**ISSUE1: SUPECT MODE**

**Reason for suspect mode:**

1. **If one or more database files are not available.**
2. **If the entire database is not available.**
3. **If one or more database files are corrupted.**
4. **If a database resource is being held by the operating system.**

**How to recover?**

* **Scenario 1: If the file is full**

**Execute sp\_resetstatus.**

**Syntax: sp\_resetstatus database\_name**

**Use ALTER DATABASE to add a data file or log file to the database.**

**Stop and restart SQL Server.**

**With the extra space provided by the new data file or log file, SQL Server should be able to complete recovery of the database**

* **Scenario2: If the data file was damaged.**

**\* Take T.Log backup**

**\* Restore last Full backup**

**\* Restore T.Log backup**

**\* Database comes online**

* **Scenario3: If the T.Log file was damaged**

**\* Take any user defined db for example: MyDB**

**\* Check the current location of files**

**sp\_helpdb MyDB**

**\* Stop server**

**\* Move the T.Log file into different folder**

**\* Start server --> DB goes into suspect mode**

**Select databasepropertyex ('mydb','status')**

**Steps to Recover:**

**Step1: Make the db into single user**

**1) Alter database mydb set Single\_User**

**Step2: Set the db into emergency mode**

**2) Alter database mydb set Emergency**

**Step3: Run checkdb with required repair level**

**3) DBCC CheckDB ('mydb', REPAIR\_ALLOW\_DATA\_LOSS)**

**Step4: Set the db into multi user mode**

**4) Alter database mydb set Multi\_User**

**ISSUE 2: MOVING MASTER DATABASE**

**1. Create two folders and grant read write permissions to service account**

**d:\master\_data**

**e:\master\_log**

**2. Find the current path**

**sp\_helpdb master**

**3. Stop SQL Server**

**4. Move the files (master.mdf, mastlog.ldf) into new folders**

**5. Go to SSCM --> R.C on respective instance SQL Server Service -- properties --> Advanced --> Startup Parameters--> Change the path of data and Log file**

**-dd:\master\_data\master.mdf;-e....**

**-le:\master\_log\mastlog.ldf**

**6. Apply --> OK**

**7. Start the service. Go to SSMS --> check the new path**

**sp\_helpdb master**

**ISSUE 3: SHRINKDATABASE (DIFFERENT PROCESS)**

**If you ever want to transfer a large DB to a new one with more than one file, here is the way I am going to use (tested and approved)**

**1. Create a file which is as large as the data in your primary file (call it "buffer")**

**2. Empty the primary file (DBCC SHRINKFILE (<FILENAME>, EMPTYFILE))**

**3. Restart SQL Server Engine**

**4. Shrink the primary file to the Data size divided by the number of files you're gonna create (DBCC SHRINKFILE (<FILENAME>, NEWSIZE))**

**5. Create all the new files with the size of data divided by the number of files**

**6. Restrict their growth in order to fill the primary file in the next operation**

**7. Empty the buffer file (DBCC SHRINKFILE (BUFFER, EMPTYFILE))**

**8. Delete the buffer file (ALTER DATABASE REMOVE FILE (NAME=BUFFER))**

**9. Set final size of data files and unrestrict their growth according to the final configuration needed**

**ISSUE 4: FINDOUT TABLE & INDEX SIZE**

1. **Create the temp table for further querying**

**CREATE TABLE #temp (**

**rec\_id int IDENTITY (1, 1),**

**table\_name varchar(128),**

**nbr\_of\_rows int,**

**data\_space decimal(15,2),**

**index\_space decimal(15,2),**

**total\_size decimal(15,2),**

**percent\_of\_db decimal(15,12),**

**db\_size decimal(15,2))**

1. **Get all tables, names, and sizes**

**EXEC sp\_msforeachtable @command1="insert into #temp (no\_of\_rows, data\_space, index\_space) exec sp\_mstablespace '?'", @command2="update #temp set table\_name = '?' where rec\_id = (select max(rec\_id) from #temp)"**

1. **Set the total\_size and total database size fields**

**UPDATE #temp SET total\_size = (data\_space + index\_space), db\_size = (SELECT SUM(data\_space + index\_space) FROM #temp)**

1. **Set the percent of the total database size**

**UPDATE #temp SET percent\_of\_db = (total\_size/db\_size) \* 100**

1. **Get the data**

**SELECT \*FROM #temp ORDER BY total\_size DESC**

1. **Comment out the following line if you want to do further querying**

**DROP TABLE #temp**

**ISSUE 5: PAGE LEVEL RESTORE**

**SET NOCOUNT ON**

**USE master;**

**GO**

**CREATE DATABASE TestPageLevelRestore**

**ON**

**( NAME = TestPageLevelRestore,**

**FILENAME = 'D:\TestPageLevelRestore.mdf',**

**SIZE = 10)**

**LOG ON**

**( NAME = TestPageLevelRestore\_log,**

**FILENAME = 'D:\TestPageLevelRestore\_log.ldf',**

**SIZE = 5MB) ;**

**GO**

**Print 'Database TestPageLevelRestore Created'**

**ALTER DATABASE TestPageLevelRestore SET RECOVERY FULL**

**Print 'Recovery Model of database TestPageLevelRestore has been changed to FULL'**

**Use TestPageLevelRestore**

**GO**

**CREATE TABLE [Shift](**

**[ShiftID] tinyint IDENTITY(1,1) NOT NULL,**

**[Name] nvarchar(50) NOT NULL,**

**[StartTime] datetime NOT NULL,**

**[EndTime] datetime NOT NULL,**

**[ModifiedDate] datetime NOT NULL,**

**CONSTRAINT [PK\_Shift\_ShiftID] PRIMARY KEY CLUSTERED ([ShiftID] ASC)**

**)**

**Print 'Creation of Table "Shift" Completed'**

**SET IDENTITY\_INSERT [Shift] ON**

**INSERT [Shift] ([ShiftID], [Name], [StartTime], [EndTime], [ModifiedDate]) VALUES (1, N'Day', '1900-01-01 07:00:00.000', '1900-01-01 15:00:00.000', '1998-06-01 00:00:00.000')**

**INSERT [Shift] ([ShiftID], [Name], [StartTime], [EndTime], [ModifiedDate]) VALUES (2, N'Evening', '1900-01-01 15:00:00.000', '1900-01-01 23:00:00.000', '1998-06-01 00:00:00.000')**

**INSERT [Shift] ([ShiftID], [Name], [StartTime], [EndTime], [ModifiedDate]) VALUES (3, N'Night', '1900-01-01 23:00:00.000', '1900-01-01 07:00:00.000', '1998-06-01 00:00:00.000')**

**SET IDENTITY\_INSERT [Shift] OFF**

**Print 'Data Insertion to table "Shift" Completed'**

**BACKUP DATABASE TestPageLevelRestore TO DISK='D:\TestPageLevelRestore\_FullBackup.bak' WITH STATS=10**

**Print 'Full Backup Completed'**

**--To get the list of index ID's from which you can choose one to corrupt**

**Use TestPageLevelRestore**

**Select \* from sys.indexes where OBJECT\_NAME(object\_id)='Shift'**

**--To get the list of pages**

**DBCC IND ('TestPageLevelRestore', 'Shift',1)**

**-- To display the contents**

**DBCC TRACEON (3604);**

**GO**

**--TO get the page level data details**

**DBCC PAGE('TestPageLevelRestore',1,147,3);**

**--Get the Offset Value. This can be obtained by multiplying the page ID with 8192.**

**--Once you get the result copy the result and set the database to offline**

**SELECT 147\*8192 AS [OffSetValue]**

**USE MASTER**

**ALTER DATABASE TestPageLevelRestore SET OFFLINE**

**Print 'Database TestPageLevelRestore is set to Offline. Now Open the TestPageLevelRestore.mdf file in the hex editor and press ctrl+g to go the page where the index data is located.**

**Choose Decimal and paste the offset value.**

**once you go to the location, then manuplate the value and save the file and exit hex editor.**

**After manuplating data bring database online.'**

**--Run the below code after manipulating and exiting hex editor.**

**/\***

**USE MASTER**

**ALTER DATABASE TestPageLevelRestore SET ONLINE**

**Print 'Database TestPageLevelRestore is set to Online'**

**--Select the data and you will get error stating that the read failed at page (x:xxxx)**

**USE TestPageLevelRestore**

**Select \* from shift**

**select \* from sys.master\_files where DB\_NAME(database\_id)='TestPageLevelRestore'**

**--Now Restore the page**

**USE master**

**-- Need to complete roll forward. So Backup the log tail.**

**BACKUP LOG TestPageLevelRestore TO DISK = 'D:\TestPageLevelRestore\_log.bak' WITH INIT, NORECOVERY;**

**GO**

**Restore DATABASE TestPageLevelRestore Page='1:147' FROM DISK='D:\TestPageLevelRestore\_FullBackup.bak'**

**-- restore the tail log backup.**

**RESTORE LOG TestPageLevelRestore FROM DISK = 'D:\TestPageLevelRestore\_log.bak';**

**GO**

**--Verify**

**USE TestPageLevelRestore**

**Select \* from shift**

**\*/**

**SET NOCOUNT OFF**

ISSUE 6: Identifying and Correcting the Transaction Log Full for User DB’s

## Error:

## Subject: SQL Server Alert System: 'Full Log' occurred on CSQL2

## Error: 9002, Severity: 17, State: 6

**DESCRIPTION: The log file for database 'Dealix' is full. Back up the transaction log for the database to free up some log space.**

