Craig S. Mullins

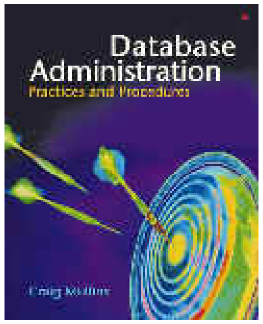
Return to Home Page

January 2005

DATABASE TRENDS AND APPLICATIONS

The DBA Corner

by Craig S. Mullins



Performance Tuning Requirements for Database Applications

A database application requires constant interaction between various disparate computing resources to operate efficiently and according to specifications. Realistically, the tuning of a database application can be broken down into three components: system tuning, database tuning, and application tuning. All of these are related, and database performance management requires an integrated approach to tuning across each of these areas.

The DBA needs to understand how the DBMS interacts with the server hardware, the operating system, and any other required software. Tuning and configuring these components and connections properly can have a dramatic impact on system performance. Therefore, system tuning occurs at the highest level and can have the greatest impact on the overall health of database applications. The reason for this is that every database application runs on the overall system. No amount of tuning is going to help a database application when the server it is running on is short on resources or improperly installed. Memory configuration is perhaps the single most important aspect of system tuning. Without sufficient memory, the entire DBMS system can grind to a halt. Another significant system bottleneck can be the database log. Every database modification is logged by the DBMS; an inefficient log can impact every application that modifies data. System tuning also includes managing and connecting other systems software with which the DBMS interacts, including the operating system, networking software, message queuing systems, middleware, and transaction processors.

Database application performance can be impacted by the physical design of the database, including normalization issues, how the database is stored on disk, number of tables, index design, and proper usage of DDL and its associated parameters. The actual physical location of database files on disk systems will have an impact on the performance of applications accessing the data. As more data is stored on the same disk device, the possibility of concurrent access and performance degradation increases. As data is inserted, updated, and deleted from the database, the efficiency of the database will degrade. Additionally, the files used to hold the data of the database may need to expand as data is added. Or perhaps, additional files, or file extents, will need to be allocated. Both disorganization and file growth can degrade performance. Indexes also need to be monitored, analyzed, and tuned to optimize data access, and to ensure that they are not having a negative impact on data modification.

The application code itself must be designed appropriately and monitored for efficiency. In fact, most experts agree that as much as 70 to 80 percent of performance problems are caused by improperly coded database applications. SQL is the primary culprit. Coding efficient SQL statements can be complicated. Developers need to be taught how to properly formulate SQL statements and SQL statements must be constantly monitored and tuned.

Programmers need to be schooled in the practice of examining SQL access paths. Every relational DBMS allows the programmer to request information on how the database optimizer will satisfy each query. Will an index be used? In what order will the tables be joined? Will the query be broken up into parallel tasks or not? These and many other factors influence the efficiency of SQL. Not all application problems are due to improperly coded

SQL. The host language application code in which the SQL has been embedded also can be inefficient, causing database application performance to suffer.
Managing the performance of your database applications requires in-depth monitoring. Be sure to allocate an appropriate budget to acquire performance management tools to ensure the efficiency of your database systems.

From Database Trends and Applications, January 2005.

© 2005 Craig S. Mullins, All rights reserved.

Home.