Central Management Server

CMS is a database to help manage a SQL Server farm and supports Virtual Machines, Managed Instances and Azure SQL databases. It helps answer questions such as.

Classic DBA questions beyond the “how much stuff do we have”:

* Which servers/databases changed settings? Are running out of space? Have grown significantly in size or logs? Are missing backups?
* Which servers are running at peak capacity? At what time? For how long?
* Which AlwaysOn clusters are unhealthy?
* Which indexes need to be Reorged? Rebuilt?
* Which jobs are taking too long or failing repeatedly?
* Which replicated databases are running behind? For How long?
* Who has more permission than should?

Classic Developer questions:

* Has a recent release caused SQL errors? Where? How many?
* Where am I getting long SQL commands? Blocks? Deadlocks?
* Which indexes are missing? Or duplicated?
* What are my hot spots for reads and writes? How do they vary across dbs?
* Where do I find a given object?
* Which databases are missing an object or have an extra object?

Stakeholder adhoc questions:

* Query all databases to find: “Patients allergic to aspirin”, “Caregivers who worked over 50 hours in a week?”, etc.

Requirements:

1. Create a new VM with the latest SQL developer edition with network access to ALL SQL services.
2. Execute the most recent CMS Schema.sql script.
3. Populate the servers table. Look up the script [insert server.sql].
4. Create linked servers. If the target database is SQL Azure we need a linked server for each database. Lookup the script [Create Linked Server.sql]
5. Create the SQL Agent Jobs to refresh data and send alerts periodically.

Utilities:

• spDailyChecks

Runs dozens of routine checks, worth to check it every morning.

• spHourlyChecks

Alerts which require prompt attention

• spSearch 'FactProductInventory'

This may be used to find where tables, routines and columns exist.

• spStopJob

Finds all jobs that start with a given name, stops and disables them. Useful to manage replication or other processes.

• spJobStart

Re enables jobs, but does not start them automatically.

• spAddColumn

One of the most common needs is to add columns to a table, this utility finds all databases and servers where a given table exists and adds the column.

If the column is already there it will skip that database. We recommend using params @Debug=1, @Exec=0 to get a printed output, so you can run commands arbitrarily.

With some creativity this proc can also be used to change data types, add constraints, etc.

exec spAddColumn @SchemaName='Person', @TableName='PersonPhone', @ColumnName='ChangedBy',@Type='varchar(100)',@Debug=1,@Exec=0

• spUpdatePublisher / spUpdateSubscriper

Allows executing commands on all publisher / subscriber databases at once, however this does not support long scripts separated by GO statements.

exec spUpdatePublisher @code='create or alter view vPersonPhone as select \* from Person.PersonPhone', @Debug=1, @Exec=0

• spDbCompare

Compares 2 databases and finds many sorts of differences. And runs a lot faster than Redgate Sql Compare.

exec spDbCompare @sourceserver='sqlvm1', @sourcedb='AdventureWorks2019Prod', @targetserver = 'sqlvm2', @targetdb='AdventureWorks2014QA'

• Diagnostics

exec spSearchServer 'sqlvm1'

exec spSearchDatabase 'AdventureWorks2019'

• spRunSql

Runs a command on servers matching some criteria

--get all server admin logins

exec [dbo].[spRunSql] @cmd='select \* from master..syslogins where sysadmin=1', @where='servername like ''%vm%'''

• spRunSqlDb

Runs a command on databases matching some criteria

--where are the gabriels?

exec [spRunSqlDb] @cmd='select [BusinessEntityID], [PersonType], [FirstName], [LastName] from [#database#].person.person where firstname=''gabriel''', @where='Databasename like ''%adventureworks201\_%'''

• Masking

--Which columnns are masked

SELECT \* FROM vwDatabaseObjectColumns

where masking\_function is not null

--Which columns are missing

select \* from [vwMissingMask]

where PurposeName <> 'prod'

• Compression candidates

select \* from vwIndexUSage

where size\_mbs > 100

and reads < 100

and isnull(data\_compression\_desc,'none') = 'none'