Tech Note

embarcadero®

The High Performance DB2 DBA

Managing Multiple Database Platforms for Performance & Availability

By Scott Walz, Sr. Director of Product Management at Embarcadero Technologies

June 2014

TABLE OF CONTENTS

INTRODUCTION

DB2 DBAs have always tackled complex tasks amidst shifting priorities, and pressures have only mounted in recent years. Several trends impacting DB2 DBAs include:

- **More Data** – Data growth is exploding. DBAs increasingly manage terabytes and even petabytes of data. This growth has created a relentless demand for increased storage and has made common database management tasks such as back-ups, capacity planning, and SLA compliance more complex.

- **More Databases and Instances** – While some DBAs use DB2 only, or manage only a handful of databases, most manage dozens and even hundreds. As databases grow in both size and number, so have database platform types and versions. Today, many DB2 DBAs manage complex heterogeneous database environments, making it more challenging to extend one's DB2 skills into other database types

- **Not Enough People** – Experienced DBAs are always in short supply. Many organizations cannot find the talent or expertise they need to manage their critical databases. This translates into a higher ratio of databases managed per DBA, which in turn puts the onus on organizations to find ways that will empower DB2 DBAs to be more productive and "scalable."

- **Consolidation / Virtualization** – The rush to consolidate and virtualize data centers has given rise to new kinds of issues in database administration, from virtualized server capacity planning, to performance monitoring and troubleshooting.

- **Multiple DB2 Versions** – DB2 continues to evolve and advance with each release, and DBAs are required to be experts on every version of DB2. New features, such as triggers and user-defined functions, have stretched DBAs from their comfort zone into a domain traditionally reserved for developers.

  Added Complexity – While all of the above trends are adding tremendous complexity to a DBA's daily work, IBM and other database vendors have been responding with databases that are increasingly "self-managed" or "self-tuning" through the use of more dynamic access to system health information and automated analysis tools. While advancements such as IBM's autonomic tuning STMM (Self Tuning Memory Manager) and advisor tools are time-saving improvements, DBMS environments are growing more complex and require knowledgeable human intervention and management. This can be especially acute when working with multiple database versions, and adding multiple database platforms can exponentially increase the complexity of the analysis needed.

Regardless of these challenges, as an overworked DB2 DBA, you are expected to keep key database systems available and optimized for high performance. Performance problems come from many sources, and you need a strategy that will contribute the most to high availability. This strategy should include the following components:

- Understanding the Database Design

- Catching Poor-Performing SQL Early
- Managing Storage
- Managing Performance
- Managing Capacity
- Managing Change

This paper will outline techniques to streamline and automate the management of these critical areas to deliver high database performance and availability—regardless of the database type or version.

UNDERSTANDING THE DATABASE DESIGN

DBAs are often required to support two or more types of database platforms. Whether the new database comes via a new application, a company acquisition, or the brainstorm of an app developer, supporting a new database can be challenging.

A great way to become more intimate with a new database and its structure is to understand the data model. Data modeling tools allow DBAs to reverse engineer the database. Doing so allows you to visualize the table structure, relationships, and stored procedures, as well as dependencies. By "seeing" the database on paper or screen, you can understand the impact of a potential change, generate reports for other users, and analyze the consistencies (or inconsistencies) among common data elements.

