyncReport.htm'

CREATE PROCEDURE usp\_Tran\_Sync\_Report

@sendmail bit = 0 OUTPUT

AS

BEGIN

SET NOCOUNT ON

Declare @pubname varchar(200), @str varchar(2000),@htmlcode varchar(8000),

@srvname varchar(50), @publisherdb varchar(100),@logreadername varchar(300)

set @sendmail=0

Create Table #distribution\_agent

(

subscriber varchar(50), status int, subscriber\_db varchar(50), type tinyint, distribtn\_agent varchar(200),

last\_action varchar(2000), action\_time datetime, start\_time datetime, duration bigint, delivery\_rate numeric(9,2),

delivery\_latency bigint, delivered\_transactions bigint, delivered\_command bigint, delivery\_time int,

average\_commands bigint, error\_id bigint, job\_id varbinary(100), local\_job int,

profile\_id int, agent\_id int, last\_timestamp varbinary(50), offload\_enabled int, offload\_server varchar(30),

subscriber\_type int

)

Create Table #LogReader\_Agent

(

publisher varchar(200), agent\_name varchar(100), publisher\_db varchar(50), status int,

action\_time varchar(2000), last\_action varchar(500)

)

Create Table #disp ( HTML\_Code varchar(8000) )

------------------------ For Distribution Agents -----------------------------

DECLARE pub CURSOR for

Select srvname, publisher\_db, publication

from distribution.dbo.MSPublications A

JOIN master.dbo.sysservers B

ON A.publisher\_id = B.srvid

where publication\_type <> 1

open pub

fetch next from pub into @srvname,@publisherdb,@pubname

while (@@fetch\_status = 0)

BEGIN

select @str = 'distribution.dbo.sp\_MSenum\_subscriptions @publisher = N''' + LTRIM(RTRIM(@srvname)) + ''', @publisher\_db = N''' + LTRIM(RTRIM(@publisherdb)) + ''', @publication = N''' + LTRIM(RTRIM(@pubname)) + ''', @exclude\_anonymous = 0'

-- print @str

Insert into #distribution\_agent

EXEC (@str)

fetch next from pub into @srvname,@publisherdb,@pubname

END

--Delete from #distribution\_agent where status IN (3,4)

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

------------------------ For Log Reader Agents -----------------------------

INSERT INTO #LogReader\_Agent

Select D.srvname as [Publisher], C.name as [Agent\_Name], C.publisher\_db, A.runstatus,

A.time as [Action\_Time], A.comments as [Last\_Action] from distribution.dbo.MSLogReader\_History as [A]

Join

(

select Agent\_id, max(time) time

from distribution.dbo.MSlogreader\_history

group by Agent\_id

) AS [B]

On A.Agent\_id = B.Agent\_id

And A.time = B.time

Join distribution.dbo.MSLogreader\_Agents as [C]

On A.agent\_id = C.id

Join master.dbo.sysservers as [D]

ON C.publisher\_id = D.srvid

-- Delete from #LogReader\_Agent where status IN (3,4)

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

IF EXISTS ( Select status from #distribution\_agent where action\_time < dateadd(mi,-30,getdate()) OR error\_id<>0)

OR EXISTS (Select status from #LogReader\_Agent where action\_time < dateadd(mi,-30,getdate()))

BEGIN

set @sendmail=1

Insert into #disp

Select '<HTML><HEAD> <TITLE> Sync. Status </TITLE> </HEAD> <BODY>

<h2 align="center"> <font face="arial" size="3"> Transactional Replication Sync Status </font> </h2>

'

IF EXISTS (Select status from #distribution\_agent where action\_time < dateadd(mi,-30,getdate()) OR error\_id<>0)

BEGIN

Insert Into #disp

Select '

<TABLE border="1" align="center" cellpadding="2" cellspacing="0">

<TR> <TH colspan=6 align="center" bgcolor="#FFCC77"><font face="arial" size="2">Distribution Agents </font> </THD> </TR>

