SQL Server Dynamic Management Views

COMMON LANGUAGE RUNTIME

Useful for understanding and troubleshooting CLR objects executing within SQL Server. CLR objects are cached for better performance after they are used and are not destroyed immediately. CLR objects are unloaded only when SQL Server comes under memory pressure.

sys.dm_clr_appdomains
Returns a row for each AppDomain, the unit of isolation for an application running in .NET, running on the server. SQL Server creates one AppDomain per database per owner so that all CLR objects are always executed in the same AppDomain.

sys.dm_clr_loaded_assemblies
Returns a row for each managed user assembly, i.e. managed code DLL files, loaded into the server address space.

sys.dm_clr_properties
Returns a row for each property of a CLR assembly available to the system, such as the CLR version, current state of the hosted CLR, or the CLR install directory.

sys.dm_clr_tasks
Returns a row for all currently running CLR tasks.

DATABASE

For SQL Server 2005, these DMVs are only usable for tempdb.

sys.dm_db_file_space_usage
Returns space usage information for each file in tempdb.

sys.dm_db_session_space_usage
Returns the number of pages allocated and deallocated by each session, user or internal, in tempdb.

sys.dm_db_partition_stats
Returns one row per partition in the database, showing page and row-count information for every partition in the current database, including the space used to store and manage in-row data, LOB data, and row-overflow data for all partitions in a database.

sys.dm_db_task_space_usage
Returns page allocation and deallocation activity by task for tempdb, excluding IAM pages.

To determine the amount of space used by internal and user objects in tempdb:

```
SELECT SUM(internal_object_reserved_page_count) AS internal_pages,
(SUM(internal_object_reserved_page_count)*1.0/128) AS internal_space_MB,
SUM(user_object_reserved_page_count) AS user_pages,
(SUM(user_object_reserved_page_count)*1.0/128) AS user_space_MB
FROM sys.dm_db_file_space_usage;
```

To see the total space consumed by internal objects in currently running tasks in tempdb:

```
SELECT session_id,
   SUM(internal_objects_alloc_page_count) AS task_pages,
   SUM(internal_objects_dealloc_page_count) AS task_dealloc_pages
FROM sys.dm_db_task_space_usage
GROUP BY session_id;
```

TRANSACTION

sys.dm_tran_active_snapshot_database_transactions
Returns a virtual table for each active transaction that could potentially generate access row versions. Each transaction it returns has an XSN (transaction sequence number) which is given to any transaction that accesses a version store.

sys.dm_tran_active_transactions
Returns information about transactions (such as the state of the transaction, when it began, whether it is read-only or not, and so forth) executing within the SQL Server instance.

sys.dm_tran_current_snapshot
Returns a virtual table of all active XSNs currently running when the current snapshot transaction starts. Returns no rows if the current transaction is not a snapshot transaction.

sys.dm_tran_current_transaction
Returns a single row that displays the state and snapshot sequence information of a transaction in the current session. Not useful for transactions running in isolation levels other than snapshot isolation level.

sys.dm_tran_database_transactions
Returns information about transactions at the database level, especially transaction log information such as the log sequence number (LSN), log bytes used, and transaction type.

sys.dm_tran_locks
Returns information about currently active requests to the lock manager, broken into a resource group and a request group. The request status may be active, convert, or may be waiting (wait) and includes details such as the resource on where the lock request wants a log (for the resource group) or the lock request itself (for the request group).

sys.dm_tran_session_transactions
Returns correlation information for associated transactions and sessions, especially useful for monitoring distributed transactions.

sys.dm_tran_top_version_generators
Returns a virtual table for the objects that are producing the most versions in the version store, as found in the *sys.dm_tran_version_store* system view. Use with caution. This is a very resource intensive DMV.

sys.dm_tran_transactions_snapshot
Returns invormation about active transactions when each snapshot transaction starts. Using this, you can find how many snapshot transaction are currently active and identify any data modifications that are ignored by any given snapshot transaction.

sys.dm_tran_version_store
Returns each versioned record, both its XSN and its binary version sequence number.
Because the version store can be quite large, this DMV can be very resource intensive.