## Identifying:

* **Configured (Log Full) alert to notify whenever there is a Transaction log Full on the User db.**

**And also Scheduled a job “SHRINKFILE” which performs the below tasks in the solution to prevent the Log Full. It is scheduled to run on Every Wednesday and Sunday 12 AM server time.**

## Solution:

**If the Regular database Transaction logs runs out of space, this is indicated in the SQL ERRORLOG files, use the following process:**

1. **Free up (unallocated) the space used by the LOG portion of the database with the following command called from the master database:**

**USE master**

**GO**

**BACKUP LOG DBname WITH TRUNCATE\_ONLY  
GO**

**Note:**

1. **After you truncate a database LOG file, the SQL server documentation recommends that you back up your database. In case of a physical failure (for example a power down or hard disk error), the SQL server cannot recover from the transaction log, as it was just truncated.**
2. **After running this command, the LDF file has been reorganized to have a lot of unallocated space, but the database must be *shrunk* to release that space to the file system. (It still looks like a large file if you view it from a command prompt directory listing). See next example for how to shrink the database.**

***Shrinking a Database***

**You can shrink a database to release the unallocated or unused space (or both) to the file system with the following command:**

**USE master**

**GO**

**DBCC SHRINKDATABASE (*database*)**

**GO**

**You can also use the SQL Enterprise Manager to shrink a database by selecting the following menu**

**Items: Right click on the Database -> All Tasks -> Shrink Database.**

**ISSUE 7: Troubleshooting host name changes**

**When the machine name is changed where we have installed SQL Server, all the instances services are started but replication, Jobs, Alerts, Maintenance plans causes errors. Hence we have to rename the instance.**

**To rename instance we can use the following SP**

**Steps:**

1. **Check the old server name as follows**

**SELECT @@servername**

1. **Drop the server and add the new server name**

**SP\_DROPSERVER <oldName>**

**SP\_ADDSERVER <newName>, local**

1. **Restart the instance**
2. **Check the server name again**

**SELECT @@servername**

**ISSUE 8: Troubleshooting User Connections**

**One of a user is unable to connect to SQL Server. What may be scenarios and how to troubleshoot it?**

**Possible Scenarios**

**1. Error: 26**

**\* SQL Browser**

**\* Firewall**

**\* No connectivity between client and server**

**2. Error: 28**

**\* Instance TCP/IP was disabled**

**3. Error: 40**

**\* Instance service is not running**

**4. Error: 18456**

**\* Login failed. (Invalid login or pwd)**

**5. Expired Timeout**

**\* Network issue**

**\* Server is busy**

**\* In server max sessions are open**

**\* No available session memory**

**6. Connection Forcibly Closed**

**Update the client computer to the server version of the SQL Server Native Client.**

**7. In single user mode if any other service is connected with the db Engine, it doesn't allow connections.**

**ISSUE 9: Troubleshooting SQL Server Service Problems**

**My SQL Server service is not started. What may be the possible scenarios?**

**Possible Scenarios**

**\* Logon Failure**

**\* Problem with service account.**

**\* 3417**

**\* Files are not present in the respective path or there are no permissions on target folder where the files are not present.**

**\* 17113**

**\* Master files are moved to different location, but not mentioned in startup parameters.**

**\* Service cannot be started in timely fashion**

**\* Insufficient resources, try to stop some other instances and start again.**

**How to find error?**

**1. Using windows event log**

**\* start --> run --> eventvwr**

**\* System**

**\* In the right side check for the errors**

**2. Using SQL Server ErrorLog file**

**\* Go to respective instance LOG folder and open ErrorLog in notepad and check for the errors.**

**ISSUE 10: Troubleshooting database suspect mode-17207**

**Database has gone into suspect mode. How to handle this scenario?**

**Possible Scenarios**

**\* If the database files are corrupted or there is disk issue.**

**\* If restoration fails.**

**\* If data file was full.**

**Steps:**

1. **Check the error log for possible reasons.**
2. **If data file was damaged or disk failure happen.**

* **Take tail log backup.**
* **Restore full backup**
* **Restore recent differential backups**
* **Restore all log backups if any, made after recent differential backup.**
* **Restore tail log backup WITH RECOVERY.**

1. **If log file was damaged or disk failure happen.**

* **Try to take tail log backup with another copy of log file if available with RAID level.**
* **If the log file is not available then make it online by running the following commands where there may be data loss.**

**--step1: Make the db into single user**

**1) ALTER DATABASE <databaseName> SET Single\_User**

**--step2: Set the db into emergency mode**

**2) ALTER DATABASE <databaseName> SET Emergency**

**--step3: Run checkdb with required repair level**

**3) DBCC CHECKDB ('<databaseName>', REPAIR\_ALLOW\_DATA\_LOSS)**

**--step4: Set the db into multi user mode**

**4) ALTER DATABASE <databaseName> SET Multi\_User**

**ISSUE 11: Troubleshooting master database corrupted**

**One of an instance master database data file was corrupted and I was unable to start the server. How to troubleshoot this scenario?**

**Possible Scenarios**

**If the master files are corrupted or damaged, instance cannot be started. We have to rebuild master database then by running the server in single user mode we have to restore latest backup to get previous settings.**

**Steps**

1. **Check the error log for exact reason.**
2. **Rebuild master database as follows by running setup from**

**C:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\Release**

**For windows authentication:**

**setup /ACTION=REBUILDDATABASE /QUIET /INSTANCENAME=<instance   
name> /SQLSYSADMINACCOUNTS=<accounts>**

**For mixed mode:**

**setup /ACTION=REBUILDDATABASE /QUIET /INSTANCENAME=<instance   
name> /SQLSYSADMINACCOUNTS=<accounts> /SAPWD=password**

1. **Once rebuilding is completed then run the server in single user mode**
2. **Restore master database by replacing existing one.**
3. **Restart the server in multi user mode.**

**ISSUE 12: Troubleshooting Tempdb moving issue**

**One of an instance master database data file was corrupted and I was unable to start the server. How to troubleshoot this scenario?**

**Possible Scenarios:**

**\* If the master files are corrupted or damaged, instance cannot be started. We have to rebuild master database then by running the server in single user mode we have to restore latest backup to get previous settings.**

**Steps:**

1. **Check the error log for exact reason.**
2. **Rebuild master database as follows by running setup from**

**C:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\Release**

**For windows authentication:**

**setup /ACTION=REBUILDDATABASE /QUIET /INSTANCENAME=<instance   
name> /SQLSYSADMINACCOUNTS=<accounts>**

**For mixed mode:**

**setup /ACTION=REBUILDDATABASE /QUIET /INSTANCENAME=<instance   
name> /SQLSYSADMINACCOUNTS=<accounts> /SAPWD=password**

1. **Once rebuilding is completed then run the server in single user mode**
2. **Restore master database by replacing existing one.**
3. **Restart the server in multi user mode.**

**ISSUE 13: Troubleshooting Insufficient Disk Space in tempdb**

**Running out of disk space in tempdb can cause significant disruptions in the SQL Server production environment and can prohibit applications that are running from completing operations.**

**Possible Scenarios:**

|  |  |
| --- | --- |
| **Error** | **Is raised when** |
| **1101 or 1105** | **Any session must allocate space in tempdb.** |
| **3959** | **The version store is full. This error usually appears after a 1105 or 1101 error in the log.** |
| **3967** | **The version store is forced to shrink because tempdb is full.** |
| **3958 or 3966** | **A transaction cannot find the required version record in tempdb.** |

**ISSUE 14: FINDOUT DATABASE SIZE**

**SELECT alt.filename [File Name] ,alt.name [Database Name] ,alt.size \* 8.0 / 1024.0 AS [Originalsize (MB)] ,files.size \* 8.0 / 1024.0 AS [Currentsize (MB)] FROM master.dbo.sysaltfiles alt INNER JOIN dbo.sysfiles files ON alt.fileid = files.fileid WHERE alt.size <> files.size**

**The above query allows us to find the current status of our databases and their corresponding final file growths. Use further filter conditions to fetch the databases that are of interest to you.**