<TR> <TH bgcolor="#FFCC99"> <font face="arial" size="2"> Publisher </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Publisher DB </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Publication </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Subscriber </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Last\_Sync\_Time </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Error Desc </font></TH> </TR>'

Insert Into #disp

select distinct '

<TR> <TD> <font face="arial" size="2">' + C.srvname + '</font></TD>

<TD> <font face="arial" size="2">' + D.Publisher\_DB + '</font></TD>

<TD> <font face="arial" size="2">' + D.Publication + '</font></TD>

<TD> <font face="arial" size="2">' + A.subscriber + '</font></TD>

<TD> <font face="arial" size="2">' + CONVERT(VARCHAR(30),A.action\_time,121) + '</font></TD>

<TD> <font face="arial" size="2">' + A.last\_action + '</font></TD></TR>' as HTML\_Code

from #distribution\_agent A join

(

select publication\_id, agent\_id, publisher\_id from distribution.dbo.MSSubscriptions

group by publication\_id, agent\_id, publisher\_id

) as B

on A.agent\_id = B.agent\_id

join master.dbo.sysservers C

on B.publisher\_id = C.srvid

join distribution.dbo.MSpublications D

on B.publisher\_id = D.publisher\_id

and B.publication\_id = D.publication\_id

Where A.action\_time < dateadd(mi,-30,getdate())

OR error\_id<>0

Insert into #disp

Select '</TABLE>'

END

IF EXISTS (Select status from #LogReader\_Agent where action\_time < dateadd(mi,-30,getdate()))

BEGIN

Insert Into #disp

Select '<br><br><TABLE border="1" align="center" cellpadding="2" cellspacing="0">

<TR> <TH colspan=6 align="center" bgcolor="#FFCC77"><font face="arial" size="2"> Log Reader Agents </font> </TH> </TR>

<TR> <TH bgcolor="#FFCC99"> <font face="arial" size="2"> Distribution ServerName </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2"> Publisher </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2"> Publisher DB </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2"> Publication </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Last\_Sync\_Time </font></TH>

<TH bgcolor="#FFCC99"> <font face="arial" size="2">Error Desc </font></TH> </TR>'

Insert Into #disp

select ' <TR> <TD> <font face="arial" size="2">' + @@servername + '</font></TD>

<TD> <font face="arial" size="2">' + publisher + '</font></TD>

<TD> <font face="arial" size="2">' + publisher\_db + '</font></TD>

<TD> <font face="arial" size="2">' + 'All' + '</font></TD>

<TD> <font face="arial" size="2">' + CONVERT(VARCHAR(30),action\_time,121) + '</font></TD>

<TD> <font face="arial" size="2">' + last\_action + '</font></TD></TR>' as HTML\_Code

from #LogReader\_Agent

Where action\_time < dateadd(mi,-30,getdate())

Insert into #disp

Select '</TABLE>'

END

declare display CURSOR for

select HTML\_code from #disp

open display

fetch next from display into @htmlcode

while (@@fetch\_status = 0)

BEGIN

PRINT @htmlcode

fetch next from display into @htmlcode

END

PRINT '</font></BODY> </HTML>'

drop table #disp

CLOSE display

deallocate display

END

drop table #distribution\_agent

drop table #LogReader\_Agent

CLOSE pub

deallocate pub

END

## Merge Replication Status

BEGIN

SET NOCOUNT ON

Declare @pubname varchar(200), @str varchar(2000),@message varchar(8000)

Create Table #Details

(

subscriber varchar(50), status int, subscriber\_db varchar(50), type tinyint, agent\_name varchar(200),

last\_action varchar(2000), action\_time datetime, start\_time datetime,duration bigint, delivery\_rate numeric(5,2),

publisher\_insertcount bigint, publisher\_updatecount bigint, publisher\_deletecount bigint, publisher\_conflicts int,

subscriber\_insertcount bigint, subscriber\_updatecount bigint, subscriber\_deletecount bigint,

subscriber\_conflicts int, error\_id int, job\_id varbinary(100), local\_job int,profile int,

agent\_id int, last\_timestamp varbinary(50), offload\_enabled int, offload\_server varchar(30), subscriber\_type int

