***what are sql server commands for soft boot***

In SQL Server, there isn't a direct "soft boot" command like you might find in operating systems or hardware. However, there are some commands and procedures you can use to effectively restart or restart SQL Server components in a less disruptive way.

Here are a few techniques that may serve similar purposes to a "soft boot" for SQL Server:

**1. Restarting SQL Server using**SHUTDOWN**and**STARTUP

You can use SQL Server commands to gracefully stop and start SQL Server services without a full machine reboot.

**Stop SQL Server:**

To stop SQL Server, you can use the SHUTDOWN command. This is similar to a graceful shutdown.

SHUTDOWN;

**Start SQL Server:**

Once SQL Server is stopped, you typically need to restart it using the SQL Server Configuration Manager, the Services applet in Windows, or the command line. SQL Server doesn't provide a direct STARTUP command from within SQL Server, so you'd restart the SQL Server instance outside of SQL Server (through SQL Server Management Studio or the SQL Server Configuration Manager).

Alternatively, you can restart it from Windows services via PowerShell:

powershell

Restart-Service -Name 'MSSQLSERVER'

Or if you are using a named instance:

powershell

Restart-Service -Name 'MSSQL$YourInstanceName'

**2. Use SQL Server Management Studio (SSMS)**

If you're working in SSMS, you can right-click on the SQL Server instance in the "Object Explorer" pane and select **Restart**. This will effectively do a soft restart of the SQL Server service.

**3. DBCC FREEPROCCACHE**

If you're trying to clear the procedure cache, you might want to use the DBCC FREEPROCCACHE command to remove cached execution plans without restarting SQL Server.

DBCC FREEPROCCACHE;

**4. DBCC DROPCLEANBUFFERS**

If you're looking to clear the buffer cache, DBCC DROPCLEANBUFFERS can be used to simulate a soft reset of memory without affecting SQL Server's operation too much.

DBCC DROPCLEANBUFFERS;

**5. Rebooting the SQL Server (Windows)**

While this is more of a hardware reboot than a "soft boot" within SQL Server, you can restart the SQL Server instance by rebooting the Windows Server it's running on. Use the following command in PowerShell or Command Prompt:

powershell

Restart-Computer

Or using Windows Services:

powershell

Stop-Service -Name 'MSSQLSERVER' Start-Service -Name 'MSSQLSERVER'

**6. Using SQL Server Agent (for jobs)**

If you're managing jobs through SQL Server Agent and you want to restart certain tasks or clear things out, you can use commands like:

EXEC msdb.dbo.sp\_stop\_job @job\_name = 'YourJobName'; EXEC msdb.dbo.sp\_start\_job @job\_name = 'YourJobName';

These won’t restart the SQL Server itself, but they are useful for controlling SQL Server jobs.

**Summary**

While there isn't a direct "soft boot" command in SQL Server, a combination of stopping and restarting the SQL Server service (SHUTDOWN for stopping and using tools like SQL Server Management Studio or PowerShell for starting) provides a way to achieve a similar effect. Additionally, clearing cache and buffers can also help refresh the server’s behavior without a full restart.

If you were looking for a specific kind of "soft boot," such as clearing memory or resetting connections, let me know, and I can provide further details.

***how to find when DBCC DROPCLEANBUFFERS was executed***

In SQL Server, DBCC DROPCLEANBUFFERS is a command that clears the clean buffers from SQL Server's buffer pool. It essentially removes pages from the buffer cache, which can be useful for testing or benchmarking purposes, but it doesn't leave a trace in the system tables or logs.

SQL Server doesn’t directly log the execution of DBCC DROPCLEANBUFFERS in a way that you can easily retrieve the exact time it was executed.

However, there are a few ways you might be able to track or infer when this command was executed.

**1. SQL Server Default Trace**

SQL Server has a default trace that records various server activities, including certain administrative tasks. While DBCC DROPCLEANBUFFERS might not specifically be logged in the default trace, you can check the trace for any associated activities such as server restarts or performance events that might coincide with the execution of this command.

To query the default trace for recent activity, you can use the following query:

SELECT \* FROM fn\_trace\_gettable(CONVERT(VARCHAR(150), (SELECT value FROM sys.fn\_trace\_getinfo(NULL) WHERE property = 2)), DEFAULT) WHERE EventClass IN (82, 83) *-- 82 and 83 refer to performance-related events* ORDER BY StartTime DESC;

However, this won't specifically tell you when DBCC DROPCLEANBUFFERS was run, but it might help if you're looking for general performance events around that time.

**2. SQL Server Extended Events**

If you want to track the execution of DBCC DROPCLEANBUFFERS specifically, you can set up an Extended Event session to capture the execution of DBCC commands. Here’s how you can create a custom Extended Event to monitor DBCC commands:

CREATE EVENT SESSION DropCleanBuffersTracking ON SERVER ADD EVENT sqlserver.sql\_statement\_completed ( WHERE (statement LIKE '%DBCC DROPCLEANBUFFERS%') ) ADD TARGET package0.ring\_buffer; ALTER EVENT SESSION DropCleanBuffersTracking ON SERVER STATE = START;

This Extended Event session will track the execution of any SQL statement that contains DBCC DROPCLEANBUFFERS. You can then query the captured events like this:

SELECT event\_data.value('(event/@timestamp)[1]', 'datetime') AS [Time], event\_data.value('(event/data[@name="statement"]/value)[1]', 'varchar(max)') AS [Statement]

FROM

(SELECT CAST(target\_data AS XML) AS event\_data FROM sys.dm\_xe\_session\_targets AS st

JOIN sys.dm\_xe\_sessions AS s ON s.address = st.event\_session\_address WHERE s.name = 'DropCleanBuffersTracking') AS event\_info;

This will give you a timestamp of when the DBCC DROPCLEANBUFFERS command was executed and any relevant details about the statement.

**3. SQL Server Profiler**

You can use SQL Server Profiler to capture the execution of DBCC commands. Start a new trace, and select the events related to SQL statements (like SQL:BatchStarting and SQL:BatchCompleted). You can filter the trace to capture only statements that contain DBCC DROPCLEANBUFFERS.

**4. SQL Server Error Log**

SQL Server logs certain actions to the error log, but DBCC DROPCLEANBUFFERS is generally not logged unless it coincides with other server events like restarts or significant errors. To check the error log, you can run:

EXEC xp\_readerrorlog;

However, this won't give you the specific time DBCC DROPCLEANBUFFERS was run unless there was some related event at the time.

**5. Custom Logging**

If you need to keep track of DBCC DROPCLEANBUFFERS for future execution, you might consider implementing custom logging. This can be done by wrapping the command in a stored procedure that logs the timestamp whenever the command is executed. For example:

CREATE PROCEDURE LogAndDropBuffers

AS

BEGIN

*-- Log the execution time*

INSERT INTO dbo.CommandLog (CommandName, ExecutionTime)

VALUES ('DBCC DROPCLEANBUFFERS', GETDATE());

*-- Execute the DBCC command*

DBCC DROPCLEANBUFFERS; END;

Whenever this procedure is executed, it will record the execution time in the CommandLog table.

**Conclusion**

Since DBCC DROPCLEANBUFFERS doesn’t natively log to system tables or error logs, the best way to track when it was executed would involve either setting up an Extended Event session or using a custom logging approach if you plan to monitor its execution over time. If you're trying to investigate past events, you may need to rely on general server monitoring tools or logs to infer when it might have occurred, but there's no built-in SQL Server feature that tracks the exact execution time of this command out-of-the-box.

Top of Form