

```

=====
--
-- SQL Source File -- Created with SAPIEN Technologies PrimalScript 4.1
--
-- NAME: Useful_SCOM2007_SQL_Queries.sql
--
-- AUTHOR: Jeremy D. Pavleck , Jeremy@Pavleck.Net
--         Pavleck.net - http://www.pavleck.net
-- PERMALINK: http://www.pavleck.net/scom-useful-sql-queries
-- DATE : 4/17/2008
-- LAST UPDATE: 01/25/2012
--
-- COMMENT: Some useful SQL queries you can use in SCOM2007.
-- Taken verbatim from Kevin Holman's OpsMgr Blog, and re-written to be SQL friendly.
--
-- URL: http://blogs.technet.com/kevinholman/archive/2007/10/18/useful-operations-manager-2007-sql-queries.aspx
--       http://myitforum.com/cs2/blogs/smoss
--
-- UPDATES:
-- 01/20/2011 - Added agents responding/not responding query
-- 04/14/2011 - Added additional queries
-- 06/22/2011 - Reformatted slightly
-- 12/14/2011 - Added Remotely Manageable queries
=====

-- Find all managed servers that are down and not pingable
-- Works with SCOM 2007 & SCOM 2007 R2
SELECT bme.DisplayName, s.LastModified
FROM state AS s, BaseManagedEntity as bme
WHERE s.basemanagedentityid = bme.basemanagedentityid AND s.monitorid
IN (SELECT MonitorId FROM Monitor WHERE MonitorName = 'Microsoft.SystemCenter.HealthService.ComputerDown')
AND s.Healthstate = '3'
ORDER BY s.Lastmodified DESC

-- Operational Database Version
-- Works with SCOM 2007 & SCOM 2007 R2
select DBVersion from __MOMManagementGroupInfo__

-- Find a computer name from it's Health Service ID (guid from agent proxy alerts)
select id, path, fullname, displayname from ManagedEntityGenericView where ID = '<GUID>'

-- ALERTS SECTION --
-----

-- Most common alerts, by alert count
SELECT AlertStringName, AlertStringDescription, AlertParams, Name, SUM(1) AS AlertCount, SUM(RepeatCount+1) AS AlertCountWith
RepeatCount
FROM Alertview WITH (NOLOCK)
WHERE ResolutionState = (0|255)
GROUP BY AlertStringName, AlertStringDescription, AlertParams, Name
ORDER BY AlertCount DESC

-- TOP 10 common alerts
SELECT Top(10) AlertStringName, AlertStringDescription, AlertParams, Name, SUM(1) AS AlertCount, SUM(RepeatCount+1) AS AlertC
ountWithRepeatCount
FROM Alertview WITH (NOLOCK)
WHERE ResolutionState = (0|255)
GROUP BY AlertStringName, AlertStringDescription, AlertParams, Name
ORDER BY AlertCount DESC

-- Most common alerts, by repeat count
SELECT AlertStringName, AlertStringDescription, AlertParams, Name, SUM(1) AS AlertCount, SUM(RepeatCount+1) AS AlertCountWith
RepeatCount
FROM Alertview WITH (NOLOCK)
WHERE ResolutionState = (0|255)
GROUP BY AlertStringName, AlertStringDescription, AlertParams, Name
ORDER BY AlertCountWithRepeatCount DESC

-- Number of console alerts per day
SELECT CONVERT(VARCHAR(20), TimeAdded, 102) AS DayAdded, COUNT(*) AS NumAlertsPerDay
FROM Alert WITH (NOLOCK)
WHERE TimeRaised is not NULL
GROUP BY CONVERT(VARCHAR(20), TimeAdded, 102)
ORDER BY DayAdded DESC

-- Number of console alerts per day by resolution state
SELECT
CASE WHEN (GROUPING(CONVERT(VARCHAR(20), TimeAdded, 102)) = 1) THEN 'All Days' ELSE CONVERT(VARCHAR(20), TimeAdded, 102) END AS
[Date],
CASE WHEN (GROUPING(ResolutionState) = 1) THEN 'All Resolution States' ELSE CAST(ResolutionState AS VARCHAR(5)) END AS [Resolu
tionState],
COUNT(*) AS NumAlerts
FROM Alert WITH (NOLOCK)
WHERE TimeRaised is not NULL
GROUP BY CONVERT(VARCHAR(20), TimeAdded, 102), ResolutionState WITH ROLLUP