)

DECLARE pub CURSOR for

Select publication from distribution.dbo.MSPublications

open pub

fetch next from pub into @pubname

while (@@fetch\_status = 0)

BEGIN

select @str = 'distribution.dbo.sp\_MSenum\_merge\_subscriptions @publisher = N''CORPSVR15'', @publisher\_db = N''Global'', @publication = N''' + LTRIM(RTRIM(@pubname)) + ''', @exclude\_anonymous = 0'

--print @str

Insert into #Details

EXEC (@str)

fetch next from pub into @pubname

END

IF EXISTS ( Select B.Publication

from #Details A join distribution.dbo.MSMerge\_agents B

on A.agent\_id = B.id

Where a.action\_time < dateadd(mi,-60,getdate())

)

BEGIN

select @@servername ServerName ,

B.Publisher\_DB PublisherDB ,

B.Publication ,

A.subscriber ,

A.action\_time ,

A.last\_action

from #Details A join distribution.dbo.MSMerge\_agents B

on A.agent\_id = B.id

Where a.action\_time < dateadd(mi,-60,getdate())

END

drop table #Details

CLOSE pub

deallocate pub

END

## Start Distribution Agent

IF OBJECT\_ID('tempdb.dbo.#FailedDistAgents') IS NOT NULL DROP TABLE #FailedDistAgents

SELECT DISTINCT a.name

INTO #FailedDistAgents

FROM distribution..MSdistribution\_agents a

JOIN msdb..sysjobs b ON a.name = b.name AND b.Enabled = 1

LEFT JOIN master..sysprocesses c ON a.name = c.program\_name

WHERE c.program\_name IS NULL

DECLARE @JobName AS VARCHAR(256)

DECLARE cur\_FailedDistAgent CURSOR FOR SELECT name FROM #FailedDistAgents

OPEN cur\_FailedDistAgent

Fetch\_curServer\_Next:

FETCH NEXT FROM cur\_FailedDistAgent INTO @JobName

WHILE @@FETCH\_STATUS = 0 BEGIN

EXEC msdb..sp\_start\_job @job\_name = @JobName

GOTO Fetch\_curServer\_Next

END

CLOSE cur\_FailedDistAgent

DEALLOCATE cur\_FailedDistAgents

## Start Log Reader Agent

IF OBJECT\_ID('tempdb.dbo.#FailedAgents') IS NOT NULL DROP TABLE #FailedAgents

SELECT DISTINCT a.name

INTO #FailedAgents

FROM distribution..mslogreader\_agents a

LEFT JOIN master..sysprocesses b ON a.name = b.program\_name

WHERE b.program\_name IS NULL

-- INSERT #FailedAgents

-- SELECT DISTINCT a.name

-- FROM main\_replication..mslogreader\_agents a

-- LEFT JOIN master..sysprocesses b ON a.name = b.program\_name

-- WHERE b.program\_name IS NULL

DECLARE @JobName AS VARCHAR(256)

DECLARE cur\_FailedAgents CURSOR FOR SELECT name FROM #FailedAgents

OPEN cur\_FailedAgents

Fetch\_curServer\_Next:

FETCH NEXT FROM cur\_FailedAgents INTO @JobName

WHILE @@FETCH\_STATUS = 0 BEGIN

EXEC msdb..sp\_start\_job @job\_name = @JobName

GOTO Fetch\_curServer\_Next

END

CLOSE cur\_FailedAgents

DEALLOCATE cur\_FailedAgents

# Other Scripts

## Script to get Server up time

DECLARE @serverstarttime datetime

SELECT @serverstarttime = login\_time

FROM master..sysprocesses WHERE spid = 1

PRINT ' Server Start : ' + CONVERT(CHAR(25), @serverstarttime, 100)