**ISSUE 15: IMPORTANT SCRIPTS IN SQLDBA**

1. **To display version, level, edition etc**

**select SERVERPROPERTY('productversion'),**

**SERVERPROPERTY('productlevel'),**

**SERVERPROPERTY('edition'),**

**SERVERPROPERTY('isclustered')**

1. **To display execution plans present in procedure cache**

**SELECT cp.objtype AS PlanType,**

**OBJECT\_NAME(st.objectid,st.dbid) AS ObjectName,**

**cp.refcounts AS ReferenceCounts,cp.usecounts AS UseCounts,**

**st.text AS SQLBatch,qp.query\_plan AS QueryPlan**

**FROM sys.dm\_exec\_cached\_plans AS cp**

**CROSS APPLY sys.dm\_exec\_query\_plan(cp.plan\_handle) AS qp**

**CROSS APPLY sys.dm\_exec\_sql\_text(cp.plan\_handle) AS st;**

1. **To display instance names with T-SQL**

**EXECUTE xp\_regread**

**@rootkey ='HKEY\_LOCAL\_MACHINE',**

**@key ='SOFTWARE\Microsoft\Microsoft SQL Server',**

**@value\_name ='InstalledInstances'**

1. **Backups Information**

**Note: To check recent backups of all databases**

**SELECT T1.Name AS DatabaseName,**

**COALESCE(CONVERT(VARCHAR(12), MAX(T2.backup\_finish\_date), 101),'Not Yet Taken') AS LastBackUpTaken**

**FROM master.sys.databases T1 LEFT OUTER JOIN**

**msdb.dbo.backupset T2**

**ON T2.database\_name = T1.name**

**GROUP BY T1.Name**

**ORDER BY T1.Name**

1. **To get complete backups information of a particular database**

**SELECT s.database\_name,**

**m.physical\_device\_name,**

**cast(s.backup\_size/1000000 as varchar(14))+' '+'MB' as bkSize,**

**CAST (DATEDIFF(second,s.backup\_start\_date , s.backup\_finish\_date)AS VARCHAR(4))+' '+'Seconds' TimeTaken,**

**s.backup\_start\_date,**

**CASE s.[type]**

**WHEN 'D' THEN 'Full'**

**WHEN 'I' THEN 'Differential'**

**WHEN 'L' THEN 'Transaction Log'**

**END as BackupType,**

**s.server\_name, s.recovery\_model**

**FROM msdb.dbo.backupset s**

**inner join msdb.dbo.backupmediafamily m**

**ON s.media\_set\_id = m.media\_set\_id**

**WHERE s.database\_name = 'AdventureWorks'**

**ORDER BY database\_name, backup\_start\_date, backup\_finish\_date**

1. **DMV to monitor locks**

**SELECT t1.resource\_type,t1.resource\_database\_id,t1.resource\_associated\_entity\_id,t­1.request\_mode,t1.request\_session\_id,**

**t2.blocking\_session\_id,o1.name 'object name',o1.type\_desc 'object**

**descr',p1.partition\_id 'partition id',p1.rows 'partition/page rows',**

**a1.type\_desc 'index descr',a1.container\_id 'index/page container\_id' FROM**

**sys.dm\_tran\_locks as t1**

**INNER JOIN sys.dm\_os\_waiting\_tasks as t2**

**ON t1.lock\_owner\_address = t2.resource\_address**

**LEFT OUTER JOIN sys.objects o1 on o1.object\_id =**

**t1.resource\_associated\_entity\_id**

**LEFT OUTER JOIN sys.partitions p1 on p1.hobt\_id =**

**t1.resource\_associated\_entity\_id**

**LEFT OUTER JOIN sys.allocation\_units a1 on a1.allocation\_unit\_id =**

**t1.resource\_associated\_entity\_id**

1. **Displaying expensive queries**

**SELECT TOP 10 SUBSTRING(qt.TEXT, (qs.statement\_start\_offset/2)+1,**

**((CASE qs.statement\_end\_offset**

**WHEN -1 THEN DATALENGTH(qt.TEXT)**

**ELSE qs.statement\_end\_offset**

**END - qs.statement\_start\_offset)/2)+1),**

**qs.execution\_count,**

**qs.total\_logical\_reads, qs.last\_logical\_reads,**

**qs.total\_logical\_writes, qs.last\_logical\_writes,**

**qs.total\_worker\_time,**

**qs.last\_worker\_time,**

**qs.total\_elapsed\_time/1000000 total\_elapsed\_time\_in\_S,**

**qs.last\_elapsed\_time/1000000 last\_elapsed\_time\_in\_S,**

**qs.last\_execution\_time,**

**qp.query\_plan**

**FROM sys.dm\_exec\_query\_stats qs**

**CROSS APPLY sys.dm\_exec\_sql\_text(qs.sql\_handle) qt**

**CROSS APPLY sys.dm\_exec\_query\_plan(qs.plan\_handle) qp**

**ORDER BY qs.total\_logical\_reads DESC -- logical reads**

**-- ORDER BY qs.total\_logical\_writes DESC -- logical writes**

**-- ORDER BY qs.total\_worker\_time DESC -- CPU time**

1. **Expensive queries we can check using**

**Activity Monitor**

1. **Using server dashboard report**

**Performance - Top Queries by Total CPU**

**IO**

1. **To view cached plans**

**SELECT cp.objtype AS PlanType,**

**OBJECT\_NAME(st.objectid,st.dbid) AS ObjectName,**

**cp.refcounts AS ReferenceCounts,**

**cp.usecounts AS UseCounts,**

**st.text AS SQLBatch,**

**qp.query\_plan AS QueryPlan**

**FROM sys.dm\_exec\_cached\_plans AS cp**

**CROSS APPLY sys.dm\_exec\_query\_plan(cp.plan\_handle) AS qp**

**CROSS APPLY sys.dm\_exec\_sql\_text(cp.plan\_handle) AS st;**

**GO**

**Note: To view the query in a particular session we can use**

**dbcc inputbuffer(spid)**

**But this command displays only first 256 chars of a query/batch.**

1. **To view complete query we can use the following DMF from SS 2005**

**sys.dm\_exec\_sql\_text**

1. **To view no of catched plans in procedure cache we can use**

**dbcc proccache**

1. **To remove execution plans from procedure cache**

**dbcc freeproccache**

1. **Recovery model, log reuse wait description, log file size, log usage size and compatibility level for all databases on instance**

**SELECT db.[name] AS [Database Name],**

**db.recovery\_model\_desc AS [Recovery Model],**

**db.log\_reuse\_wait\_desc AS [Log Reuse Wait Description],**

**ls.cntr\_value AS [Log Size (KB)], lu.cntr\_value AS [Log Used (KB)],**

**CAST(CAST(lu.cntr\_value AS FLOAT) / CAST(ls.cntr\_value AS FLOAT)AS DECIMAL(18,2)) \* 100 AS [Log Used %],**

**db.[compatibility\_level] AS [DB Compatibility Level], db.page\_verify\_option\_desc AS [Page Verify Option]**

**FROM sys.databases AS db**

**INNER JOIN sys.dm\_os\_performance\_counters AS lu**

**ON db.name = lu.instance\_name**

**INNER JOIN sys.dm\_os\_performance\_counters AS ls**

**ON db.name = ls.instance\_name**

**WHERE lu.counter\_name LIKE 'Log File(s) Used Size (KB)%'**

**AND ls.counter\_name LIKE 'Log File(s) Size (KB)%';**

1. **Backup and Restoration**

**Note: To check recent backups of all databases**

**SELECT**

**T1.Name AS DatabaseName,**

**COALESCE(CONVERT(VARCHAR(12), MAX(T2.backup\_finish\_date), 101),'Not Yet Taken') AS LastBackUpTaken**

**FROM master.sys.databases T1 LEFT OUTER JOIN**

**msdb.dbo.backupset T2**

**ON T2.database\_name = T1.name**

**GROUP BY T1.Name**

**ORDER BY T1.Name**

1. **To get complete backups information of a particular database**

**SELECT s.database\_name,**

**m.physical\_device\_name,**

**cast(s.backup\_size/1000000 as varchar(14))+' '+'MB' as bkSize,**

**CAST (DATEDIFF(second,s.backup\_start\_date , s.backup\_finish\_date)AS VARCHAR(4))+' '+'Seconds' TimeTaken,**

**s.backup\_start\_date,**

**CASE s.[type]**

**WHEN 'D' THEN 'Full'**

**WHEN 'I' THEN 'Differential'**

**WHEN 'L' THEN 'Transaction Log'**

**END as BackupType,**

**s.server\_name, s.recovery\_model**

**FROM msdb.dbo.backupset s**

**inner join msdb.dbo.backupmediafamily m**

**ON s.media\_set\_id = m.media\_set\_id**

**WHERE s.database\_name = 'AdventureWorks'**

**ORDER BY database\_name, backup\_start\_date, backup\_finish\_date**

**ISSUE 16: DBCC & SP’S**

**DBCC:**

**1.DBCC CHECKALLOC  
DBCC CHECKALLOC checks page usage and allocation in the database. Use this command if allocation errors are found for the database. If you run DBCC CHECKDB, you do not need to run DBCC CHECKALLOC, as DBCC CHECKDB includes the same checks (and more) that DBCC CHECKALLOC performs.  
  
  
2.DBCC CHECKCATALOG  
This command checks for consistency in and between system tables. This command is not executed within the DBCC CHECKDB command, so running this command weekly is recommended.  
  
3.DBCC CHECKCONSTRAINTS  
DBCC CHECKCONSTRAINTS alerts you to any CHECK or constraint violations.  
Use it if you suspect that there are rows in your tables that do not meet the constraint or CHECK constraint rules.  
  
4.DBCC CHECKDB  
A very important DBCC command, DBCC CHECKDB should run on your SQL Server instance on at least a weekly basis. Although each release of SQL Server reduces occurrences of integrity or allocation errors, they still do happen. DBCC CHECKDB includes the same checks as DBCC CHECKALLOC and DBCC CHECKTABLE. DBCC CHECKDB can be rough on concurrency, so be sure to run it at off-peak times.  
  
