SSAS DMV (Dynamic Management View)

What are SSAS DMVs?

SSAS Dynamic Management Views are Analysis Services schema rowsets exposed as tables that we can query using T-SQL SELECT statement. For example, to query discover\_connections DMV, we can write:  
select \* from $system.discover\_connections

DMVs are more convenient to use than schema rowsets because:  
a)    we use SELECT statement rather than XMLA  
b)    the output is a table rather than XML

Not all DMVs are schema rowsets. The DMVs which expose cube and dimension data are not schema rowsets. For example, to get the members of product dimension we can write:  
select \* from AW.[$Dim Product].[$Dim Product] where Color = ‘Red’

Even though they are [T-SQL statements](http://msdn.microsoft.com/en-us/library/ms126272.aspx), in the Management Studio we write these SSAS DMV SELECT statements in the DMX query window, not in the SQL query window. This is because we need to connect to SSAS, rather than SQL Server relational engine. The SSAS DMVs are available in SSAS 2008 and 2008 R2; **they are not available in SSAS 2005 and 2000**.

What are they for?

SSAS DMV can be used to monitor the server resources such as:

* who are connecting to the SSAS server
* which database/cube they are using
* what MDX they are running
* how much memory and CPU is used by each command and each user
* how long they took to execute, and when did they run
* which aggregates are used/hit and which are missed
* which objects are locked and by whom
* how much memory and CPU is used by each dimension and each measure
* how much memory is used by each data cache and objects of SSAS engine
* which cubes, dimensions, partitions and measures are used most and which are rarely used (how many times they are read, how many rows returned, how many KB read)

SSAS DMVs can also be used to find out the structure of SSAS databases in the server:

* List of dimensions, hierarchies, measures, measure groups, data sources, cubes, actions and KPIs in each SSAS database
* Members of every hierarchy and level in every cube
* List of MDX functions, with their descriptions and parameters
* List of mining structures, models, and algorithms.

How do I use them?

dbschema\_tables lists all the DMVs:  
select \* from $system.dbschema\_tables

discover\_connections lists who’s connecting to the server, since when, for how long, from which PC (IP) and using what client:  
select \* from $system.discover\_connections

discover\_sessions and discover\_commands lists the MDX/command used by each user, which SSAS database/cube they are using, how much CPU and memory is used by each command:  
select \* from $system.discover\_sessions  
select \* from $system.discover\_commands

We can join discover\_connections and discover\_sessions on Connection\_ID column. And we can join discover\_sessions and discover\_commands on Session\_SPID column. To join them we can:

* Create a linked server from a SQL Server to the SSAS server, then use OpenQuery. See Midas Matelis post [here](http://www.ssas-info.com/VidasMatelisBlog/144_using-ssrs-to-report-ssas-2008-database-structure-using-dmvs#more-144).
* Use SSIS to query the 3 DMVs separately and the join them in SSIS using Merge Join transformation. See Chris Webb post [here](http://sqlserverpedia.com/blog/sql-server-bloggers/killing-sessions-automatically-with-ssis/).
* Use ADOMD to query the DMV on the SSAS server, and then join them using DataSet. See my post [here](http://dwbi1.wordpress.com/2009/12/29/ssas-dmv-join-using-dataset/).

discover\_object\_memory\_usage and discover\_object\_activity lists the memory usage and CPU time for each object i.e. cube, dimension, cache, measure, partition, etc. They also show which aggregations were hit or missed, how many times these objects were read, and how many rows were returned by them:  
select \* from $system.discover\_object\_memory\_usage  
select \* from $system.discover\_object\_activity

The following DMVs provide information on the cube structure:  
select \* from $system.mdschema\_cubes  
select \* from $system.mdschema\_dimensions  
select \* from $system.mdschema\_hierarchies  
select \* from $system.mdschema\_levels  
select \* from $system.mdschema\_measuregroups  
select \* from $system.mdschema\_measuregroup\_dimensions  
select \* from $system.mdschema\_measures  
select \* from $system.mdschema\_properties  
select \* from $system.mdschema\_members  
select \* from $system.mdschema\_sets  
select \* from $system.mdschema\_kpis  
select \* from $system.mdschema\_input\_datasources

We can use where clause and order by:  
select \* from $system.discover\_object\_memory\_usage where object\_parent\_path = ‘Global.ObjectPool’ order by object\_id desc

We can use ‘and’, ‘or’ and ‘not’. But there is no ‘like’, ‘<>’ (not equal to), ‘in’ and ‘join’:  
select \* from AW.[$Dim Product].[$Dim Product] where Color = ‘Blue’ or Color = ‘Silver’  
select \* from $system.discover\_object\_activity where not object\_rows\_returned > 0

In 2008 R2 we can use ‘<>’ (not equal to):  
select \* from $system.dbschema\_tables where table\_catalog <> ‘Adventure Works DW’ (only works in 2008 R2)

We don’t have ‘like’ so use left or right instead. Remember we can use ‘not’:  
select table\_schema from $system.dbschema\_tables where table\_catalog = ‘Adventure Works DW’ and left(table\_schema,1) = ‘$’  
select \* from … where … and not left(table\_schema,1) = ‘$’

We can use distinct:  
select distinct object\_parent\_path from $system.discover\_object\_activity

On some DMVs we need to write ‘SystemRestrictSchema’. For example if we write:  
select \* from $system.discover\_partition\_stat  
SSAS will respond with:  
The ‘CUBE\_NAME’ restriction is required but is missing from the request.  Consider using SYSTEMRISTRICTSCHEMA to provide restrictions.