PRINT ' Uptime (Days) : ' + CONVERT(CHAR(25), DATEDIFF(Day, @serverstarttime, GETDATE()))

## Script to get comma Separated values in Row

declare @a varchar(100)

declare @b varchar(4000)

set @a = 'ram,raj,rajesh,kiran'

set @b = 'SELECT ''' + REPLACE(LTRIM(RTRIM(@a)),',','''' + ' AS Result'+ CHAR(10) + 'UNION ALL' + CHAR(10)+ 'SELECT ''') + ''''

--select @b

EXEC (@b)

## Shrink Data\Log file in chunk with Stop Time Paramaeter

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

/\*

EXEC DV\_SHRINKDATA\_LOGFILE

@chunkSize =100, -- Will Shrunk in 100 MB

@percentfree =10, -- Shrink keeping 10% free space

@Print =1, -- IF @Print= 1 Print the DBCC Command for LogginG purpose else IF @Print= 0 Don't Print the DBCC Command

@exec =0, -- IF @exec=1 Execute the DBCC SHRINK () ELSE IF @exec=0 DO NOT Execute DBCC SHRINK (

@ONLY\_LOG =0, -- IF @ONLY\_LOG=0 (SHRINK BOTH DATA AND LOG FILE) ELSE @ONLY\_LOG=1 (SHRINK ONLY LOG FILE)

@DbName ='ashsqldb', -- SPECIFY DATABASE NAME TO EXECUTE AGAINST SINGLE DATABASE

@StopAfter =150 -- in min

\*/

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

--Version Modifer Date Description

--1.0 Ajay (DATAVAIL) July 29 2010 Shrink Data and log file base on condition

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

--sp\_rename 'DV\_SHRINKDATA\_LOGFILE','DV\_SHRINKDATA\_LOGFILE\_Bkp\_23082010'

ALTER PROCEDURE DV\_SHRINKDATA\_LOGFILE

@chunkSize int,

@percentfree float = 10,

@Print bit =1,

@exec bit =0,

@ONLY\_LOG int =1,

@DbName varchar(500)=NULL,

@StopAfter int = 0

AS

--Declare @DbName varchar(100)

Declare @cmd varchar(5000)

--Declare @percentfree float = 10

--Declare @Print bit =1

--Declare @exec bit =0

--DECLARE @ONLY\_LOG int =1

--Declare @chunkSize int

--SET @chunkSize=500

declare @Stoptime datetime

--declare @status bit

SET @Stoptime=dateadd(mi , @StopAfter, getdate())

if @StopAfter = 0

--begin

return

--end

--else

--begin

SET @percentfree = @percentfree/100

Declare ShrinkallDB\_Cur cursor for

select name

from sysdatabases where dbid>4

AND name = CASE WHEN LEN(ISNULL(@DbName,''))= 0 THEN name ELSE @DbName END

Open ShrinkallDB\_Cur fetch next from ShrinkallDB\_Cur into @DbName

while (@@fetch\_status = 0)

begin

Print 'Database Name : ' + @DbName

SET @cmd ='

USE ['+@DbName+']

Declare @Print bit

Declare @exec bit

declare @stoptime datetime

--Declare @ONLY\_LOG bit

SET @Print='+ convert(varchar,@Print)+'

SET @exec='+ convert(varchar,@exec)+ '

SET @stoptime='''+ convert(varchar,@Stoptime) +'''

DECLARE @Fileid INT, @chunkSize INT

DECLARE @flSize DECIMAL(18,2), @spFree DECIMAL(18,2), @ActSize INT, @bufSpace INT

DECLARE @flName VARCHAR(100), @SQLstr VARCHAR(1000)

SET @chunkSize ='+convert(varchar,@chunkSize)+' -- No of MB to be shrunk in chunks.