ORDER BY DATE DESC

-- Manage computer, alert description and alert severity
-- Scott Moss
-- 10/27/2008
-- SCOM 2007 SP1 OperationsManager DB Query
-- Get listing of MonitoringObject, Alert description and Severity
-- http://myitforum.com/cs2/blogs/smoss

select
MonitoringObjectDisplayName,
AlertStringName,

```

```

AlertStringDescription,
'Severity' = Case
When AlertView.Severity = 0 Then 'INFORMATIONAL'
When AlertView.Severity = 1 Then 'WARNING'
When AlertView.Severity = 2 Then 'CRITICAL'
Else 'Undetermined'
End
From AlertView

-- Number of alerts per day
SELECT CONVERT(VARCHAR(20), TimeAdded, 101) AS DayAdded, COUNT(*) AS NumAlertsPerDay
FROM Alert WITH (NOLOCK)
GROUP BY CONVERT(VARCHAR(20), TimeAdded, 101)
ORDER BY DayAdded DESC

-- Number of alerts per day by resolution state
SELECT
CASE WHEN (GROUPING(CONVERT(VARCHAR(20), TimeAdded, 101)) = 1) THEN 'All Days' ELSE CONVERT(VARCHAR(20), TimeAdded, 101) END AS
[Date],
CASE WHEN (GROUPING(ResolutionState) = 1) THEN 'All Resolution States' ELSE CAST(ResolutionState AS VARCHAR(5)) END AS [Resolu
tionState],
COUNT(*) AS NumAlerts
FROM Alert WITH (NOLOCK)
GROUP BY CONVERT(VARCHAR(20), TimeAdded, 101), ResolutionState WITH ROLLUP
ORDER BY DATE DESC

-- EVENTS SECTION --
-----

-- Most common events by day by count
SELECT CASE WHEN (GROUPING(CONVERT(VARCHAR(20), TimeAdded, 101)) = 1)
THEN 'All Days'
ELSE CONVERT(VARCHAR(20), TimeAdded, 101) END AS DayAdded,
COUNT(*) AS NumEventsPerDay
FROM EventAllView
GROUP BY CONVERT(VARCHAR(20), TimeAdded, 101) WITH ROLLUP
ORDER BY DayAdded DESC

-- Most common events by event number
SELECT Number, COUNT(*) AS "Number of Events"
FROM EventView
GROUP BY Number
ORDER BY "Number of Events" DESC

-- Most common events by event number and event publishername
SELECT top 20 Number as EventID, COUNT(*) AS TotalEvents, Publishername as EventSource
FROM EventAllView eav with (nolock)
GROUP BY Number, Publishername
ORDER BY TotalEvents DESC

-- Most common events, grouped by identical event number, publishername, and event parameters: (This shows use completely red
undant events with identical data - but might be different than the above queries... you need to see both data outputs to
fully tune)
SELECT top 100 Number as EventID, COUNT(*) AS TotalEvents, Publishername as EventSource, EventParameters
FROM EventAllView with (NOLOCK)
GROUP BY Number, Publishername, EventParameters
ORDER BY TotalEvents DESC

-- Computers generating the most events: (This shows us which computers create the most event traffic and use the most databa
se space)
SELECT top 20 LoggingComputer as ComputerName, COUNT(*) AS TotalEvents
FROM EventallView with (NOLOCK)
GROUP BY LoggingComputer
ORDER BY TotalEvents DESC

-- Computers generating the most events, by event number: (This shows the noisiest computers, group by unique event numbers)
SELECT top 20 LoggingComputer as ComputerName, COUNT(*) AS TotalEvents, Number as EventID
FROM EventallView with (NOLOCK)
GROUP BY LoggingComputer, Number
ORDER BY TotalEvents DESC

-- Computers generating the most events, grouped by identical event number and publishername:
SELECT top 20 LoggingComputer as ComputerName, COUNT(*) AS TotalEvents, PublisherName as EventSource, Number as EventID
FROM EventallView with (NOLOCK)
GROUP BY LoggingComputer, PublisherName, Number