This is because discover\_partition\_stat requires 4 restrictions: database\_name, cube\_name, measure\_group\_name and partition\_name. In other words, we need to specify these 4 parameters. So we need to write:  
select \* from SystemRestrictSchema($system.discover\_partition\_stat, CUBE\_NAME = ‘Adventure Works’, DATABASE\_NAME = ‘Adventure Works DW’, MEASURE\_GROUP\_NAME = ‘Internet Orders’, PARTITION\_NAME = ‘Internet\_Orders\_2004′)

Another example on restriction:  
select \* from systemrestrictschema($system.discover\_performance\_counters, PERF\_COUNTER\_NAME = ‘\MSAS 2008:Connection\Current connections’, PERF\_COUNTER\_NAME = ‘\MSAS 2008:MDX\Total Cells Calculated’)

If we have a SSAS database with 1 cube, 1 dimension and 1 measure group, when we execute: “select \* from $system.dbschema\_tables”, at the top of the list we will see 4 rows with table\_catalog = ‘DatabaseName’:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | table\_catalog | table\_schema | table\_name | table\_type |
| 1 | DB Name | $Dim1 | Dim1MG | system\_table |
| 2 | DB Name | $Dim1 | $Dim1 | table |
| 3 | DB Name | Cube1 | MG1 | system\_table |
| 4 | DB Name | Cube1 | $Dim1 | table |

Each of these 4 tables is a DMV that we can query using SELECT statement.

* No 2 is a database dimension
* No 4 is cube dimension (a database dimension that is used in a cube).
* No 3 is the measure group.
* No 1 is the ‘dimension cube’. For every dimension, SSAS create a cube. This cube is called ‘dimension cube’. This cube contains only 1 dimension with no measure. This cube can’t be accessed from the client. It is only available internally within SSAS engine.

So to query these 4 DMVs, we need to fully qualify the table name, i.e. table\_catalog.table\_schema.table\_name:

#1 select \* from [DB Name].[$Dim1].[$Dim1]  –database dimension  
#2 select \* from [DB Name].[Cube1].[$Dim1]  –cube dimension  
#3 select \* from [DB Name].[Cube1].[MG1]    –-measure group  
#4 select \* from [DB Name].[$Dim1].[Dim1MG] –dimension cube

#1 returns the members of Dim1.  
#2 also returns the members of Dim1.  
#3 returns the fact rows for measure group1.  
#4 returns nothing

DMV on Books Online

The only mention about DMV in the SQL Server Books Online is on the [Data Mining Schema Rowsets](http://msdn.microsoft.com/en-us/library/ms126272.aspx) page of the November 2009 edition:

“In SQL Server 2008, the data mining schema rowsets are also exposed as tables in the Transact-SQL language, in the $SYSTEM schema. For example, the following query on an Analysis Services instance returns a list of the schemas that are available on the current instance: SELECT \* FROM [$system].[DBSCHEMA\_TABLES]”

The Books Online explains every column of the 46 schema rowsets in SSAS: 16 XMLA, 4 OLE DB, 16 OLE DB for OLAP and 10 Data Mining. However, there are 54 DMVs in 2008. The following 12 discover\_\* DMVs are not mentioned in Nov 2009 SQL Server Books Online’s schema rowsets documentation: traces, trace\_definition\_providerinfo, trace\_columns, trace\_event\_categories, memoryusage, memorygrant, transactions, master\_key, performance\_counters, partition\_dimension\_stat, partition\_stat, dimension\_stat. But, they are mentioned in the SSAS Protocol Specification [section 3.1.4.2.3](http://msdn.microsoft.com/en-us/library/ee301466.aspx), about Discover Request Types.

There are 3 schema rowsets which are not made into DMV: mdschema\_actions, discover\_datasources and discover\_xml\_metadata. To use them we still need to use XMLA. Those of you who wonder about the numbers don’t tally (54 <> 46+12-3), that’s because there is 1 duplicate: discover\_keyword is available both as XMLA and OLE DB for OLAP.

In 2008 R2 (I’m using Nov CTP), there 3 new DMVs: discover\_storage\_tables, discover\_storate\_table\_columns, and discover\_storage\_tables\_column\_segments. Again, they are not mentioned in SQL Server Books Online’s schema rowsets documention, but they are mentioned in the SSAS Protocol Specification [section 3.1.4.2.3](http://msdn.microsoft.com/en-us/library/ee301466.aspx).