5.DBCC CHECKTABLE  
DBCC CHECKTABLE is almost identical to DBCC CHECKDB, except that it is performed at the table level, not the database level. DBCC CHECKTABLE verifies index and data page links, index sort order, page pointers, index pointers, data page integrity, and page offsets. DBCC CHECKTABLE uses schema locks by default, but can use the TABLOCK option to acquire a shared table lock. CHECKTABLE also performs object checking using parallelism by default (if on a multi-CPU system).  
  
6.DBCC CHECKFILEGROUP  
DBCC CHECKFILEGROUP works just like DBCC CHECKDB, only DBCC CHECKFILEGROUP checks the specified filegroup for allocation and structural issues. If you have a very large database (this term is relative, and higher end systems may be more apt at performing well with multi-GB or TB systems ) , running DBCC CHECKDB may be time-prohibitive.  
If your database is divided into user defined filegroups, DBCC CHECKFILEGROUP will allow you to isolate your integrity checks, as well as stagger them over time.  
  
7.DBCC CHECKIDENT  
DBCC CHECKIDENT returns the current identity value for the specified table, and allows you to correct the identity value if necessary.  
  
8.DBCC DBREINDEX  
If your database allows modifications and has indexes, you should rebuild your indexes on a regular basis. The frequency of your index rebuilds depends on the level of database activity, and how quickly your database and indexes become fragmented. DBCC DBREINDEX allows you to rebuild one or all indexes for a table. Like DBCC CHECKDB, DBCC CHECKTABLE, DBCC CHECKALLOC, running DBREINDEX during peak activity times can significantly reduce concurrency.  
  
9.DBCC INDEXDEFRAG  
Microsoft introduced the excellent DBCC INDEXDEFRAG statement beginning with SQL Server 2000. This DBCC command, unlike DBCC DBREINDEX, does not hold long term locks on indexes. Use DBCC INDEXDEFRAG for indexes that are not very fragmented, otherwise the time this operation takes will be far longer then running DBCC DBREINDEX. In spite of it's ability to run during peak periods, DBCC INDEXDEFRAG has had limited effectiveness compared to DBCC DBREINDEX (or drop/create index).  
  
10.DBCC INPUTBUFFER  
The DBCC INPUTBUFFER command is used to view the last statement sent by the client connection to SQL Server. When calling this DBCC command, you designate the SPID to examine. (SPID is the process ID, which you can get from viewing current activity in Enterprise Manager or executing sp\_who. )  
  
11.DBCC OPENTRAN  
DBCC OPENTRAN is a Transact-SQL command that is used to view the oldest running transaction for the selected database. The DBCC command is very useful for troubleshooting orphaned connections (connections still open on the database but disconnected from the application or client), and identification of transactions missing a COMMIT or ROLLBACK. This command also returns the oldest distributed and undistributed replicated transactions, if any exist within the database. If there are no active transactions, no data will be returned. If you are having issues with your transaction log not truncating inactive portions, DBCC OPENTRAN can show if an open transaction may be causing it.  
  
12.DBCC PROCCACHE  
You may not use this too frequently, however it is an interesting DBCC command to execute periodically, particularly when you suspect you have memory issues. DBCC PROCCACHE provides information about the size and usage of the SQL Server procedure cache.  
  
13.DBCC SHOWCONTIG  
The DBCC SHOWCONTIG command reveals the level of fragmentation for a specific table and its indices. This DBCC command is critical to determining if your table or index has internal or external fragmentation. Internal fragmentation concerns how full an 8K page is.  
When a page is underutilized, more I/O operations may be necessary to fulfill a query request than if the page was full, or almost full.  
External fragmentation concerns how contiguous the extents are. There are eight 8K pages per extent, making each extent 64K. Several extents can make up the data of a table or index. If the extents are not physically close to each other, and are not in order, performance could diminish.  
  
14.DBCC SHRINKDATABASE  
DBCC SHRINKDATABASE shrinks the data and log files in your database.  
Avoid executing this command during busy periods in production, as it has a negative impact on I/O and user concurrency. Also remember that you cannot shrink a database past the target percentage specified, shrink smaller than the model database, shrink a file past the original file creation size, or shrink a file size used in an ALTER DATABASE statement.  
  
15.DBCC SHRINKFILE  
DBCC SHRINKFILE allows you to shrink the size of individual data and log files. (Use sp\_helpfile to gather database file ids and sizes).  
  
16. DBCC TRACEOFF, TRACEON, TRACESTATUS  
Trace flags are used within SQL Server to temporarily enable or disable specific SQL Server instance characteristics. Traces are enabled using the DBCC TRACEON command, and disabled using DBCC TRACEOFF. DBCC TRACESTATUS is used to displays the status of trace flags. You'll most often see TRACEON used in conjunction with deadlock logging (providing more verbose error information).  
  
17.DBCC USEROPTIONS  
Execute DBCC USEROPTIONS to see what user options are in effect for your specific user connection. This can be helpful if you are trying to determine if you current user options are inconsistent with the database options.**

**18. DBCC SQLPERF(LOGSPACE) - To check the current size of log(.LDF) files of all the databases.**

**(in case of disk space issue or on log file autogrowth error)**

**19.DBCC OPENTRAN - To check the active transaction(s) of the current database.**

**20. DBCC ERRORLOG: If you rarely restart SQL Server service, resulting server log gets very large and takes a long time to**

**load and view. You can truncate (essentially create a new log) the Current Server log by this.**

**You can accomplish the same thing using this stored procedure: sp\_cycle\_errorlog.**

**21. DBCC DROPCLEANBUFFERS: To remove all the data from SQL Server's data cache (buffer) between performance tests to ensure fair testing. Fyi, this command only removes clean buffers, not dirty buffers.**

**So, before running the DBCC DROPCLEANBUFFERS command, you may first want to run the CHECKPOINT command.**

**Running CHECKPOINT will write all dirty buffers to disk. So, when you run DBCC DROPCLEANBUFFERS, you can be assured that all data buffers are cleaned out, not just the clean ones.**

**22. DBCC updatestaistics**

**GENERAL HELP PROCEDURES:**

|  |  |
| --- | --- |
| [**sp\_\_depends**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__depends) | **Better version of sp\_depends** |
| [**sp\_\_help**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__help) | **Better sp\_help** |
| [**sp\_\_helpdb**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helpdb) | **Database Information** |
| [**sp\_\_helpdevice**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helpdevice) | **Break down database devices into a nice report** |
| [**sp\_\_helpgroup**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helpgroup) | **List groups in database by access level** |
| [**sp\_\_helpindex**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helpindex) | **Shows indexes by table** |
| [**sp\_\_helpsegment**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helpsegment) | **Segment Information** |
| [**sp\_\_helprotect**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helprotect) | **Simple Protection Info for the database** |
| [**sp\_\_helptext**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helptext) | **Show comments with line splits ok** |
| [**sp\_\_helpuser**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helpuser) | **Lists users in current database by group (includes aliases)** |
| [**sp\_\_lock**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__lock) | **Lock information** |
| [**sp\_\_syntax**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__syntax) | **Works on any procedure to give you syntax** |
| [**sp\_\_who**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__who) | **sp\_who that fits on a page** |

**SYSTEM ADMINISTRATOR PROCEDURES:**

|  |  |
| --- | --- |
| [**sp\_\_block**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__block) | **Blocking processes.** |
| [**sp\_\_dbspace**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__dbspace) | **Summary of current database space information.** |
| [**sp\_\_dumpdevice**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__dumpdevice) | **Listing of Dump devices** |
| [**sp\_\_diskdevice**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__diskdevice) | **Listing of Disk devices** |
| [**sp\_\_helpdbdev**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helpdbdev) | **Show how Databases use Devices** |
| [**sp\_\_helplogin**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helplogin) | **Show logins and remote logins to server** |
| [**sp\_\_helpmirror**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helpmirror) | **Shows mirror information, discover broken mirrors** |
| [**sp\_\_segment**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__segment) | **Segment Information** |
| [**sp\_\_server**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__server) | **Server summary report (very useful)** |
| [**sp\_\_stat**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__stat) | **Give basic server performance information (loops)** |
| [**sp\_\_vdevno**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__vdevno) | **Who's who in the device world** |

**DBA PROCEDURES:**

|  |  |
| --- | --- |
| [**sp\_\_badindex**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__badindex) | **list badly formed indexes (allow nulls) or those needing statistics** |
| [**sp\_\_collist**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__collist) | **list all columns in database** |
| [**sp\_\_find\_missing\_index**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__find_missing_index) | **Finds keys that do not have associated index** |
| [**sp\_\_flowchart**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__flowchart) | **Makes a flowchart of procedure nesting** |
| [**sp\_\_groupprotect**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__groupprotect) | **Permission info by group** |
| [**sp\_\_indexspace**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__indexspace) | **Space used by indexes in database** |
| [**sp\_\_id**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__id) | **Gives information on who you are and which db you are in** |
| [**sp\_\_noindex**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__noindex) | **list of tables without indexes.** |
| [**sp\_\_helpcolumn**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helpcolumn) | **show columns for given table** |
| [**sp\_\_helpdefault**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helpdefault) | **list defaults (part of objectlist)** |
| [**sp\_\_helpobject**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helpobject) | **list objects** |
| [**sp\_\_helpproc**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helpproc) | **list procs (part of objectlist)** |
| [**sp\_\_helprule**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helprule) | **list rules (part of objectlist)** |
| [**sp\_\_helptable**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helptable) | **list tables (part of objectlist)** |
| [**sp\_\_helptrigger**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helptrigger) | **list triggers (part of objectlist)** |
| [**sp\_\_helpview**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__helpview) | **list views (part of objectlist)** |
| [**sp\_\_objprotect**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__objprotect) | **Permission info by object** |
| [**sp\_\_read\_write**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__read_write) | **list tables by # procs that read, # that write, # that do both** |
| [**sp\_\_trigger**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__trigger) | **Useful synopsis report of current database trigger schema** |
| [**sp\_\_whodo**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__whodo) | **sp\_\_who - filtered for only active processes** |