DECLARE cur\_SHRINK CURSOR FOR

SELECT fileid, name,

CAST((size \* 8 / 1024.0) AS DECIMAL(18,2)) AS FILESIZE,

CAST((size \* 8 / 1024.0) - (FILEPROPERTY(name,''SpaceUsed'')/ 128.0) AS DECIMAL(15,2)) SPACEFREE

FROM sysfiles

WHERE groupid '+ CASE WHEN @ONLY\_LOG=1 THEN '=0' ELSE '>=0' END +'

OPEN cur\_SHRINK

FETCH NEXT FROM cur\_SHRINK

INTO @Fileid, @flName, @flSize, @spFree

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @ActSize = @flSize

SET @bufSpace = @flSize - @spFree + ((@flSize - @spFree)\*'+convert(varchar,@percentfree)+')

--PRINT RTRIM(@flName) + ''...''

--PRINT @ActSize

-- set @Stoptime = getdate()

--set @status =0

WHILE @ActSize > @bufSpace --OR @status=1

BEGIN

SET @ActSize = @ActSize - @chunkSize

IF @ActSize > @bufSpace

BEGIN

--SET @ActSize = @bufSpace

SET @SQLstr = ''DBCC SHRINKFILE('' + CAST(@Fileid AS VARCHAR) + '','' + CAST(@ActSize AS VARCHAR) + '')''

If @Print=1

PRINT @SQLstr

if @exec=1

EXEC(@SQLstr)

END

if @stoptime <= dateadd(mi,-15,getdate())

--set @status =1

BREAK

--GOTO OUT

END

FETCH NEXT FROM cur\_SHRINK

INTO @Fileid, @flName, @flSize, @spFree

END

--OUT:

CLOSE cur\_SHRINK

DEALLOCATE cur\_SHRINK

'

EXEC (@cmd)

-- Print @cmd

fetch next from ShrinkallDB\_Cur into @DbName

End

close ShrinkallDB\_Cur

deallocate ShrinkallDB\_Cur

--end

## Shrink File in Chunk

DECLARE @Fileid INT, @chunkSize INT

DECLARE @flSize DECIMAL(18,2), @spFree DECIMAL(18,2), @ActSize INT, @bufSpace INT

DECLARE @flName VARCHAR(100), @SQLstr VARCHAR(1000)

DECLARE @logicalfilename VARCHAR(100)

-- sp\_helpfile

set @logicalfilename = 'AdventureWorks\_Log'

SET @chunkSize = 500 -- No of MB to be shrunk in chunks.

SELECT \* from sysfiles

DECLARE cur\_SHRINK CURSOR FOR

SELECT Fileid, Name,

CAST((SIZE \* 8 / 1024.0) AS DECIMAL(18,2)) AS FILESIZE,

CAST((SIZE \* 8 / 1024.0) - (FILEPROPERTY(Name,'SpaceUsed')/ 128.0) AS DECIMAL(15,2)) SPACEFREE

FROM SYSFILES

WHERE name = @logicalfilename

OPEN cur\_SHRINK

FETCH NEXT FROM cur\_SHRINK

INTO @Fileid, @flName, @flSize, @spFree

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @ActSize = @flSize

SET @bufSpace = @flSize - @spFree + ((@flSize - @spFree)\* 0.10)

PRINT RTRIM(@flName) + '...'

WHILE @ActSize > @bufSpace

BEGIN

SET @ActSize = @ActSize - @chunkSize

IF @ActSize < @bufSpace

SET @ActSize = @bufSpace

SET @SQLstr = 'DBCC SHRINKFILE(' + CAST(@Fileid AS VARCHAR) + ',' + CAST(@ActSize AS VARCHAR) + ')'

PRINT @SQLstr

EXEC(@SQLstr)

END

FETCH NEXT FROM cur\_SHRINK

INTO @Fileid, @flName, @flSize, @spFree

END

CLOSE cur\_SHRINK

DEALLOCATE cur\_SHRINK

## Get SQL Port

--SQL 2000/2005 Version

set nocount on

go

DECLARE @SqlPort Nvarchar(10)