To find all active distributed transactions on the SQL Server instance

```
SELECT *
FROM sys.dm_tran_active_transactions
WHERE transaction_type = 4;
```

To seeing blocked and blocking transactions on the server:

```
SELECT t.resource_type,
   t.resource_database_id,
   t.resource_associated_entity_id,
   t.request_mode,
   t.request_session_id,
   w.blocking_session_id
FROM sys.dm_tran_locks as t1
INNER JOIN sys.dm_os_waiting_tasks AS w
   ON t.lock_owner_address = w.resource_address;
```

EXECUTION & THREAD

These DMVs and functions show what activity is executing on the server. The DMV sys.dm_exec_query_transformation_stats is used internally by SQL Server 2005.

sys.dm_exec_background_job_queue
Returns a row for each query processor job that is scheduled for asynchronous, i.e. background, execution. In SQL Server 2005, this will show update statistics jobs.

sys.dm_exec_background_job_queue_stats
Returns a row that provides aggregate statistics for each query processor job submitted for background execution.

sys.dm_exec_cached_plans
Similar to *sys.syscacheobjects* in SQL Server 2000. Returns a row for each query plan that is held in procedure cache, and containing information like the cached query plans, the cached query text, the amount of memory taken by cached plans, and the reuse count of the cached plans.

sys.dm_exec_cached_plan_dependent_objects
Returns a row for each TSQL execution plan, CLR execution plan, and cursor associated with a plan.

sys.dm_exec_connections
Returns server-level information about a connection to SQL Server, such as the client network address, client TCP port, and client authorization scheme.

sys.dm_exec_cursors
Returns information about cursors that are open in one or more databases on the server.

sys.dm_exec_plan_attributes
Returns one row per plan attribute for a specific plan identified by its plan handle, such as information like the cache key values or the number of current simultaneous executions of the plan.

sys.dm_exec_query_memory_grants
Returns information about queries that have acquired memory grants or that still require a memory grant to execute. This DMV is useful for determining query timeouts, since only queries that have to wait on a memory grant will appear in this view.

sys.dm_exec_query_optimizer_info
Returns detailed statistics about the operation of the SQL Server query optimizer, such as the total number of optimizations, the elapsed time value, or sophisticated assessments like comparing the query optimization cost of the current workload to that of a tuned workload. Some counters shown by this DMV are for internal use only.

sys.dm_exec_query_plan
Returns a given query's Showplan output in XML format, as specified by the plan handle.

sys.dm_exec_query_resource_semaphores
Returns two rows, one for the regular resource semaphore and the other for the small-query resource semafor, information about the current query-resource semaphore status. When used with *sys.dm_os_memory_clerks*, this DMV provides a complete picture of memory status information about query executions and allows you to determine whether the system can access enough memory.

sys.dm_exec_query_stats
Returns one row per query statement within a cached plan, detailing aggregated performance statistics for cached query plans. Because the information is aggregated, you may sometimes get better information by rerunning this DMV.

sys.dm_exec_requests
Returns one row for each request executing within SQL Server, but does not contain information for code that executes outside of SQL Server, such as distributed queries or extended stored procedures.

sys.dm_exec_sessions
Returns one row per authenticated session, including both active users and internal tasks, running anywhere on SQL Server.

sys.dm_exec_sql_text
Similar to *fn_get_sql* in SQL Server 2000, this DMV returns the text of the SQL batch that is identified by the specified *sql_handle*.

sys.dm_exec_text_query_plan
Returns Showplan output in text format for a given TSQL batch or a statement within the batch. It's similar *sys.dm_exec_query_plan*, except that it returns data as text, is not limited in size, and may be specific to an individual statement within a batch.

sys.dm_exec_xml_handles
Returns information about one or all active handles that opened with the *sp_xml_preparedocument* system stored procedure.