ORDER BY TotalEvents DESC

-- PERFORMANCE SECTION --
-----

-- Performance Insertions per day
SELECT CASE WHEN (GROUPING(CONVERT(VARCHAR(20), TimeSampled, 101)) = 1)
THEN 'All Days' ELSE CONVERT(VARCHAR(20), TimeSampled, 101)
END AS DaySampled, COUNT(*) AS NumPerfPerDay
FROM PerformanceDataAllView
GROUP BY CONVERT(VARCHAR(20), TimeSampled, 101) WITH ROLLUP
ORDER BY DaySampled DESC

-- Most common performance insertions by perf object and counter name:
select pcv.objectname, pcv.countername, count (pcv.countername) as total from performancedataallview as pdv, performancecount
erview as pcv
where (pdv.performancesourceinternalid = pcv.performancesourceinternalid)

```

```

group by pcv.objectname, pcv.countername
order by count (pcv.countername) desc

-- Most common performance insertions by perf object name:
select pcv.objectname, count (pcv.countername) as total from performedataallview as pdv, performancecounterview as pcv
where (pdv.performancesourceinternalid = pcv.performancesourceinternalid)
group by pcv.objectname
order by count (pcv.countername) desc

-- Most common performance insertions by perf counter name:
select pcv.countername, count (pcv.countername) as total from performedataallview as pdv, performancecounterview as pcv
where (pdv.performancesourceinternalid = pcv.performancesourceinternalid)
group by pcv.countername
order by count (pcv.countername) desc

-- To view all performance insertions for a given computer:
select Path, ObjectName, CounterName, InstanceName, SampleValue, TimeSampled
from PerformanceDataAllView pdv with (NOLOCK)
inner join PerformanceCounterView pcv on pdv.performancesourceinternalid = pcv.performancesourceinternalid
inner join BaseManagedEntity bme on pcv.ManagedEntityId = bme.BaseManagedEntityId
where path = 'omterm.opsmgr.net'
order by countername, timesampled

-- To refine a the above query to pull all perf data for a given computer, object, counter, and instance:
select Path, ObjectName, CounterName, InstanceName, SampleValue, TimeSampled
from PerformanceDataAllView pdv with (NOLOCK)
inner join PerformanceCounterView pcv on pdv.performancesourceinternalid = pcv.performancesourceinternalid
inner join BaseManagedEntity bme on pcv.ManagedEntityId = bme.BaseManagedEntityId
where path = 'monitoredcomputer.css-security.com' AND
objectname = 'LogicalDisk' AND
countername = 'Free Megabytes'
order by timesampled

-- STATE SECTION --
-----

-- State changes per day:
SELECT CASE WHEN (GROUPING(CONVERT(VARCHAR(20), TimeGenerated, 101)) = 1)
THEN 'All Days' ELSE CONVERT(VARCHAR(20), TimeGenerated, 101)
END AS DayGenerated, COUNT(*) AS NumEventsPerDay
FROM StateChangeEvent WITH (NOLOCK)
GROUP BY CONVERT(VARCHAR(20), TimeGenerated, 101) WITH ROLLUP
ORDER BY DayGenerated DESC

-- MANAGEMENT PACK INFO --
-----

-- To find all installed Management Packs and their version:
SELECT MPName, MPFriendlyName, MPVersion, MPISealed
FROM ManagementPack WITH(NOLOCK)
ORDER BY MPName

-- Rules per MP:
SELECT mp.MPName, COUNT(*) As RulesPerMP
FROM Rules r
INNER JOIN ManagementPack mp ON mp.ManagementPackID = r.ManagementPackID
GROUP BY mp.MPName
ORDER BY RulesPerMP DESC

-- Rules per MP by category:
SELECT mp.MPName, r.RuleCategory, COUNT(*) As RulesPerMPPerCategory
FROM Rules r
INNER JOIN ManagementPack mp ON mp.ManagementPackID = r.ManagementPackID
GROUP BY mp.MPName, r.RuleCategory