VISUALIZING THE DATABASE

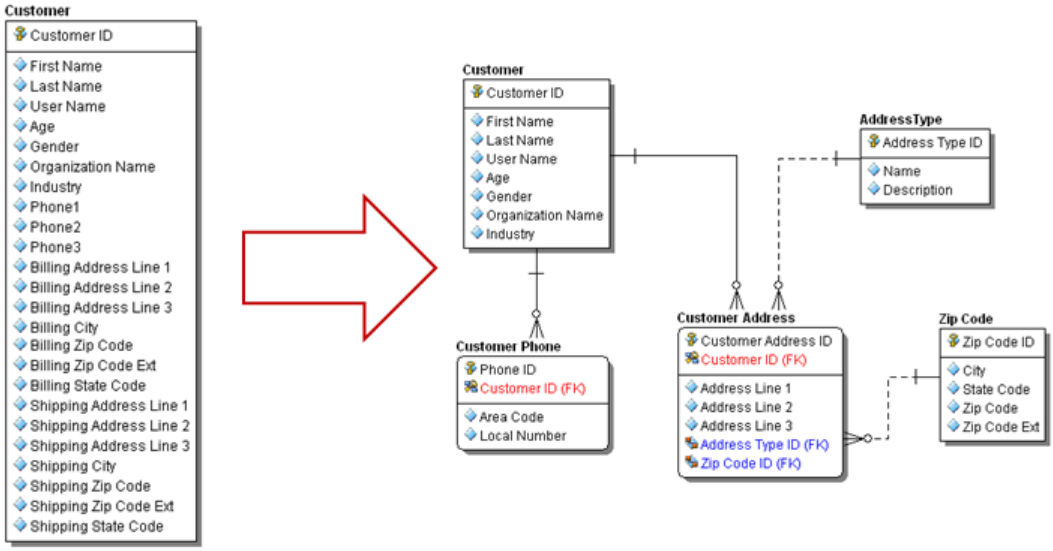
A logical data model is an invaluable tool for managing common objects across databases, especially across multiple database platforms. The logical model removes all platform-specific properties and allows you to use business names for tables and columns. This provides traceability for business terms and rules enforced in the model while maintaining traceability back to the physical implementations. Logical modeling helps DBAs by:

- Reducing data redundancy
- Organizing business requirements
- Providing a foundation of the database design
- Reducing development and maintenance time

Whether you inherit an existing database or build a new one, optimizing performance often starts with a thorough understanding of the database design. By staying in the loop with data architects and modelers during the development phase, you will have opportunities to examine the physical data model and its potential impact on performance.

DATA NORMALIZATION VS. PERFORMANCE

Logical data architects like to normalize the structure so that data redundancy is reduced and the design is as flexible as possible. A flexible design is good, but not at the expense of performance. Providing your expertise with the physical design on what tables can be denormalized (i.e. merged, rolled up/down, or horizontally/vertically split) can ensure that the design is both flexible and optimized for performance.



In the above example, we have provided flexibility by normalizing a bit to support 1 to N addresses and phone numbers for each customer rather than limiting it to two and three. However, we have kept some denormalization in the Zip Code table with the City and State Codes columns and in the Customer table with the Organization Name and Industry, allowing us to minimize the number of joins when accessing the data.

Performance issues with an existing database may make denormalizing the structure too drastic of a change. In that case, most data modeling tools allow you to reverse engineer an existing database with current row counts and partitioning properties. Then you can easily create a report that reveals which tables may be good candidates for new partitioning, and you can rethink existing partitioning strategies. You can also get a clearer picture of table relationships and start strategizing how to index specific tables.

OPTIMIZING POOR-PERFORMING SQL

CATCHING THE PROBLEM EARLY

Performance problems can occur at any stage of the development lifecycle, but catching them early is always much less expensive than letting them reach a production environment. Traditionally, developers have been responsible for eliminating poor-performing SQL code. For developers, the database is a black box, and they do not necessarily have access to the information they need to pinpoint and resolve problems.

Also, the roles of the database developer and database administrator are expanding. As a high performance DBA managing multiple platforms, you should enable developers to solve, or at least recognize and communicate, how their code is affecting database performance. One strategy is to create opportunities to offload SQL performance tuning to development. Enabling developers to identify and fix poor performing code before it hits production can save everyone time and energy, while improving database performance.

However, the first step is to identify that the performance problem is related to poor-performing SQL or a different type of bottleneck.