DECLARE @instance\_name Nvarchar(30)

DECLARE @reg\_key Nvarchar(500)

Declare @value\_name Nvarchar(20)

if left(CAST(SERVERPROPERTY('ProductVersion')AS sysname),1) = '9'

BEGIN

select @instance\_name = CAST(SERVERPROPERTY('instancename')AS sysname)

if @instance\_name is NULL

BEGIN

set @reg\_key = 'SOFTWARE\Microsoft\MSSQLServer\MSSQlServer\SuperSocketNetLib\Tcp'

END

ELSE BEGIN

set @reg\_key = 'SOFTWARE\Microsoft\Microsoft SQL Server\' + @instance\_name + '\MSSQLServer\SuperSocketNetLib\Tcp'

END

EXEC master..xp\_regread @rootkey='HKEY\_LOCAL\_MACHINE',

@key=@reg\_key, @value\_name='TcpPort',

@value=@SqlPort output

select CAST(SERVERPROPERTY('ServerName')AS sysname) as ServerName, @SqlPort as Port

END

if left(CAST(SERVERPROPERTY('ProductVersion')AS sysname),1) = '8'

BEGIN

Create table #Port\_2000 (value nvarchar(20),Data nVarchar(10))

insert into #Port\_2000 exec master..xp\_instance\_regread 'HKEY\_LOCAL\_MACHINE', 'SOFTWARE\Microsoft\MSSQLServer\MSSQLServer\Supersocketnetlib\tcp', 'tcpPort'

select @SqlPort = Data from #Port\_2000

select CAST(SERVERPROPERTY('ServerName')AS sysname) as ServerName, @SqlPort as Port

drop table #Port\_2000

END

## SQL Server Property Information

--SQL 2000/2005 Version

set nocount on

go

DECLARE @SqlPort Nvarchar(10)

DECLARE @instance\_name Nvarchar(30)

DECLARE @reg\_key Nvarchar(500)

Declare @value\_name Nvarchar(20)

if left(CAST(SERVERPROPERTY('ProductVersion')AS sysname),1) = '9'

BEGIN

select @instance\_name = CAST(SERVERPROPERTY('instancename')AS sysname)

if @instance\_name is NULL

BEGIN

set @reg\_key = 'SOFTWARE\Microsoft\MSSQLServer\MSSQlServer\SuperSocketNetLib\Tcp'

END

ELSE BEGIN

set @reg\_key = 'SOFTWARE\Microsoft\Microsoft SQL Server\' + @instance\_name + '\MSSQLServer\SuperSocketNetLib\Tcp'

END

EXEC master..xp\_regread @rootkey='HKEY\_LOCAL\_MACHINE',

@key=@reg\_key, @value\_name='TcpPort',

@value=@SqlPort output

--select CAST(SERVERPROPERTY('ServerName')AS sysname) as ServerName, @SqlPort as Port

END

if left(CAST(SERVERPROPERTY('ProductVersion')AS sysname),1) = '8'

BEGIN

Create table #Port\_2000 (value nvarchar(20),Data nVarchar(10))

insert into #Port\_2000 exec master..xp\_instance\_regread 'HKEY\_LOCAL\_MACHINE', 'SOFTWARE\Microsoft\MSSQLServer\MSSQLServer\Supersocketnetlib\tcp', 'tcpPort'

select @SqlPort = Data from #Port\_2000

--select CAST(SERVERPROPERTY('ServerName')AS sysname) as ServerName, @SqlPort as Port

drop table #Port\_2000

END

Declare @ipLine varchar(200), @ip varchar(100)

Declare @pos int

set nocount on

set @ip = NULL

Create table #temp (ipLine varchar(200))