ORDER BY RulesPerMPPerCategory DESC

-- Monitors Per MP:
SELECT mp.MPName, COUNT(*) As MonitorsPerMPPerCategory
FROM Monitor m
INNER JOIN ManagementPack mp ON mp.ManagementPackID = m.ManagementPackID
GROUP BY mp.MPName
ORDER BY COUNT(*) Desc

-- To find your Monitor by common name:
select * from Monitor
Inner join LocalizedText LT on LT.ElementName = Monitor.MonitorName
where LTValue = 'My Monitor Name'

-- To find all Rules per MP that generate an alert:
-- CHECK FOR R2
declare @mpid as varchar(50)
select @mpid= managementpackid from managementpack where
mpName='Microsoft.BizTalk.Server.2006.Monitoring'
select rl.rulename,rl.ruleid,md.modulename from rules rl, module md
where md.managementpackid = @mpid
and rl.ruleid=md.parentid
and moduleconfiguration like '%<AlertLevel>50</AlertLevel>%'

-- To find all rules per MP with a given alert severity:
declare @mpid as varchar(50)
select @mpid= managementpackid from managementpack where
mpName='Microsoft.BizTalk.Server.2006.Monitoring'
select rl.rulename,rl.ruleid,md.modulename from rules rl, module md
where md.managementpackid = @mpid

```

```
and rl.ruleid=md.parentid
and moduleconfiguration like '%<Severity>2</Severity>%'
```

```
-- Number of instances of a type: (Number of disks, computers, databases, etc that OpsMgr has discovered)
SELECT mt.ManagedTypeID, mt.TypeName, COUNT(*) AS NumEntitiesByType
FROM BaseManagedEntity bme WITH (NOLOCK)
LEFT JOIN ManagedType mt WITH (NOLOCK) ON mt.ManagedTypeID = bme.BaseManagedTypeID
WHERE bme.IsDeleted = 0
GROUP BY mt.ManagedTypeID, mt.TypeName
ORDER BY COUNT(*) DESC
```

```
-- Number of Views per Management Pack:
SELECT mp.MPName, v.ViewVisible, COUNT(*) As ViewsPerMP
FROM [Views] v
INNER JOIN ManagementPack mp ON mp.ManagementPackID = v.ManagementPackID
GROUP BY mp.MPName, v.ViewVisible
ORDER BY v.ViewVisible DESC, COUNT(*) Desc
```

```
-- Grooming:
SELECT * FROM PartitionAndGroomingSettings WITH (NOLOCK)
```

```
-- All managed computers count:
SELECT COUNT(*) AS NumManagedComps FROM (
SELECT bme2.BaseManagedEntityID
FROM BaseManagedEntity bme WITH (NOLOCK)
INNER JOIN BaseManagedEntity bme2 WITH (NOLOCK) ON bme2.BaseManagedEntityID = bme.TopLevelHostEntityID
WHERE bme2.IsDeleted = 0
AND bme2.IsDeleted = 0
AND bme2.BaseManagedTypeID = (SELECT TOP 1 ManagedTypeID FROM ManagedType WHERE TypeName = 'microsoft.windows.com
puter')
GROUP BY bme2.BaseManagedEntityID
) AS Comps
```

```
-- Classes available in the DB:
SELECT * FROM ManagedType
```

```
-- Classes available in the DB for Microsoft Windows type:
SELECT * FROM ManagedType
WHERE TypeName LIKE 'Microsoft.Windows.%'
```

```
-- Every property of every class:
SELECT * FROM MT_Computer
```

```
-- All instances of all types once discovered
SELECT * FROM BaseManagedEntity
```

```
-- To get the state of every instance of a particular monitor the following query can be run, (replace <MonitorName> with the
name of the monitor):
SELECT bme.FullName, bme.DisplayName, s.HealthState FROM state AS s, BaseManagedEntity as bme
WHERE s.basemanagedentityid = bme.basemanagedentityid
AND s.monitorid IN (SELECT MonitorId FROM Monitor WHERE MonitorName = '<MonitorName>')
```

```
-- For example, this gets the state of the Microsoft.SQLServer.2005.DBEngine.ServiceMonitor for each instance of the SQL 2005
Database Engine class.