To see the percentage of failed background jobs for all executed queries:

```
SELECT CASE ended_count WHEN 0
   THEN 'No jobs ended'
   ELSE CAST((failed_lock_count + failed_giveup_count + failed_other_count) /
     CAST(ended_count AS float) * 100 AS varchar(20)) END AS percent_failed
FROM sys.dm_exec_background_job_queue_stats;
```

To see the SQL text of all cached plans who have been used at least three times:

```
SELECT usecounts, cacheobjtype, objtype, text
FROM sys.dm_exec_cached_plans
   CROSS APPLY sys.dm_exec_sql_text(plan_handle)
WHERE usecounts >= 3
ORDER BY usecounts DESC;
```

To find any cursors that have been open for more than 24 hours:

```
SELECT name, cursor_id, creation_time, c.session_id, login_name
FROM sys.dm_exec_cursors(0) AS c
   JOIN sys.dm_exec_sessions AS s
   ON c.session_id = s.session_id
WHERE DATEDIFF(hh, c.creation_time, GETDATE()) > 24;
```

To find the fraction of optimized queries containing a hint:

```
SELECT (SELECT CAST (occurrence AS float)
          FROM sys.dm_exec_query_optimizer_info WHERE counter = 'hints') /
  (SELECT CAST (occurrence AS float)
     FROM sys.dm_exec_query_optimizer_info WHERE counter = 'hints')
AS [Fraction Containing Hints];
```

To see the top 5 SQL statements executing on the server by CPU time:

```
SELECT TOP 5 total_worker_time/execution_count AS AVG_CPU_Time,
   SUBSTRING(st.text, (qs.statement_start_offset/2)+1,
      ((CASE qs.statement_end_offset
         WHEN -1 THEN DATALENGTH(st.text)
         ELSE qs.statement_end_offset
         END - qs.statement_start_offset)/2) + 1) AS statement_text
FROM sys.dm_exec_query_stats AS qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) AS st
ORDER BY total_worker_time/execution_count DESC;
```

SQL SERVER OS

The following SQL Server Operating System related dynamic management views are for internal use only: *sys.dm_os_function_symbolic_name*, *sys.dm_os_ring_buffers*, *sys.dm_os_memory_allocations*, *sys.dm_os_sublatches*, and *sys.dm_os_worker_local_storage*. The remaining DMVs are user-accessible and contain a panoply of information to help you understand and manage the SQL Server Operating System (SQLOS) responsible for governing resources within the system.

sys.dm_os_buffer_descriptors
Returns information about all data pages that are in the SQL Server buffer cache, excluding free, stolen, or erroneously read pages. SQL Server 2005 tracks anything put into the buffer cache using buffer descriptors. You can easily scope the results to show one or all database and one or all of the objects within the database.

sys.dm_os_child_instances
Returns a row showing the state and pipe name of each user instance spawned from the parent server instance.

sys.dm_os_cluster_nodes
Returns a row for each node in the failover cluster instance configuration or an empty rowset if the server is not configured as in a failover cluster

sys.dm_os_hosts
Returns a row for each host (such as an OLE DB provider) currently registered in an instance of SQL Server and each resource used by these hosts.

sys.dm_os_latch_stats
Returns information about all latch waits, organized by latch class. Latches are holds placed on very lightweight and transient system resources, such as an address in memory. This DMV is useful for troubleshooting latch contention. It does not track latch usage where the latch was granted immediately or failed immediately.

sys.dm_os_loaded_modules
Returns a row for each module loaded into the server.

