DBA Services

Most Common Performance Issues (64% of all performance problems)

1. Insufficient or poor indexes
    a. Table scans impact disk performance and memory use, as well as lead to blocking
    b. It’s possibly to have too many indexes, which lead to performance hits on data modification queries (INSERT, DELETE, or UPDATE operation)
2. Inaccurate or missing statistics
    a. The query optimizer makes choices based on row estimates that come from these statistics
3. Bad T-SQL
    a. Moving too much data, writing overcomplicated code, using wrong object types, etc.
4. Problematic execution plans
    a. Most of times, these are fixed through code changes, statistics updates, or new indexes
    b. Other times, this occurs due to parameter sniffing gone wrong
5. Excessive blocking
    a. A lack of resources, not enough memory, CPU, or fast enough disks can lead to additional blocking
6. Deadlocks
    a. Caused by blocking, but is something separate
    b. If all your queries complete fast enough, the chances of a deadlock are very slim
7. Non-set-based operations
    a. Caused by cursors and other types of loop operations to force a row-by-row style processing
8. Incorrect database design
    a. Ensuring that your database is properly normalized and data is stored properly (ex. dates go into a datetime column)
9. Poor execution plan reuse
    a. Caused by things like dynamic T-SQL or inappropriate parameters, preventing plan reuse or parameterization
10. Frequent recompilation of queries
    a. While recompilation is generally desirable, there can be too much due to volatile data or poor code

Overview of (Recursive) Query Performance Tuning Process

1. Set performance target for application
2. Analyze application performance
    a. Ensure servers are not overwhelmed
        i. Process of capturing performance metrics varies depending on if the server is on VMware, Hyper-V, Docker, AWS, Azure, etc.
        ii. In general, focus on collecting metrics on waits and queues, especially around disk I/O, memory, and CPU
        iii. Network (health of the routers, cables, Wi-Fi repeaters, etc.) can also affect performance
3. Identify resource bottlenecks
4. Ensure proper configuration for hardware, OS, platform, SQL Server, database, and applications
5. Identify costliest query associated with bottleneck
6. Optimize query

Creating a Baseline

https://learning.oreilly.com/library/view/sql-server-2017/9781484238882/html/323849_5_En_5_Chapter.xhtml

Identify resource bottlenecks

This is a repetitive process that goes as follows:

1. Identify the bottleneck
2. Fix it
3. Validate the fix
4. Measure the impact and current performance
5. Start again with the next bottleneck

This process should be done for one bottleneck at a time, making one change at a time and validating that one change at a time.

In-depth Query Tuning

https://learning.oreilly.com/library/view/t-sql-querying/9780133986631/ch02.html

Process Overview

1. Baseline performance and resource use of costliest query
2. Set performance target for query (ex. every query has to meet a three-second minimum operation, with a few exceptions)
3. Analyze and optimize factors (such as statistics) that influence query execution
4. Analyze query for common problems
5. Analyze query execution plan
6. Analyze and prioritize operators to identify bottlenecks
7. If warranted, modify query and/or index. Afterwards:
    a. Measure performance and resource use again
    b. Determine if query performance improved
        i. If not, undo changes!
8. Determine if query performance is acceptable
    a. If not, return to step 4