SELECT bme.FullName, bme.DisplayName, s.HealthState
FROM state AS s, BaseManagedEntity as bme
WHERE s.basemanagedentityid = bme.basemanagedentityid
AND s.monitorid IN (SELECT MonitorId FROM Monitor WHERE MonitorName = 'Microsoft.SQLServer.2005.DBEngine.ServiceMonitor')
```

```
-- To find the overall state of any object in OpsMgr the following query should be used to return the state of the System.Ent
ityState monitor:
SELECT bme.FullName, bme.DisplayName, s.HealthState
FROM state AS s, mt_managedcomputer AS mt, BaseManagedEntity as bme
WHERE s.basemanagedentityid = bme.basemanagedentityid
AND s.monitorid IN (SELECT MonitorId FROM Monitor WHERE MonitorName = 'System.Health.EntityState')
```

```
-- Rules are stored in a table named Rules. This table has columns linking rules to classes and Management Packs.
-- To find all rules in a Management Pack use the following query and substitute in the required Management Pack name:
SELECT * FROM Rules WHERE ManagementPackID = (SELECT ManagementPackID from ManagementPack WHERE MPName = 'Microsoft.Windows.S
erver.2003')
```

```
-- To find all rules targeted at a given class use the following query and substitute in the required class name:
SELECT * FROM Rules WHERE TargetManagedEntityType = (SELECT ManagedTypeId FROM ManagedType WHERE TypeName = 'Microsoft.Window
s.Computer')
```

```
-- The Alert table contains all alerts currently open in OpsMgr. This includes resolved alerts until they are groomed out of
the database. To get all alerts across all instances of a given monitor use the following query and substitute in the req
uired monitor name:
```

```
SELECT * FROM Alert WHERE ProblemID IN (SELECT MonitorId FROM Monitor WHERE MonitorName = 'Microsoft.SQLServer.2005.DBEngine.
ServiceMonitor')
```

```
-- To retrieve all alerts for all instances of a specific class use the following query and substitute in the required table
name, in this example MT_DBEngine is used to look for SQL alerts:
SELECT * FROM Alert WHERE BaseManagedEntityID IN (SELECT BaseManagedEntityID from MT_DBEngine)
```

```
-- To determine which table is currently being written to for event and performance data use the following query:
SELECT * FROM PartitionTables WHERE IsCurrent = 1
```

```
-- To retrieve events generated by a specific rule use the following query and substitute in the required rule ID:
SELECT * FROM Event_00 WHERE RuleId = (SELECT RuleId FROM Rules WHERE RuleName = 'Microsoft.Windows.Server.2003.OperatingSyst
```

```

em.CleanShutdown.Collection ')

-- To retrieve all events generated by rules in a specific Management Pack the following query can be used where the Manage
nt Pack name is substituted with the required value:
SELECT * FROM EventAllView
WHERE RuleID IN (SELECT RuleId FROM Rules WHERE ManagementPackId =
(SELECT ManagementPackId FROM ManagementPack WHERE MPName = 'Microsoft.Windows.Server.2003'))

-- To retrieve all performance data for a given rule in a readable format use the following query:
SELECT pc.ObjectName, pc.CounterName, ps.PerfmonInstanceName, pd.SampleValue, pd.TimeSampled
FROM PerformanceDataAllView AS pd, PerformanceCounter AS pc, PerformanceSource AS ps
WHERE pd.PerformanceSourceInternalId IN (SELECT PerformanceSourceInternalId FROM PerformanceSource
WHERE RuleId = (SELECT RuleId FROM Rules WHERE RuleName = 'Microsoft.Windows.Server.2003.LogicalDisk.FreeSpace.Collection'))

-- Information on existing User Roles:
SELECT UserRoleName, IsSystem from userrole

-- Grooming in the DataWarehouse:
-- Grooming no longer uses SQL agent jobs. Grooming is handled by scheduled stored procedures, that run much more frequently
, which provides less impact than in the previous version.
-- Default grooming for the DW for each dataset, to examine Data Warehouse grooming settings:
-- VERSION FOR 2007 RTM ONLY:
USE OperationsManagerDW
SELECT AggregationIntervalDurationMinutes, BuildAggregationStoredProcedureName, GroomStoredProcedureName, MaxDataAgeDays, Gro
omingIntervalMinutes FROM StandardDatasetAggregation

-- VERSION FOR

-- The first row is the interval in minutes.
-- NULL is raw data, 60 is hourly, and 1440 is daily.