Sys.dm_os_memory_brokers
Returns a row for each memory broker that is currently active. Memory brokers are used by SQL Server 2008 to track memory allocations between various SQL Server internal and external components, based on current and projected usage.

sys.dm_os_memory_cache_clock_hands
Returns the status (suspended or running) of each hand for a specific cache clock. The process used to age items out of cache is called the cache clock and each clock can have one or more processes (called a hand) to sweep the cache clean.

sys.dm_os_memory_cache_counters
Returns an overview of the cache health, including run-time imformation about cache entries allocated, the source of memory for the cache entries, and how they are used.

sys.dm_os_memory_cache_entries
Returns information, such as statistics, for all entries in the various caches. This DMV is useful for linking cache entries back to their associated database objects.

sys.dm_os_memory_cache_hash_tables
Returns information about each active cache in the instance of SQL Server, such as the type of cache, the number and type of hash buckets, the length of time the hash buckets have been in use, etc.

sys.dm_os_memory_clerks
Returns all currently active memory clerks within SQL Server. A memory clerk is the primary means for allocating memory to the various users of SQL Server.

sys.dm_os_memory_objects
Returns all memory objects that are currently allocated by SQL Server. Memory objects are more granular than memory clerks and are used by internal SQL Server processes and components, but not users like memory clerks. This DMV is ideal for analyzing memory use and identifying memory leaks.

sys.dm_os_memory_pools
Returns a row for each memory pool object store in the SQL Server instance. Memory pool objects are certain homogeneous, equally important, stateless types of data. This information is sometimes useful for identifying bad caching behavior.

Sys.dm_os_nodes
Returns information about the currently active nodes on the instance. Nodes are created by SQL OS to mimic hardware processor locality and can be altered by SQL OS using soft-NUMA techniques.

Sys.dm_os_process_memory
Returns information providing a complete picture of the process memory address space in kilobytes, including things like the *page_fault_count*, amount of virtual address space available and committed, process physical and virtual memory, and so forth.

sys.dm_os_performance_counters
Returns a row per performance counter, the same counters as in Windows PerfMon, maintained by the server. To calculate a discreet value for the various per-second counters, you must sample two discrete values and then subtract the earlier cumulative per-second value from the later.

sys.dm_os_schedulers
Returns one row per scheduler for schedulers mapped to an individual processor. Active_worker scheduler threads are those devoted to the regular user-controlled work of the server, such as queries and SSIS jobs, while a variety of system and hidden schedulers may be active on the system at any time. This DMV is very useful for monitoring the thread scheduler and to find runaway tasks.

sys.dm_os_stacks
Used internally by SQL Server track debug data, such as outstanding allocations, and to validate the logic in a component that assumes that a certain call was made.

sys.dm_os_sys_info
Returns a miscellaneous set of useful information about the computer, such as the hyperthread ratio, the max worker count, and other resources used by and available to SQL Server.

Sys.dm_os_sys_memory
Returns a complete picture of memory at the operating system level, including information about total and available physical memory, total and available page memory, system cache, kernel space and so forth.

sys.dm_os_tasks
Returns one row for each OS task that is active in the instance of SQL Server.

sys.dm_os_threads
Returns a row for each SQLOS threads running under the SQL Server process.

sys.dm_os_virtual_address_dump
Returns information about the region, i.e. the range of pages in the virtual address space, used by a calling process.

sys.dm_os_wait_stats
Returns information about waits encountered by currently executing threads. There are a limited number of reasons why a thread might be forced to wait before it can complete execution. Refer to the SQL Server Books On-Line for more information about all possible wait states. This is an excellent DMV to use to diagnose performance issues with SQL Server and also with specific queries and batches because it records any time anything within SQL Server has to wait.

sys.dm_os_waiting_tasks
Returns information about the wait queue of SQLOS tasks that are waiting on some resource, such as blocking and latch contention.

sys.dm_os_workers
Returns a row detailing information for every worker in the system, what it is doing and what it is waiting to do.

