Spencer Pease
Info 445
Quiz 4

Name four Dynamic Management Views (DMVs) presented in lecture and describe their use.

Dynamic Management Views are one method of monitoring a database system. Presented as system views, DMVs help a database technician to understand the current state of the database. Four examples of DMVs are:

- sys.dm_exec_sessions
    - This DMV provides an overview of who is connected to a database. This is useful for determining which users' activities are causing trouble with the performance of the database, allowing you to quickly remove them if need be to restore baseline operations.
- sys.dm_exec_requests
    - The DMV shows all of the current queries against the database. A bad or poorly formed query is often the cause of database hang-ups, so being able to easily identify which queries are problematic quickly lets you resolve issues with more precision and speed, aiding you in honoring SLAs and other requirements.
- sys.dm_os_performance_counters
    - Similar to the system monitor provide an easy to digest overview of the current state of the hardware, such as CPU, RAM usage, and disk activity. Not having to spend time searching for this information allows you to move to addressing problems much faster and with more confidence.
- sys.dm_db_index_usage_stats
    - Properly maintained indexes speed up read operations in a database, so it is important to monitor the current state of each index in the database. This DMV presents this information to you, allowing you to preemptively address problematic indexes before they become a significant performance issue, keeping the database in an optimal state for fast queries. This lets people answer questions faster, driving quicker decisions to outmaneuver competition.

Explain what is meant by 'Fault-Tolerance' and identify three system component examples.

The concept of 'fault-tolerance' in relation to databases came into popularity in the 50's and remained so until the 90's, due to the difficulties in figuring out how to build robust database hardware. During this period, several methods developed to add tolerance to a system, so that when hardware inevitably failed, there wasn't catastrophic data loss. Data loss would often lead to the collapse of the business as well, since a lack of information gathered from analyzing data prevents you from making mission-critical decisions with speed or confidence, allowing competitors to out-maneuver you.

One popular method of adding fault tolerance is to implement a RAID array, specifically RAID 1 for the best fault tolerance. In RAID 1, redundancy is added at the hardware level by sending data through a

controller, which mirrors the data on two or more completely separate disk ‘pools’, writing in serial. If some disks in the pool failed, there was a copy of the data in another group. Without this redundancy at the hardware level, data loss is a guarantee, as disk hardware has a limited lifespan, and preemptively predicting disk failure is hard to do.

Another way to add fault tolerance is to integrate a battery backup into your caching systems. Caching allows a database to operate faster by keeping relevant data and procedures closer to the processing center, which leads to less traversing data to perform an operation. Fast caching systems are volatile, however, meaning they lose data when they don’t have power. A battery, then, would provide the cache with the ability to maintain its state in the case of power failure, so that when power is restored there is less spin-up time, since relevant data is already close to the processor, meaning a business can resume operations faster, which is what consumers want and expect out of a product.

Multi-channel CPUs are another method of implementing fault-tolerance. Each channel in a CPU can perform its own operation, so if one channel of a CPU gets stuck in a software error or fails completely, operations can still be performed on other channels. This prevents hardware from becoming the bottleneck in a database system, meaning users can rely on the system more and ask complex, business-driving questions without worrying about failure.