IDENTIFYING THE MAIN SOURCE OF THE BOTTLENECK

When looking at database bottlenecks, it's important to clearly identify the source, which usually falls into one of four categories:

- Insufficient hardware
- Database configuration issue
- Application code problem
- Poor-performing SQL
- Missing indexes or indexes that do not match the SQL

Understanding the source of the bottleneck helps you allocate resources efficiently. For example, there are many cases where a manager purchases additional hardware to address a performance issue that, in the end, turns out to be a database configuration or SQL tuning issue. The costly hardware upgrade fails to improve the bottleneck.

So, how can we identify the source of a performance bottleneck? The key is to profile the activity of the database to calculate the load on the database, pinpoint poor-performing SQL, and top sessions. These performance views describe the current state of the connections on the database and which SQL statements the connections are executing.

By sampling these tables, you can create a clear picture of load on the database and identify poor-performing SQL, as well as where time is being spent in the database. If time is being spent waiting for CPU resources and the highest CPU consuming SQL statement is only 3% of the total activity (as shown in the first profiling session graphic on page 7), then it’s not worth the time to tune all your SQL to free up the CPU. In this case, a hardware upgrade is the best way to allocate resources.

On the other hand, if the worst performing SQL statement is consuming 86% of the CPU (see second profiling graphic on page 7), you might spend a day or longer trying to tune the SQL statement. Clearly, this would be a much more efficient approach than upgrading the hardware.

If the bottleneck turns out to be access to a shared resource, the developer needs to ask the DBA what the issue is so the DBA can take the appropriate action. However, if the issue is row level locking or if the application is doing a lot of single row operations (when it should be using arrays), the solution to the problem is in the hands of the developer.

SQL TUNING TECHNIQUES

Often times, the main bottleneck is a SQL statement taking up too many resources. Once the offending SQL has been identified, developers can analyze the code. Ask yourself these questions to help you identify poor-performing SQL:

- Are the table and index statistics up to date?
- Are there any missing indexes?
- Do any of the columns need extended histogram or frequency statistics?
- Are all the unique and not-null constraints correctly defined on the columns and tables?
- Is the database choosing the right access path?

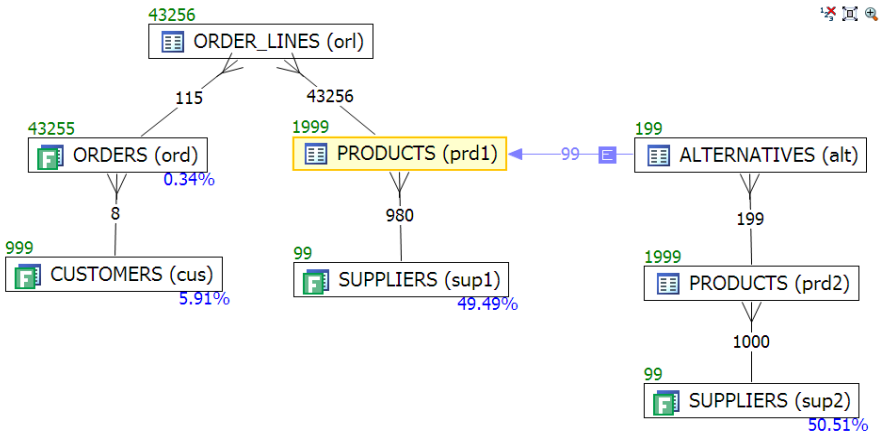
For the final step, of knowing whether the database chose the right path, or even if the SQL statement is reasonable, the best and most efficient step is to lay the query out visually. This is often referred to as Visual SQL Tuning or VST. Try drawing out the tables (either on your own or using a tool) with the detail tables hovering above the master tables, indicating table sizes, filter ratios, and two table join sizes.

By drawing the picture, you can find the best path through the query. Start at the most selective filter ratio, and follow the join to the next table with the smallest result set size, then on to the smallest two table join size.