To find out the number of data pages in the buffer cache for each database, from largest to smallest consumers of the cache:

```
SELECT count(*)AS cached_pages,
   CASE database_id
        WHEN 32767 THEN 'ResourceDb'
        ELSE db_name(database_id)
   END AS database
FROM sys.dm_os_buffer_descriptors
GROUP BY db_name(database_id) ,database_id
ORDER BY cached_pages_count DESC
```

To see all of the memory consumed by hosted components:

```
SELECT h.type, SUM(single_pages_kb + multi_pages_kb) AS commited_memory
FROM sys.dm_os_memory_clerks AS mc
   INNER JOIN sys.dm_os_hosts AS h
   ON mc.memory_clerk_address = h.default_memory_clerk_address
GROUP BY h.type;
```

To associate a SQL Server session ID value with a Windows thread ID that you could then track with Windows PerfMon, use a query like this:

```
SELECT STasks.session_id, SThreads.os_thread_id
FROM sys.dm_os_tasks AS STasks
   INNER JOIN sys.dm_os_threads AS SThreads
   ON STasks.worker_address = SThreads.worker_address
WHERE STasks.session_id IS NOT NULL
ORDER BY STasks.session_id;
```

To find if you have more currently running tasks than the maximum number of runnable tasks for the server and thus a likely CPU bottleneck:

```
SELECT scheduler_id,
   current_tasks_count,
   runnable_tasks_count
FROM sys.dm_os_schedulers
WHERE scheduler_id < 255;
```

To find out if any active queries are running parallel for a given instance requires a more sophisticated query:

```
SELECT r.session_id,
   r.request_id,
   MAX(ISNULL(exec_context_id, 0)) as nbr_of_workers,
   r.sql_handle,
   r.statement_start_offset,
   r.statement_end_offset,
   r.plan_handle
FROM sys.dm_exec_requests r
   JOIN sys.dm_os_tasks t ON r.session_id = t.session_id
   JOIN sys.dm_exec_sessions s ON r.session_id = s.session_id
WHERE s.is_user_process = 0x1
GROUP BY r.session_id, r.request_id, r.sql_handle, r.plan_handle,
   r.statement_start_offset, r.statement_end_offset
HAVING MAX(ISNULL(exec_context_id, 0)) > 0
```

SECURITY

sys.asymmetric_keys
Returns a row for each asymmetric key, containing (by default) both a public and private key. (The private key is protected by the database master key.)

sys.certificates
Returns a row for each certificate in the database. A certificate is a database-level security object loaded from a file or an assembly which follows the X.509 standard.

sys.credentials
Returns a row for each credential in the instance. A credential is a record containing authentication information needed to connect to resource external to SQL Server, usually including a Windows user and password.

sys.crypt_properties
Returns a row for each cryptographic property associated with each symmetric or asymmetric key on the instance.

sys.database_permissions
Returns a row for each permission or counter-permission in the database (that is, permissions on a column that are different from the corresponding higher-level, object permission).

sys.database_principals
Returns a row for each principal in the database. A principal is an entity (such as a Windows login, a SQL Server login, or an application role)

Sys.dm_audit_actions
TBD

Sys.dm_cryptographic_provider_keys
Returns information about keys provided by an Extensible Key Management provider.

Sys.dm_cryptographic_provider_sessions
Returns information about a cryptographic provider for either the current connection or all cryptographic connections.

Sys.dm_database_encryption_keys
Returns information about the encryption state of a database and its encryption keys.

Sys.dm_provider_algorithms
Returns options and details, including the algorithm details, of a specific Extensible Key Management provider.

Sys.dm_provider_properties
Returns a row for each registered cryptographic provider.