**AUDIT PROCEDURES:**

|  |  |
| --- | --- |
| [**sp\_\_auditsecurity**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__auditsecurity) | **Security Audit On Server** |
| [**sp\_\_auditdb**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__auditdb) | **Audit Current Database For Potential Problems** |
| [**sp\_\_checkkey**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__checkkey) | **Generate script for referential integrity problems (uses key info from sp\_foreignkey)** |

**REVERSE ENGINEERING PROCEDURES:**

|  |  |
| --- | --- |
| [**sp\_\_revalias**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__revalias) | **get alias generation script for current database** |
| [**sp\_\_revdb**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__revdb) | **get database generation script for server** |
| [**sp\_\_revdevice**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__revdevice) | **get device generation script for server** |
| [**sp\_\_revgroup**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__revgroup) | **get group generation script for current database** |
| [**sp\_\_revindex**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__revindex) | **get index generation script for current database** |
| [**sp\_\_revlogin**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__revlogin) | **get login generation script for server** |
| [**sp\_\_revmirror**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__revmirror) | **get mirror generation script for current database** |
| [**sp\_\_revsegment**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__revsegment) | **get segment generation script for current database** |
| [**sp\_\_revtable**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__revtable) | **get table generation script for current database** |
| [**sp\_\_revuser**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__revuser) | **get user generation script for current database** |

**OTHER PROCEDURES:**

|  |  |
| --- | --- |
| [**sp\_\_bcp**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__bcp) | **Create unix script to bcp in/out database** |
| [**sp\_\_date**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__date) | **Who can remember all the date styles?** |
| [**sp\_\_iostat**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__iostat) | **Loops n times showing active processes only** |
| [**sp\_\_grep**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__grep) | **Search for patern** |
| [**sp\_\_isactive**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__isactive) | **Shows info about a single active process** |
| [**sp\_\_ls**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__ls) | **Lists specific objects** |
| [**sp\_\_quickstats**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__quickstats) | **Quick dump of server summary information** |
| [**sp\_\_whoactive**](http://www.edbarlow.com/gem/procs_only/readme.htm#sp__whoactive) | **Show info about who is active** |

**ISSUE 17: QUERY ARCHITECTURE**

**Performance Tuning, Monitoring and Troubleshooting  
  
\* As part of performance tuning we have to analyze and work with  
    \* Physical I/O and Logical I/O  
    \* CPU usage  
    \* Memory usage  
    \* Database Design  
    \* Application's db programming methods  
  
Query Architecture  
\* Once the query is submitted to Database Engine for first time it performs the following          tasks.  
    \* Parsing        (Compiling)  
    \* Resolving    (Verifying syntax, table, col names etc)  
    \* Optimizing    (Generating execution plan)  
    \* Executing    (Executing query)  
\* For next time if the query was executed with same case and same no of characters i.e            with no extra spaces then the query is executed by taking existing plan.  
\* To display cached plans  
       SELECT cp.objtype AS PlanType,  
       OBJECT\_NAME(st.objectid,st.dbid) AS ObjectName,  
       cp.refcounts AS Reference Counts, cp. usecounts AS UseCounts,  
       st.text AS SQLBatch,qp.query\_plan AS QueryPlan  
    FROM sys.dm\_exec\_cached\_plans AS cp  
    CROSS APPLY sys.dm\_exec\_query\_plan(cp.plan\_handle) AS qp  
    CROSS APPLY sys.dm\_exec\_sql\_text(cp.plan\_handle) AS st;  
    GO  
\* To remove plans from cache memory  
    DBCC FREEPROCCACHE  
  
Execution Plan  
\* Step by step process followed by SS to execute a query is called execution plan.  
\* It is prepared by Query Optimizer using STATISTICS.  
\* Query optimizer prepares execution plan and stores in Procedurec Cache.  
\* Execution plans are different for   
    \* Different case statements  
    \* Different size statements (spaces.)  
\* To view graphical execution plan  
    \* select the query --> press ctrl+M/L  
\* To view xml execution plan  
    \* set showplan\_xml on/off  
    \* Execute the query  
\* To view text based execution plan  
    \* set showplan\_text on/off  
    \* Execute the query.  
  
Statistics  
\* Consists of meta data of the table or index.  
\* If statistics are out of date, query optimizer may prepare poor plan.  
\* We have to update statistics weekly with maintenance plan.  
USE master  
GO  
-- Enable Auto Update of Statistics  
ALTER DATABASE AdventureWorks SET AUTO\_UPDATE\_STATISTICS ON;  
GO  
-- Update Statistics for whole database  
EXEC sp\_updatestats  
GO  
-- Get List of All the Statistics of Employee table  
sp\_helpstats 'Human Resources .Employee', 'ALL'  
GO  
-- Get List of statistics of  AK\_Employee\_NationalIDNumber index  
DBCC SHOW\_STATISTICS ("HumanResources.Employee",AK\_Employee\_NationalIDNumber)  
-- Update Statistics for single table  
UPDATE STATISTICS Human Resources. Employee  
GO  
-- Update Statistics for single index on single table  
UPDATE STATISTICS Human Resources.Employee AK\_Employee\_NationalIDNumber  
GO  
  
Index  
\* It is another database objects which can be used  
    \* To reduce searching process  
    \* To enforce uniqueness  
\* By default SS search for the rows by following the process called table scan.  
\* If the table consists of huge data then table scan provides less performance.  
\* Index is created in tree-like structure which consists of root, node and leaf level.  
\* At leaf level, index pages are present by default.  
\* We can place max 250 indexes per table.  
\* Indexes are automatically placed if we place  
    \* Primary key (clustered)  
    \* Unique      (unique non clustered index)  
\* We can place indexes as follows  
    create [unique][clustered/nonclustered] index <indexName> on                      <tname>/<viewName>(col1,col2,....)  
    [include(.....)]  
  
Types   
-------  
    \* Clustered  
    \* NonClustered  
  
1. Clustered Index-----------------------  
\* It physically sorts the rows in the table.  
\* A table can have only ONE clustered index.  
\* Both data and index pages are merged and stored at third level (Leaf level).  
\* We can place on columns which are used to search a range of rows,  
Ex:  
Create table prods(pid int,pname varchar(40), qty int)  
insert prods values(4,'Books',50),(2,'Pens',400)  
  
select \*  from prods (run the query by pressing ctrl+L)  
  
create clustered index pid\_indx on prods(pid)  
  
select \* from prods -- check the rows are sorted in asc order to pid  
  
  
FAQ:- Difference between P.K and Clustered Index?  
\* Primary key enforce uniqueness and allows to eshtablish relationship. But by default       clustered index cannot.  
  
select \* from prods where pid=2 -- press ctrl+L to check execution plan  
insert prods values(3,'Pencils',500) -- Check this row is inserted as second record.  
  
Note:     A table without clustered index is called HEAP where the rows and pages of the         table are not  present in any order.  
  
NonClustered Index-----------------------  
\* It cannot sort the rows physically.  
\* We can place max 249 nonclustered indexes on table.  
\* Both data and index pages are stored seperately.  
\* It locates rows either from heap (Table scan) or from clustered index.  
\* Always we have to place first clustered index then nonclustered.  
\* If the table is heap the index page consists of  
    IndexKeyColvalues        rowreference  
\* If the table consists of clustered index then index page consists of  
    IndexKeyColValues        Clusteredindexkeycolvalues  
\* Nonclustered indexes are rebuilded when  
    \* Clustered index is created/droped/modified  
  
Ex: Create nonclustered index on pname column of prods table.  
    create index indx1 on prods(pname)  
    select \* from prods where pname='Books' -- check execution plan  
  
\* To disp indexes present on a table  
    sp\_helpindex <tname>  
\* To drop index  
    drop index prods.pid\_indx  
\* To disp space used by the index  
    sp\_spaceused prods  
  
Using Included Columns in NonClustered Index--------------------------------------------------------  
\* We can maintain regularly used columns in nonclustered index so that   no need that      SQL Server should take data from heap or clustered   index.  
\* If the no of rows are more it provides better performance.  
  