-- The second and third row shows what data it is
-- MaxDataAgeDays has the retention period in days - this is the field to update if the administrator wants to lower the days
of retention.
-- RAW alert - 400 days
-- RAW event - 100 days
-- RAW perf - 10 days (hourly and daily perf = 400 days)
-- RAW state - 180 days (hourly and daily state = 400 days)

-- AEM Queries (Data Warehouse):

-- Default query to return all RAW AEM data:
select * from [CM].[vCMAemRaw] Rw
inner join dbo.AemComputer Computer on Computer.AemComputerRowID = Rw.AemComputerRowID
inner join dbo.AemUser Usr on Usr.AemUserRowId = Rw.AemUserRowId
inner join dbo.AemErrorGroup EGrp on Egrp.ErrorGroupRowId = Rw.ErrorGroupRowId
Inner join dbo.AemApplication App on App.ApplicationRowId = Egrp.ApplicationRowId

-- Count the raw crashes per day:
SELECT CONVERT(char(10), DateTime, 101) AS "Crash Date (by Day)", COUNT(*) AS "Number of Crashes"
FROM [CM].[vCMAemRaw]
GROUP BY CONVERT(char(10), DateTime, 101)
ORDER BY "Crash Date (by Day)" DESC

-- Count the total number of raw crashes in the DW database:
select count(*) from CM.vCMAemRaw

-- Default grooming for the DW for the AEM dataset: (Aggregated data kept for 400 days, RAW 30 days by default)
SELECT AggregationTypeID, BuildAggregationStoredProcedureName, GroomStoredProcedureName, MaxDataAgeDays, GroomingIntervalMinu
tes
FROM StandardDatasetAggregation WHERE BuildAggregationStoredProcedureName = 'AemAggregate'

-- MISCELLANEOUS SECTION --
-----

-- Simple query to display large tables, to determine what is taking up space in the database:
SELECT so.name,
8 * Sum(CASE WHEN si.indid IN (0, 1) THEN si.reserved END) AS data_kb,
Coalesce(8 * Sum(CASE WHEN si.indid NOT IN (0, 1, 255) THEN si.reserved END), 0) AS index_kb,
Coalesce(8 * Sum(CASE WHEN si.indid IN (255) THEN si.reserved END), 0) AS blob_kb
FROM dbo.sysobjects AS so JOIN dbo.sysindexes AS si ON (si.id = so.id)
WHERE 'U' = so.type GROUP BY so.name ORDER BY data_kb DESC

-- Is SQL broker enabled?
SELECT is_broker_enabled FROM sys.databases WHERE name = 'OperationsManager'

-- Determine the number of agents responding and not responding
SELECT 'Responding' as Status, COUNT(*) as TotalMachines FROM ManagedEntityGenericView INNER JOIN ManagedTypeView
ON ManagedEntityGenericView.MonitoringClassId = ManagedTypeView.Id
WHERE (ManagedEntityGenericView.IsAvailable = 'True') AND (ManagedTypeView.Name = 'Microsoft.SystemCenter.Agent')
Union
SELECT 'NotResponding' as Status, COUNT(*) as TotalMachines FROM ManagedEntityGenericView INNER JOIN ManagedTypeView
ON ManagedEntityGenericView.MonitoringClassId = ManagedTypeView.Id
WHERE (ManagedEntityGenericView.IsAvailable = 'false') AND (ManagedTypeView.Name = 'Microsoft.SystemCenter.Agent')

-- Return the size of database tables in Megabytes
select object_name(id) [Table Name],
[Table Size] = convert (varchar, dpages * 8 / 1024) + 'MB'

from sysindexes where indid in (0,1)
order by dpages desc

-- Large table query
SELECT TOP 1000

```

```

a2.name AS [tablename], (a1.reserved + ISNULL(a4.reserved,0))* 8 AS reserved,