Sys.dm_server_audit_status
TBD

sys.key_encriptions
Returns a row for each symmetric key encryption that was created using the statement CREATE SYMMETRIC KEY with ENCRYPTION BY.

sys.login_token
Returns a row for each server principle (such as a Windows login, SQL Server login, or application role) that is part of the login token.

sys.master_key_passwords
Returns a row for each database master key password (used to protect the master keys kept in the credential store) created with the system stored procedure **sp_control_dbmasterkey_password** stored procedure.

sys.openkeys
Returns a row for each encryption key that is open in the current session.

sys.securable_classes
Returns a list of all securable classes on the instance.

sys.server_permissions
Returns a row for each server permission granted, including servers, server principals and endpoints.

sys.server_principals
Returns a row for each server-level principal, such as Windows and SQL Server logins, as well as logins mapped to certificates and asymmetric keys.

sys.server_role_members
Returns a row for each member of each fixed server role on the instance.

sys.sql_logins
Returns a row for each SQL login on the instance.

sys.system_components_surface_area_configuration
Returns a row for each executable system object, like a stored procedure or user-defined function, that can be enabled or disabled by a surface area configuration component.

sys.symmetric_keys
Returns a row for each symmetric key created using the statement CREATE SYMMETRIC KEY.

sys.user_token
Returns a row for each database principal (such as a Windows login, SQL Server login, or application role) that is part of the user token.

INDEX

sys.dm_db_index_operational_stats
Returns current low-level I/O, locking, latching, and access method activity for one or all databases, and one or all tables, indexes and partitions within each database.

sys.dm_db_index_physical_stats
Returns size and fragmentation information for the data pages and index pages for one or all databases, and one or all tables, indexes and partitions within each database. You may also specify a scanning mode to speed processing. You will usually get multiple rows even if specifying a specific index on a specific table in a specific database. The DMV returns one row for indexes for each level of the B-tree in each partition. One row is returned for heaps for the IN_ROW_DATA allocation unit of each partition. One row is returned for LOB data and rwo-overflow data, if they exist in each partition.

sys.dm_db_index_usage_stats
Returns a count of different types of index operations (such as seeks, scans, lookups, and updates) and the last time each operation was performed.

sys.dm_db_missing_index_columns
Returns information about columns that are missing an index, which can then be used to create indexes on those columns.

sys.dm_db_missing_index_details
Returns detailed information about missing indexes. Used in conjunction with the other *sys.xxx_missing_inde_xxx* DMVs and functions.

sys.dm_db_missing_index_group_stats
Returns summary information about groups of missing indexes. Used in conjunction with the other *sys.xxx_missing_inde_xxx* DMVs and functions.

sys.dm_db_missing_index_groups
Returns information showing which specific missing indexes are contained in a specific missing index group. Used in conjunction with the other *sys.xxx_missing_inde_xxx* DMVs and functions.

To grab the physical index information from AdventureWorks.Person.Address as quickly as possible:

```
SELECT *
FROM sys.dm_db_index_physical_stats(DB_ID(N'AdventureWorks'), OBJECT_ID(N'AdventureWorks.
Person.Address'), NULL, NULL , 'LIMITED');
```

I/O

This section contains the following dynamic management objects.

sys.dm_io_backup_tapes
Returns information about tape devices and the status of mount requests for backups.

sys.dm_io_cluster_shared_drives
Similar to *fn_servershareddrives* in SQL Server 2000, this DMV returns the drive name of each of the shared drives on a clustered server. Only returns data for clustered instances of SQL Server.

sys.dm_io_pending_io_requests
Returns one row for each pending I/O request, possibly returning large result sets on very active SQL Servers.

sys.dm_io_virtual_file_stats
Similar to *fn_virtualfilestats* in SQL Server 2000, this DMV function returns I/O statistics for data and log files for one or all databases and one or all files within the database(s).

SQL SERVER 2005 = ■ SQL SERVER 2008 = ■

OBJECT

The DMVs in this section help you keep track of objects that are referenced by other objects in their definition, such as triggers, stored procedures, and functions.

Sys.dm_sql_referenced_entities
Returns a row for each user-defined entity that is referenced by a server-level DDL trigger, a database-level DDL trigger, or a specific object in the current database context. (You can get server level information only when in the *master* database.) No dependency information is kept for rules, defaults, system objects and temporary tables and procedures.

Sys.dm_sql_referencing_entities
Returns a row for each entity that references by name an object you specify, whether that object is a partition_function, an XML_schema_collection, a type, or a specific object in the current database context. (You can get server level information only when in the *master* database.) No dependency information is kept for rules, defaults, system objects and temporary tables and procedures.