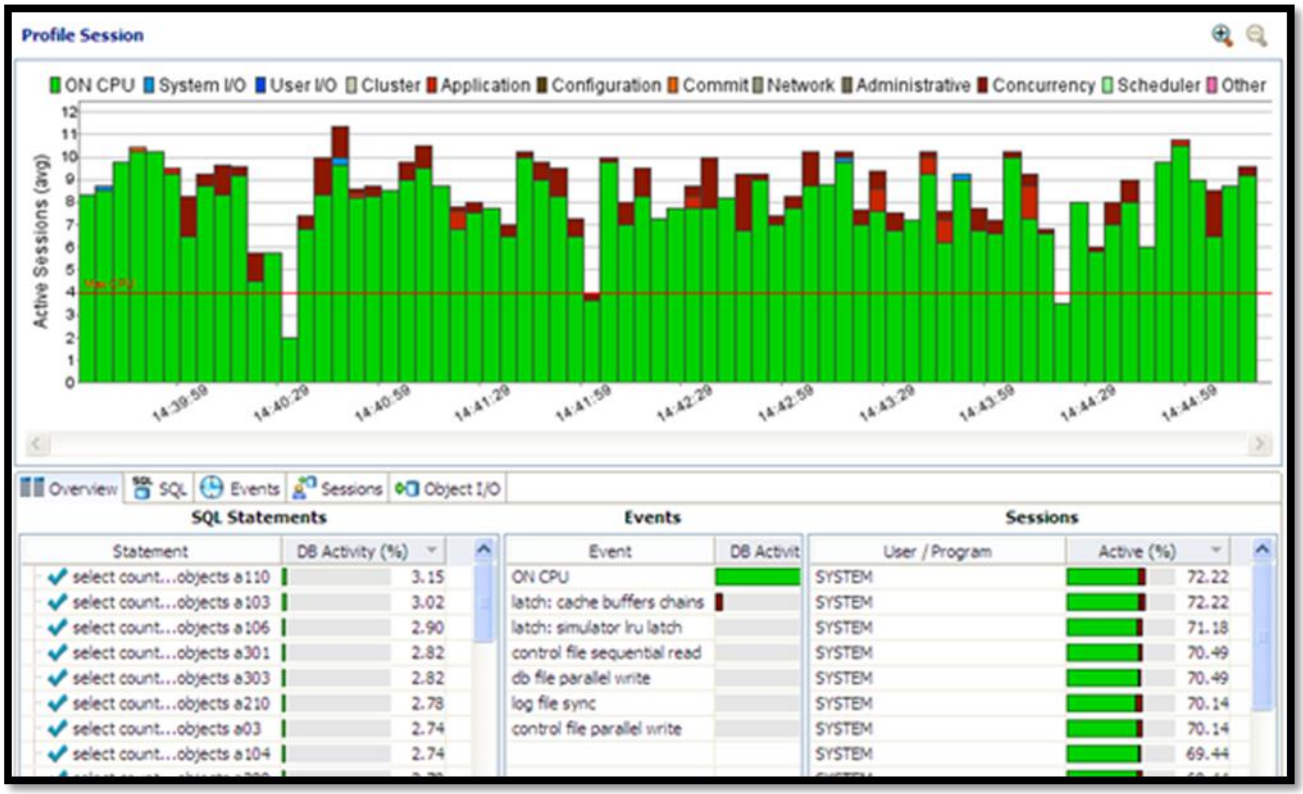
To the right is an example of a Visual SQL Tuning diagram. A good candidate for the best execution path is clear.

First we start where the amount of rows returned after filtering is the lowest, in this case ORDERS at 0.34%.