a1.rows as row_count, a1.data * 8 AS data,
(CASE WHEN (a1.used + ISNULL(a4.used,0)) > a1.data THEN (a1.used + ISNULL(a4.used,0)) - a1.data ELSE 0 END) * 8 AS index_size
,
(CASE WHEN (a1.reserved + ISNULL(a4.reserved,0)) > a1.used THEN (a1.reserved + ISNULL(a4.reserved,0)) - a1.used ELSE 0 END) *
8 AS unused,
(row_number() over(order by (a1.reserved + ISNULL(a4.reserved,0)) desc)%2 as ll,
a3.name AS [schemaname]
FROM (SELECT ps.object_id, SUM (CASE WHEN (ps.index_id < 2) THEN row_count ELSE 0 END) AS [rows],
SUM (ps.reserved_page_count) AS reserved,
SUM (CASE WHEN (ps.index_id < 2) THEN (ps.in_row_data_page_count + ps.lob_used_page_count + ps.row_overflow_used_page_count)
ELSE (ps.lob_used_page_count + ps.row_overflow_used_page_count) END ) AS data,
SUM (ps.used_page_count) AS used
FROM sys.dm_db_partition_stats ps
GROUP BY ps.object_id) AS a1
LEFT OUTER JOIN (SELECT it.parent_id,
SUM(ps.reserved_page_count) AS reserved,
SUM(ps.used_page_count) AS used
FROM sys.dm_db_partition_stats ps
INNER JOIN sys.internal_tables it ON (it.object_id = ps.object_id)
WHERE it.internal_type IN (202,204)
GROUP BY it.parent_id) AS a4 ON (a4.parent_id = a1.object_id)
INNER JOIN sys.all_objects a2 ON ( a1.object_id = a2.object_id )
INNER JOIN sys.schemas a3 ON (a2.schema_id = a3.schema_id)
WHERE a2.type <> N'S' and a2.type <> N'IT'
```

-- Database size and used space

```

USE OperationsManager
select a.FILEID,
[FILE_SIZE_MB]=convert(decimal(12,2),round(a.size/128.000,2)),
[SPACE_USED_MB]=convert(decimal(12,2),round(fileproperty(a.name,'SpaceUsed')/128.000,2)),
[FREE_SPACE_MB]=convert(decimal(12,2),round((a.size-fileproperty(a.name,'SpaceUsed'))/128.000,2)) ,
[GROWTH_MB]=convert(decimal(12,2),round(a.growth/128.000,2)),
NAME=left(a.NAME,15),
FILENAME=left(a.FILENAME,60)
from dbo.sysfiles a
```

-- ## REMOTELY MANAGEABLE AGENTS VIEW

-- ## If an agent has been manually installed, there is less functionality from the OpsMgr console for certain things, like changing it's Management Server.

-- ## By default, they are marked as Remotely Manageable = No. Usually you can change this to "YES" via SQL.

-- ## See <http://blogs.technet.com/b/kevinholman/archive/2010/02/20/how-to-get-your-agents-back-to-remotely-manageable-in-opsmgr-2007-r2.aspx>

-- for more information about what this is and if you should change it.

-- See which SCOM agents are marked as Manually Installed

```

select bme.DisplayName from MT_HealthService mths
INNER JOIN BaseManagedEntity bme on bme.BaseManagedEntityId = mths.BaseManagedEntityId
where IsManuallyInstalled = 1
```

-- Change all agents marked as "Remotely Manageable = No" to "Remotely Manageable = Yes"

```

UPDATE MT_HealthService
SET IsManuallyInstalled=0
WHERE IsManuallyInstalled=1
```

-- Change a selected agent from "Remotely Manageable = No" to "Remotely Manageable = Yes"

```

UPDATE MT_HealthService
SET IsManuallyInstalled=0
WHERE IsManuallyInstalled=1
AND BaseManagedEntityId IN
(select BaseManagedEntityID from BaseManagedEntity
where BaseManagedTypeId = 'AB4C891F-3359-3FB6-0704-075FBFE36710'