For example, to find the entities that reference the alias type *dbo.my_type*:

```
USE AdventureWorks;
GO
SELECT referencing_schema_name, referencing_entity_name
FROM sys.dm_sql_referencing_entities ('dbo.my_type', 'TYPE');
GO
```

If, on the other hand, we'd like to see the schema and entity (both tables and columns) that are referenced by a database-level DDL trigger called *ddl_db_trigger_log*, then we might use this query:

```
USE AdventureWorks;
GO
SELECT referenced_schema_name, referenced_entity_name
FROM sys.dm_sql_referenced_entities ('dd_db_trigger_log', 'DATABASE_DDL_TRIGGER');
GO
```

RESOURCE GOVERNOR

Sys.dm_resource_governor_configuration
Returns a row containing the current configuration state of the Resource Governor in memory. To see the same data, but stored in the long-term metadata, refer to *sys.resource_governor_configuration*.

Sys.dm_resource_governor_resource_pools
Returns a row for each current resource pool that is active in memory – their states, current configurations, and resource pool statistics. To see the same data, but stored in the long-term metadata, refer to *sys.resource_governor_resource_pools*.

Sys.dm_resource_governor_workload_groups
Returns a row for each current workload group that is active in memory, including configuration information and workload group statistics. To see the same data, but stored in the long-term metadata, refer to *sys.resource_governor_workload_groups*.

In the following example, we want to see both the original resource governor classifiers and the currently active classifiers for all schemas on the SQL Server instance:

```
USE MASTER
GO
SELECT OBJECT_SCHEMA_NAME(classifier_function_id) AS 'Metadata Schema',
   OBJECT_NAME(classifier_function_id) AS 'Metadata UDF Name'
FROM SYS.RESOURCE_GOVERNOR_CONFIGURATION
GO
SELECT OBJECT_SCHEMA_NAME(classifier_function_id) AS 'In-Memory Schema',
   OBJECT_NAME(classifier_function_id) as 'In-Memory UDF name'
FROM SYS.DM_RESOURCE_GOVERNOR_CONFIGURATION
GO
```

SQL SERVER EXTENDED EVENTS

SQL Server 2008 Introduces the concept of extended events, a general event-handling system that can correlate data from SQL Server, the OS, and applications using Event Tracing for Windows.

Sys.dm_os_dispatcher_pools
Returns a row for each currently active session dispatcher pool, that is, those thread pools used by system components which are performing background processing.

Sys.dm_xe_map_values
Returns the extended event mapping of internal number keys as human-readable text.

Sys.dm_xe_object_columns
Returns the extended event schema information for specified object. The *object_name* must exist in *sys.dm_xe_objects*.

Sys.dm_xe_objects
Returns a row for each object exposed by an event package, including events, actions, targets, predicates, and types.

Sys.dm_xe_packages
Returns a row for each package registered with the extended events engine.

Sys.dm_xe_session_event_actions
Returns a row for each event session action, including the number of times the action has been fired and its total run time.

Sys.dm_xe_session_events
Returns a row for each session event, its memory address, package GUID, and predicate.

Sys.dm_xe_session_object_columns
Returns a row for each configuration value assigned to objects that are bound to a given session.

Sys.dm_xe_session_targets
Returns information about session targets, such as the number of times the target has been executed for the session and the total time that the target has been executing.

Sys.dm_xe_sessions
Returns information about the active extended event sessions (e.g. events, actions, and targets) currently active in memory.

Author, ***Kevin Kline***

Kevin Kline has more than 18 years of experience in the IT industry and is a recognized SQL Server expert. In addition to being a Microsoft SQL Server MVP, Kevin is a regular contributor to database technology magazines and discussion forums. Kevin is the author of the O'Reilly titles *SQL in a Nutshell and Transact-SQL Programming.*