Insert #temp exec master..xp\_cmdshell 'ipconfig'

select @ipLine = ipLine

from #temp

where upper (ipLine) like '%IP ADDRESS%'

if (isnull (@ipLine,'\*\*\*') != '\*\*\*')

begin

set @pos = CharIndex (':',@ipLine,1);

set @ip = rtrim(ltrim(substring (@ipLine ,

@pos + 1 ,

len (@ipLine) - @pos)))

end

drop table #temp

set nocount off

select @@servername 'InstanceName', @ip as 'IP Addr', @SqlPort as 'SQL Port',

case

when charindex('8.0',cast(serverproperty('Productversion')as varchar(15)))>0 then '2000'

when charindex('9.0',cast(serverproperty('Productversion')as varchar(15)))>0 then '2005' end as 'SQL Server Version',

serverproperty('Edition') 'Edition',

serverproperty('Productversion') 'Productversion',

serverproperty('Productlevel') 'Productlevel',

case

when charindex('86',@@version)>0 then '32-Bit'

when charindex('64',@@version)>0 then '64-Bit' end as 'SQL Server Bit Version',

serverproperty('collation') 'Server Collation',@@version '@@Version'

## Script out Triggers using DMO (VB Script)

Dim fso

Dim iFile

Dim strTbl

Dim strFilename

strFilename = "d:\sql\_dmo\Triggers.sql"

Dim objDMO

Set objDMO = CreateObject("SQLDMO.SQLServer")

objDMO.loginsecure = true

objDMO.Connect "DV-L-AJAY"

Dim objDB

Set objDB = objDMO.Databases("YogoNationalBank")

'Dim oTable

For Each oTable In objDB.Tables

' MsgBox oTable.Script()

if mid(oTable.name, 1, 3) <> "sys" then

For Each oIdx In oTable.Triggers

strTbl = strTbl & oIdx.Script() & vbCrLf

Next

end if

Next

Set fso = CreateObject("Scripting.FileSystemObject")

Set iFile = fso.CreateTextFile(strFilename, True)

iFile.Write (strTbl)

iFile.Close

objDMO.DisConnect

Set objDMO = nothing

## Script out SQL Jobs using DMO (VB Script)

Dim conServer

Dim fso

Dim iFile

Dim oJB

Dim strJob

Dim strFilename

Const ioModeAppend = 8

Set conServer = CreateObject("SQLDMO.SQLServer")

conServer.LoginSecure = True

conServer.Connect "SV-L-Rajnikant"

strFilename = "D:\SQL\_DMO\JOBS.sql"

For Each oJB In conServer.JobServer.Jobs

strJob = strJob & "--------------------------------------------------" & vbCrLf

strJob = strJob & "-- SCRIPTING JOB: " & oJB.Name & vbCrLf

strJob = strJob & "--------------------------------------------------" & vbCrLf

strJob = strJob & oJB.Script() & vbCrLf

Next

Set conServer = Nothing

Set fso = CreateObject("Scripting.FileSystemObject")

Set iFile = fso.CreateTextFile(strFilename, True)

iFile.Write (strJob)

iFile.Close

Set fso = Nothing

## Script out Indexes using DMO (VB Script)

Dim fso

Dim iFile

Dim strTbl

Dim strFilename

strFilename = "d:\sql\_dmo\indexes.sql"

Dim objDMO

Set objDMO = CreateObject("SQLDMO.SQLServer")

objDMO.loginsecure = true

objDMO.Connect "Sv-l-rajnikant"

Dim objDB

Set objDB = objDMO.Databases("northwind")

'Dim oTable

For Each oTable In objDB.Tables

' MsgBox oTable.Script()

if mid(oTable.name, 1, 3) <> "sys" then

For Each oIdx In oTable.Indexes

strTbl = strTbl & oIdx.Script() & vbCrLf

Next

end if

Next

Set fso = CreateObject("Scripting.FileSystemObject")

Set iFile = fso.CreateTextFile(strFilename, True)

iFile.Write (strTbl)

iFile.Close

objDMO.DisConnect

Set objDMO = nothing