AND DisplayName = 'agentname.domain.com')
```

-- How to identify your version of SQL server:

```

SELECT SERVERPROPERTY('productversion'), SERVERPROPERTY ('productlevel'), SERVERPROPERTY ('edition')
```

```

-- SQL 2005:
-- SQL Server 2005 RTM                2005.90.1399
-- SQL Server 2005 SP1                2005.90.2047
-- SQL Server 2005 SP1 plus 918222    2005.90.2153
-- SQL Server 2005 SP2                2005.90.3042
-- SQL Server 2008 R2 Enterprise      10.50.1600.1
-- SQL Server 2008 RTM 2007.100.1600
/*
10.5.1753.0 SQL Server 2008 R2 CU5 20 Dec 2010
10.5.1746.0 SQL Server 2008 R2 CU4 18 Oct 2010
10.5.1734.0 SQL Server 2008 R2 CU3 17 Aug 2010
10.5.1720.0 SQL Server 2008 R2 CU2 25 Jun 2010
10.5.1702.0 SQL Server 2008 R2 CU1 18 May 2010
10.5.1660.1 SQL Server 2008 R2 RTM 12 Apr 2010
SQL Server 2008
10.00.4266 SQL Server 2008 SP2 CU1 15 Nov 2010
10.00.4000 SQL Server 2008 SP2 29 Sep 2010
10.00.2804 SQL Server 2008 SP1 CU11 15 Nov 2010
10.00.2799 SQL Server 2008 SP1 CU10 21 Sep 2010
10.00.2789 SQL Server 2008 SP1 CU9 19 Jul 2010
10.00.2775 SQL Server 2008 SP1 CU8 17 May 2010
10.00.2766 SQL Server 2008 SP1 CU7 15 Mar 2010
10.00.2757 SQL Server 2008 SP1 CU6 18 Jan 2010
10.00.2746 SQL Server 2008 SP1 CU5 24 Nov 2009
10.00.2734 SQL Server 2008 SP1 CU4 22 Sep 2009
10.00.2723 SQL Server 2008 SP1 CU3 21 Jul 2009
10.00.2714 SQL Server 2008 SP1 CU2 18 May 2009
10.00.2710 SQL Server 2008 SP1 CU1 16 Apr 2009
10.00.2531 SQL Server 2008 SP1 7 Apr 2009
```

```
10.00.1835 SQL Server 2008 RTM CU10 15 Mar 2010
10.00.1828 SQL Server 2008 RTM CU9 18 Jan 2009
10.00.1823 SQL Server 2008 RTM CU8 16 Nov 2009
10.00.1818 SQL Server 2008 RTM CU7 21 Sep 2009
10.00.1812 SQL Server 2008 RTM CU6 21 Jul 2009
10.00.1806 SQL Server 2008 RTM CU5 18 May 2009
10.00.1798 SQL Server 2008 RTM CU4 17 Mar 2009
10.00.1787 SQL Server 2008 RTM CU3 19 Jan 2009
10.00.1779 SQL Server 2008 RTM CU2 17 Nov 2008
10.00.1763 SQL Server 2008 RTM CU1 22 Sep 2008
10.00.1600 SQL Server 2008 RTM 6 Aug 2008
*/
```

```
/*
THIRD PARTY TOOLS AND SUCH
*/
```

```
-- HP ProLiant Server Information
```

```
-- Scott Moss
-- 01-03-2008
-- OM 2007 Requires HP ProLiant MP v 1.0
-- Visit myitforum blog
-- http://myitforum.com/cs2/blogs/smos/default.aspx
-- Get your HP Info out of OM with out opening Operator Console
```

```
select
NetworkName_A14810C3_AA91_31C4_6864_D897E7B7BE48 as 'Server Name',
Manufacturer_A14810C3_AA91_31C4_6864_D897E7B7BE48 as 'Manufacture',
Model_A14810C3_AA91_31C4_6864_D897E7B7BE48 as 'Model',
SystemType_A14810C3_AA91_31C4_6864_D897E7B7BE48 as 'SystemType',
SerialNumber_A14810C3_AA91_31C4_6864_D897E7B7BE48 as 'Serial Number',
PhysicalMemory_A14810C3_AA91_31C4_6864_D897E7B7BE48 as 'Physical Memory',
SystemFirmware_A14810C3_AA91_31C4_6864_D897E7B7BE48 as 'System Firmware',
TotalDisk_A14810C3_AA91_31C4_6864_D897E7B7BE48 as 'Total Disk',
InsightLightsOut_52CCB151_362E_1B40_728E_1E495D3D348B as 'iLO IP',
ManagementVersion_A14810C3_AA91_31C4_6864_D897E7B7BE48 as 'Management Agent Version'
from dbo.MT_HPProLiantServer
Order By [Server Name]
```

```
-- Enjoy!
```