Ex:  
--step1  
USE AdventureWorks  
GO  
CREATE NONCLUSTERED INDEX IX\_Address\_PostalCode  
ON Person.Address (PostalCode)  
INCLUDE (AddressLine1, AddressLine2, City, StateProvinceID)  
GO   
  
--step2  
SELECT AddressLine1, AddressLine2, City, StateProvinceID, PostalCode  
FROM Person.Address  
WHERE PostalCode BETWEEN '98000'  
AND '99999';  
GO  
  
Index Management  
  
Fill Factor------------  
\* Percentage of space used in leaf level index pages.  
\* By default it is 100%.  
\* To reduce page splits when the data is manipulated in the base table we can set proper       FillFactor.  
\* It allows online index processing   
    \* While the index rebuilding process is going on users can work with the table.  
  
Page Split------------  
\* Due to regular changes in the table if the index pages are full to allocate memory for          the index key   columns SS takes remaining rows into new page. This process is called          Page split.  
\* Page split increases size of index and the index pages order changes.  
\* This situation where unused free space is available and the index pages are not in the        order of key column values is called fragmentation.  
\* To find fragmentation level we can use  
    dbcc showcontig  
    or  
    We can use sys.dm\_db\_index\_physical\_stats DMF as follows  
  
    SELECT a.index\_id, name, avg\_fragmentation\_in\_percent  
    FROM sys.dm\_db\_index\_physical\_stats   
    (DB\_ID('AdventureWorks'),  
    OBJECT\_ID('Production.Product'), NULL, NULL, NULL)   
    AS a JOIN sys.indexes AS b   
    ON a.object\_id = b.object\_id   
    AND a.index\_id =b.index\_id;  
  
\* To control fragmentation we can either reorganize the index or rebuild the index.  
  
1. Reorganizing Index    \* It is the process of arranging the index pages according to the order of index key column values.  
    \* If the fragmentation level is more than 5 to 8% and less than 28to 30% then we can reorganize the indexes.  
    \* It cannot reduce the index size as well as statistics are not updated.  
syn:  
    ALTER INDEX <indexName>/<All> on <tname> REORGANIZE  
2. Index Rebuilding    \* It is the process of deleting and creating fresh index.  
    \* It reduces the size of index and updates statistics  
    \* If the fragmentation level is more than 30% then we can rebuild indexes.  
syn:  
    ALTER INDEX <indexName>/<ALL> on <tname> REBUILD  
Note:  
    If we have mentioned ONLINE INDEX PROCESSING option then rebuilding takes space in TEMPDB.  
    To check consistancy of a database we can use DBCC CHECKDB('dbName') it disp if any corrupted pages are present, use space in tempdb.  
  
Transactions and Locks  
----------------------------  
\* A transaction is single unit of work which may consists of one or more commands.  
\* Transactions works with ACID properties  
    \* Automicity  
    \* Consistancy  
    \* Isolation  
    \* Durability  
\* SQL Server supports 2 types of transactions   
    \* Implicit  
    \* Explicit  
\* By default SS supports implicit transaction where for every insert, update and delete 3            records are stored in T.Log file  
    Begin tran  
        insert/update/delete  
    commit tran  
\* To implement business logic i.e. according to the required if we want to commit or                rollback the changes we can use explicit transactions.  
    Begin Tran      
        ----  
    commit/rollback tran  
\* Any transaction which consists of manipulations places locks on the tables.  
\* By default when we make a db as current db automatically Shared Lock is placed.  
\* While working with insert,update,delete by default SS places Exclusive lock.  
\* Type of locks placed on objects depends on isolation levels.  
  
Isolation Levels  
-------------------  
\* It is a transaction property.  
\* Types of locks placed by SS on the resource depends on isolation levels.  
\* SS supports 5 isolation levels  
    \* Read Committed (Default)  
    \* Read Uncommitted  
    \* Repeatable Reads  
    \* Snapshot  
    \* Serializable  
\* To check the isolation level  
    dbcc useroptions  
\* To set the isolation level  
    SET TRANSACTION ISOLATION LEVEL <requiredisolationlevel>  
\* To handle the concurrency related problems SS places locks  
\* SS supports 2 types of concurrencies  
    \* Optimistic Concurrency  
        \* Uses Shared Locks  
        \* More concurrency  
    \* Pessimistic Concurrency  
        \* Uses Exclusive Locks  
        \* Low concurrency  
  
Ex:    Open new query window  
    --user1  
    use Test  
    go  
    begin tran  
        update emp set sal=5000  
    Take new query -->   
    --user2  
    use Test  
    go  
    select \* from emp    (--query runs continuesly till user1 session releases lock)  
      
    Take new query  
    --user3   
    set transaction isolation level read uncommitted  
    select \* from emp  
  
    --Take new query  
    sp\_lock            -- To view locks information  
    or  
    select \* from sys.dm\_tran\_locks  
  
    --check blocking using  
    sp\_who/sp\_who2  
  
    -- To check locks placed by a particular session  
    sp\_lock <spid>  
    sp\_lock 56**

**ISSUE 18: PROFILER EVENTS**

**What Data to Collect:**

**Profiler allows you to specify which events you want to capture and which data columns from those event to capture. In addition, you can use filters to reduce the incoming data to only what you need for this specific analysis.**

**Events to Capture:**

**         Stored Procedures--RPC:Completed**

**         TSQL--SQL:BatchCompleted**

**You may be surprised that only two different events need to be captured: one for capturing stored procedures and one for capturing all other Transact-SQL queries.**

**Data Columns to Capture:**

**         Duration (data needs to be grouped by duration)**

**         Event Class**

**         DatabaseID (If you have more than one database on the server)**

**         TextData**

**         CPU**

**         Writes**

**         Reads**

**         StartTime  (optional)**

**         EndTime  (optional)**

**         ApplicationName (optional)**

**         NTUserName (optional)**

**         LoginName  (optional)**

**         SPID**

**The data you want to actually capture and view includes some that are very important to you, especially duration and TextData; and some that are not so important, but can be useful, such as ApplicationName or NTUserName.**

**Filters to Use:**

**         Duration > 5000 milliseconds (5 seconds)**

**         Don’t collect system events**

**         Collect data by individual database ID, not all databases at once**

**         Others, as appropriate**

**Filters are used to reduce the amount of data collected, and the more filters you use, the more data you can filter out that is not important. Generally, I use three filters, but others can be used, as appropriate to your situation. And of these, the most important is duration. I only want to collect information on those that have enough duration to be of importance to me, as we have already discussed.**

**Collecting the Data:**

**Depending on the filters you used, and the amount of time you run Profiler to collect the data, and how busy your server is, you may collect a lot of rows of data. While you have several choices, I suggest you configure Profiler to save the data to a file on you local computer (not on the server you are Profiling), and not set a maximum file size. Instead, let the file grow as big as it needs to grow. You may want to watch the growth of this file, in case it gets out of hand. In most cases, if you have used appropriate filters, the size should stay manageable. I recommend using one large file because it is easier to identify long running queries if you do.**

**As mentioned before, collect your trace file during a typical production period, over a period of 3-4 hours or so. As the data is being collected, it will be sorted for you by duration, with the longest running queries appearing at the bottom of the Profiler window. It can be interesting to watch this window for awhile while you are collecting data. If you like, you can configure Profiler to automatically turn itself off at the appropriate time, or you can do this manually.**

**Once the time is up and the trace stopped, the Profiler trace is now stored in the memory of the local computer, and on disk. Now you are ready to identify those long running queries.**

**Analyzing the Data:**

**Guess what, you have already identified all queries that ran during the trace collection that exceed your specified duration, whatever it was. So if you selected a duration of 5 seconds, you will only see those queries that took longer than five seconds to run. By definition, all the queries you have captured need to be tuned. "What! But over 500 queries were captured! That's a lot of work!" It is not as bad as you think. In most cases, many of the queries you have captured are duplicate queries. In other words, you have probably captured the same query over and over again in your trace. So those 500 captured queries may only be 10, or 50, or even 100 distinct queries. On the other hand, there may be only a handful of queries captured (if you are lucky).**

**Whether you have just a handful, or a lot of slow running queries, you next job is to determine which are the most critical for you to analyze and tune first. This is where you need to set priorities, as you probably don't have enough time to analyze them all.**

**To prioritize the long running queries, you will probably want to first focus on those that run the longest. But as you do this, keep in mind how often each query is run.**

**For example, if you know that a particular query is for a report that only runs once a month (and you happened to have captured it when it was running), and this query took 60 second to run, it probably is not as high as a priority to tune as a query that takes 10 seconds to run, but runs 10 times a minute. In other words, you need to balance the length of how long a query takes to run, to how often it runs. With this in mind, you need to identify and prioritize those queries that take the most physical SQL Server resources to run. Once you have done this, then you are ready to analyze and tune them.**

**Traces that you want to replay must contain a minimum set of events and data columns. If the trace doesn't contain the necessary elements, you won't be able to replay the trace. The required elements are in addition to any other elements that you want to monitor or display with traces. Events that you must capture in order to allow a trace to be replayed and analyzed correctly are**

**                     Connect**

**                     CursorExecute (required only when replaying server-side cursors)**

**                     CursorOpen (required only when replaying server-side cursors)**

**                     CursorPrepare (required only when replaying server-side cursors)**

**                     Disconnect**

**                     Exec Prepared SQL (required only when replaying server-side prepared SQL statements)**

**                     ExistingConnection**

**                     Prepare SQL (required only when replaying server-side prepared SQL statements)**

**                     RPC:OutputParameter**

**                     RPC:Starting**

**                     SQL:BatchStarting**

**Data columns that you must capture to allow a trace to be replayed and analyzed correctly are:**

**                     Application Name**

**                     Binary Data**

**                     Connection ID or SPID**

**                     Database ID**

**                     Event Class**

**                     Event SubClass**

**                     Host Name**

**                     Integer Data**

**                     Server Name**

**                     SQL User Name**

**                     Start Time**

**                     Text**

**ISSUE 19: DMV**

**Sys.dm\_os\_wait\_stats is the DMV that contains wait statistics, which are aggregated across all session ids since the last restart of SQL Server or since the last time that the wait statistics were reset manually using DBCC SQLPERF ('sys.dm\_os\_wait\_stats', CLEAR). Resetting wait statistics can be helpful before running a test or workload.**

**Anytime a session\_id waits for a resource, the session\_id is moved to the waiter list along with an associated wait type. The DMV sys.dm\_os\_waiting\_tasks shows the waiter list at a given moment in time. Waits for all session\_ids are aggregated in sys.dm\_os\_wait\_stats.**

**The stored procedures track\_waitstats\_2005 and get\_waitstats\_2005 can be used to measure the wait statistics for a given workload.**

**What are DMVs  
Dynamic Management Views are views and functions introduced in sql server 2005 for monitoring and tuning sql server performance Two types of dynamic management views:**

1. **Server-scoped DMV: Stored in Master Database**
2. **Database-scoped DMV: Specific to each database**

**Permission to Execute DMV [Security]**

**To query a server scoped DMV, the database user must have SELECT privilege on VIEW SERVER STATE and for database scoped DMV, the user must have SELECT privilege on VIEW DATABASE STATE.**

* **GRANT VIEW SERVER STATE to <Login>**
* **GRANT VIEW DATABASE STATE to <User>**

**If you want to deny a user permission to query certain DMVs, you can use the DENY command to restrict access to a specific DMV.**

**All the DMVs exits in SYS schema and their names start with DM\_. So when you need to query a DMV, you should prefix the view name with SYS. As an example, if you need to see the total physical memory of the SQL Server machine;**

**SELECT**

**(Physical\_memory\_in\_bytes/1024.0)/1024.0 AS Physical\_memory\_in\_Mb**

**FROM sys.dm\_os\_sys\_info**

**how many DMV/DMF are there in SQL Server, to get that information (see Pinal's** [**post**](http://blog.sqlauthority.com/2010/05/15/sql-server-list-all-the-dmv-and-dmf-on-server/)**)**

**SELECT name, type, type\_desc FROM sys.system\_objects WHERE name LIKE 'dm\_%' ORDER BY name**

**or**

**SELECT name, type, type\_desc FROM sys.system\_objects WHERE name LIKE 'dm[\_]%' ORDER BY name**

**Frequently used**

1. **SQL Server related [Hardware Resources] DMV**
2. **Database related DMV**
3. **Index related DMV**
4. **Execution related DMV**

### 1. SQL Server Related DMV

**This section details the DMVs associated with SQL Server system. SQL DMV is responsible to manage server level resources specific to a SQL Server instance.**

**This section covers DMVs related to OS, Disk and Memory.**

#### a. sys.dm\_os\_sys\_info

**This view returns the information about the SQL Server machine, available resources and the resource consumption.**

**This view returns information like the following:**

1. **CPU Count: Number of logical CPUs in the server**
2. **Hyperthread-ratio: Ratio of logical and physical CPUs**
3. **Physical\_memory\_in\_bytes: Amount of physical memory available**
4. **Virtual\_memory\_in\_bytes: Amount of virtual memory available**
5. **Bpool\_commited: Committed physical memory in buffer pool**
6. **OS\_Priority\_class: Priority class for SQL Server process**
7. **Max\_workers\_thread: Maximum number of workers which can be created**

#### b. sys.dm\_os\_hosts

**This view returns all the hosts registered with SQL Server 2005. This view also provides the resources used by each host.**

1. **Name: Name of the host registered**
2. **Type: Type of hosted component [SQL Native Interface/OLE DB/MSDART]**
3. **Active\_tasks\_count: Number active tasks host placed**
4. **Active\_ios\_count: I/O requests from host waiting**

#### c. sys.dm\_os\_schedulers

**Sys.dm\_os\_schedulers view will help you identify if there is any CPU bottleneck in the SQL Server machine. The number of runnable tasks is generally a nonzero value; a nonzero value indicates that tasks have to wait for their time slice to run. If the runnable task counts show high values, then there is a symptom of CPU bottleneck.**

**http://www.codeproject.com/images/minus.gifCollapse |** [**Copy Code**](http://www.codeproject.com/KB/database/Dynamic_Management_Views.aspx)

**SELECT**

**scheduler\_id,current\_tasks\_count,runnable\_tasks\_count**

**FROM sys.dm\_os\_schedulers**

**WHERE scheduler\_id < 255**

**The above query will list all the available schedulers in the SQL Server machine and the number of runnable tasks for each scheduler.**

#### d. sys.dm\_io\_pending\_io\_requests

**This dynamic view will return the I/O requests pending in SQL Server side. It gives you information like:**

1. **Io\_type: Type of pending I/O request**
2. **Io\_pending: Indicates whether the I/O request is pending or has been completed by Windows**
3. **Scheduler\_address: Scheduler on which this I/O request was issued**

#### e. sys.dm\_io\_virtual\_file\_stats

**This view returns I/O statistics for data and log files [MDF and LDF file]. This view is one of the commonly used views and will help you to identify I/O file level. This will return information like:**

1. **Sample\_ms: Number of milliseconds since the instance of SQL Server has started**
2. **Num\_of\_reads: Number of reads issued on the file**
3. **Num\_of\_bytes\_read: Total number of bytes read on this file**
4. **Io\_stall\_read\_ms: Total time, in milliseconds, that the users waited for reads issued on the file**
5. **Num\_of\_writes: Number of writes made on this file**
6. **Num\_of\_bytes\_written: Total number of bytes written to the file**
7. **Io\_stall\_write\_ms: Total time, in milliseconds, that users waited for writes to be completed on the file**
8. **Io\_stall: Total time, in milliseconds, that users waited for I/O to be completed**
9. **Size\_on\_disk\_bytes: Number of bytes used on the disk for this file**

#### f. sys.dm\_os\_memory\_clerks

**This DMV will help how much memory SQL Server has allocated through AWE.**

**http://www.codeproject.com/images/minus.gifCollapse |** [**Copy Code**](http://www.codeproject.com/KB/database/Dynamic_Management_Views.aspx)

**SELECT**

**SUM(awe\_allocated\_kb) / 1024 as [AWE allocated, Mb]**

**FROM sys.dm\_os\_memory\_clerks**

**The same DMV can be used to get the memory consumption by internal components of SQL Server 2005.**

**http://www.codeproject.com/images/minus.gifCollapse |** [**Copy Code**](http://www.codeproject.com/KB/database/Dynamic_Management_Views.aspx)

**SELECT TOP 10 type,**

**SUM(single\_pages\_kb) as [SPA Mem, Kb]**

**FROM sys.dm\_os\_memory\_clerks**

**GROUP BY type**

**ORDER BY SUM(single\_pages\_kb) DESC**

#### g. sys.dm\_os\_ring\_buffers

**This DMV uses RING\_BUFFER\_RESOURCE\_MONITOR and gives information from resource monitor notifications to identify memory state changes. Internally, SQL Server has a framework that monitors different memory pressures. When the memory state changes, the resource monitor task generates a notification. This notification is used internally by the components to adjust their memory usage according to the memory state.**

**http://www.codeproject.com/images/minus.gifCollapse |** [**Copy Code**](http://www.codeproject.com/KB/database/Dynamic_Management_Views.aspx)

**SELECT**

**Record FROM sys.dm\_os\_ring\_buffers**

**WHERE ring\_buffer\_type = 'RING\_BUFFER\_RESOURCE\_MONITOR'**

**The output of the above query will be in XML format. The output will help you in detecting any low memory notification.**

**RING\_BUFFER\_OOM: Ring buffer oom contains records indicating server out-of-memory conditions.**

**http://www.codeproject.com/images/minus.gifCollapse |** [**Copy Code**](http://www.codeproject.com/KB/database/Dynamic_Management_Views.aspx)

**SELECT**

**record FROM sys.dm\_os\_ring\_buffers**

**WHERE ring\_buffer\_type = 'RING\_BUFFER\_OOM'**

### 2. Database Related DMV

**This section details the DMVs associated with SQL Server Databases. These DMVs will help to identify database space usages, partition usages, session information usages, etc...**

#### a. sys.dm\_db\_file\_space\_usage

**This DMV provides the space usage information of TEMPDB database.**

#### b. sys.dm\_db\_session\_space\_usage

**This DMV provides the number of pages allocated and de-allocated by each session for the database**

#### c. sys.dm\_db\_partition\_stats

**This DMV provides page and row-count information for every partition in the current database.**

**The below query shows all counts for all partitions of all indexes and heaps in the MSDB database:**

**http://www.codeproject.com/images/minus.gifCollapse |** [**Copy Code**](http://www.codeproject.com/KB/database/Dynamic_Management_Views.aspx)

**USE MSDB;**

**GO**

**SELECT \* FROM sys.dm\_db\_partition\_stats;**

**The following query shows all counts for all partitions of Backup set table and its indexes**

**http://www.codeproject.com/images/minus.gifCollapse |** [**Copy Code**](http://www.codeproject.com/KB/database/Dynamic_Management_Views.aspx)

**USE MSDB**

**GO**

**SELECT \* FROM sys.dm\_db\_partition\_stats**

**WHERE object\_id = OBJECT\_ID('backupset');**

#### d. sys.dm\_os\_performance\_counters

**Returns the SQL Server / Database related counters maintained by the server.**

**The below sample query uses the dm\_os\_performance\_counters DMV to get the Log file usage for all databases in KB.**

**http://www.codeproject.com/images/minus.gifCollapse |** [**Copy Code**](http://www.codeproject.com/KB/database/Dynamic_Management_Views.aspx)

**SELECT instance\_name**

**,cntr\_value 'Log File(s) Used Size (KB)'**

**FROM sys.dm\_os\_performance\_counters**

**WHERE counter\_name = 'Log File(s) Used Size (KB)'**

### 3. INDEX Related DMV

**This section details the DMVs associated with SQL Server Databases. These DMVs will help to identify database space usages, Partition usages, Session information usages, etc.**

#### a. sys.dm\_db\_index\_usage\_stats

**This DMV is used to get useful information about the index usage for all objects in all databases. This also shows the amount of seeks and scan for each index.**

**http://www.codeproject.com/images/minus.gifCollapse |** [**Copy Code**](http://www.codeproject.com/KB/database/Dynamic_Management_Views.aspx)

**SELECT object\_id, index\_id, user\_seeks, user\_scans, user\_lookups**

**FROM sys.dm\_db\_index\_usage\_stats**

**ORDER BY object\_id, index\_id**

**All indexes which have not been used so far in as database can be identified using the below Query:**

**http://www.codeproject.com/images/minus.gifCollapse |** [**Copy Code**](http://www.codeproject.com/KB/database/Dynamic_Management_Views.aspx)

**SELECT object\_name(i.object\_id),**

**i.name,**

**s.user\_updates,**

**s.user\_seeks,**

**s.user\_scans,**

**s.user\_lookups**

**from sys.indexes i**

**left join sys.dm\_db\_index\_usage\_stats s**

**on s.object\_id = i.object\_id and**

**i.index\_id = s.index\_id and s.database\_id = 5**

**where objectproperty(i.object\_id, 'IsIndexable') = 1 and**

**s.index\_id is null or**

**(s.user\_updates > 0 and s.user\_seeks = 0**

**and s.user\_scans = 0 and s.user\_lookups = 0)**

**order by object\_name(i.object\_id)**

**Replace the Database\_id with the database you are looking at.**

### 4. Execution Related DMV

**Execution related DMVs will provide information regarding sessions, connections, and various requests which are coming into the SQL Server.**

#### a. sys.dm\_exec\_sessions

**This DMV will give information on each session connected to SQL Server. This DMV is similar to running sp\_who2 or querying Master..sysprocesses table.**

**http://www.codeproject.com/images/minus.gifCollapse |** [**Copy Code**](http://www.codeproject.com/KB/database/Dynamic_Management_Views.aspx)

**SELECT**

**session\_id,login\_name,**

**last\_request\_end\_time,cpu\_time**

**FROM sys.dm\_exec\_sessions**

**WHERE session\_id >= 51 – All user Sessions**

#### b. sys.dm\_exec\_connections

**This DMV shows all the connection to SQL Server. The below query uses sys.dm\_exec\_connections DMV to get connection information. This view returns one row for each user connection (Sessionid > =51).**

**http://www.codeproject.com/images/minus.gifCollapse |** [**Copy Code**](http://www.codeproject.com/KB/database/Dynamic_Management_Views.aspx)

**SELECT**

**connection\_id,**

**session\_id,client\_net\_address,**

**auth\_scheme**

**FROM sys.dm\_exec\_connections**

#### c. sys.dm\_exec\_requests

**This DMV will give details on what each connection is actually performing in SQL Server.**

**http://www.codeproject.com/images/minus.gifCollapse |** [**Copy Code**](http://www.codeproject.com/KB/database/Dynamic_Management_Views.aspx)

**SELECT**

**session\_id,status,**

**command,sql\_handle,database\_id**

**FROM sys.dm\_exec\_requests**

**WHERE session\_id >= 51**

#### d. sys.dm\_exec\_sql\_text

**This dynamic management function returns the text of a SQL statement given a SQL handle.**

**http://www.codeproject.com/images/minus.gifCollapse |** [**Copy Code**](http://www.codeproject.com/KB/database/Dynamic_Management_Views.aspx)

**SELECT**

**st.text**

**FROM**

**sys.dm\_exec\_requests r**

**CROSS APPLY**

**sys.dm\_exec\_sql\_text(sql\_handle) AS st**

**WHERE r.session\_id = 51**

## Conclusion

**Dynamic Management views (DMV) and Dynamic Management Functions (DMF) in SQL Server 2005 give a transparent view of what is going on inside various areas of SQL Server. By using them, we will be able to query the system for information about its current state in a much more effective manner and provide solutions much faster. DMVs can be used to performance tune and for troubleshooting server and queries. This article has shown an overview of what they are and how we can use them.**

**Diagnosing problems in SQL Server 2000 has always been a point of concern from both developers and DBA's. More often than not we would have had a need to use undocumented and DBCC commands which are sometimes very difficult to understand too. SQL Server 2005 on the contrary is like a open book, no need to use bit based operations and undocumented column values. Welcome the introduction of Dynamic Management Views and Fuctions a.k.a DMV's and DMF's.**

**From the basic definition these dynamic management views and functions very much replace all the DBCC command outputs and the pseudo table outputs. Hence it is far more easier to detect the health of SQL Server using these views and functions. All these are defined in the sys schema. There are two scope for these views and function: *Server scoped* and *Database scoped*. Incidentally unlike in SQL Server 2000 now to view these objects the user needs to have SELECT permissions and VIEW SERVER/DATABASE STATE permissions. Now that I mentioned about SQL Server 2000, try this yourself, create a readonly user in a database and select the sysobjects table and check the results returned in SQL Server 2000 and SQL Server 2005.**

**There are multiple categories in which these views and functions have been organized. The below table shows the split:**

|  |  |
| --- | --- |
| **Categories** | **Count** |
| **dm\_broker\*** | **4** |
| **dm\_clr\*** | **4** |
| **dm\_db\*** | **12** |
| **dm\_exec\*** | **14** |
| **dm\_fts\*** | **5** |
| **dm\_io\*** | **4** |
| **dm\_os\*** | **27** |
| **dm\_qn\*** | **1** |
| **dm\_repl\*** | **4** |
| **dm\_tran\*** | **10** |

**So we have 85 of these views and function. To give a further split, 76 of these are views and 9 of them are functions. So these information can be queried from the system\_objects system catalog table. A typical query I used was:**

**select \* from sys.system\_objects**

**Where name like 'dm\_%' Order by 1**

**Each of these views and functions have different parameters or output columns and in the next couple of queries we will try to find out how to get these values.**

**-- Getting the column details of the DMV's**

**Select o.name, c.name, t.name, c.column\_id, c.max\_length, c.precision, c.scale**

**FROM sys.system\_columns c**

**INNER JOIN sys.system\_objects o**

**ON c.object\_id = o.object\_id**

**INNER JOIN sys.types t**

**ON c.user\_type\_id = t.user\_type\_id**

**Where o.name = '*dm\_os\_loaded\_modules*'**

**order by 1**

**In the above query we query we get the output columns for the DMV (dm\_os\_loaded\_modules) using the system objects. In the above query we get details like name of the output column, datatype and other length specific values. Even though this will not get us the values for the table valued functions. We will have to tweak the above query for DMF's.**

**-- Getting the column details of the DMF's**

**Select o.name, t.name, p.\***

**FROM sys.system\_parameters p**

**INNER JOIN sys.system\_objects o**

**ON p.object\_id = o.object\_id**

**INNER JOIN sys.types t**

**ON p.user\_type\_id = t.user\_type\_id**

**Where o.name = '*dm\_exec\_sql\_text*'**

**order by 1**

**In the above query we try to get the parameters for the DMF (dm\_exec\_sql\_text) using the systtem\_parameters system catalog. So the output would show the above DMF has a parameter @handle. So if we queried this function for the sql text for a given query in the cache. The handle can be got from dm\_exec\_query\_stats or other related views.**

# 20. Fixing Orphaned Users:

Orphaned user User1 in the SURESHDB database.

When we run **sp\_change\_users\_login** with the **REPORT** option, we can see that an orphaned user.

EXEC sp\_change\_users\_login 'REPORT'   
   
UserName UserSID   
-------------------------------------------   
User1 0xA5B5548F3DC81D4693E769631629CE1D

To fix this orphaned user all we have to do is run **sp\_change\_users\_login** with the **UPDATE\_ONE** action and tell SQL Server the name of orphaned user and the name of the appropriate login.

EXEC sp\_change\_users\_login 'UPDATE\_ONE','User1','User1'