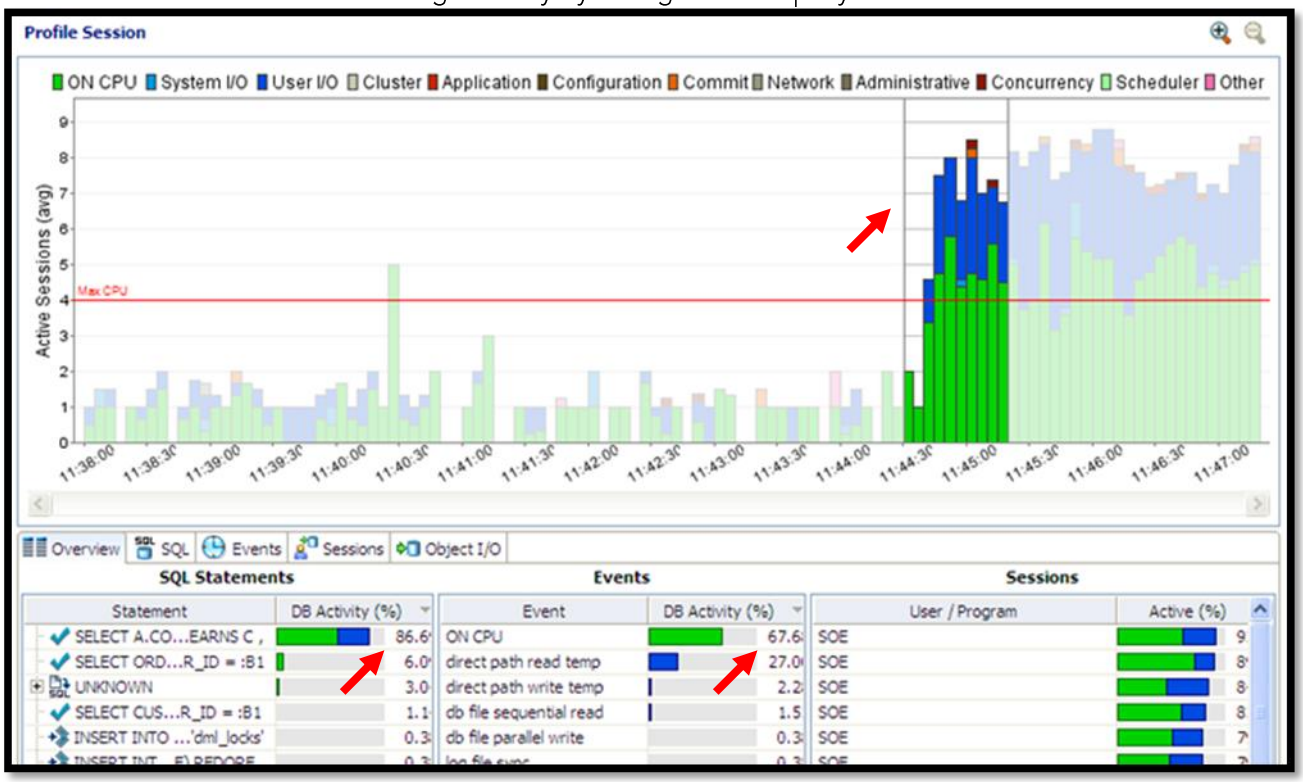
Then down to CUSTOMERS as that join only returns 8 rows. Then we join up to ORDER_LINES only giving us 115 rows, etc.



In the profiling session below, a DBA may spend too much time to make small gains by trying to tune individual statements.



However, in the profiling session below, we can see that there is one SQL statement that accounts for 86% of the load. The load can be reduced significantly by tuning this one query.



STORAGE MANAGEMENT

Storage management is complex enough, but when you manage different database platforms and versions on a variety of operating systems, the challenge becomes extreme. For DB2, you need to account for acceptable file placement (isolating log files) and ensure your operating system volumes are properly mirrored and configured. Today's DBA may be dealing with Sybase systems, databases, transaction logs, and segments in the morning and then have to switch gears to deal with Oracle tablespaces, tempfiles, and redo logs in the afternoon. Just remembering the storage syntax commands to manage the many diverse platforms can be a difficult task.

The two major storage management concerns are always availability and performance. Storage problems can quickly bring down a database. Just let an Oracle archive log destination reach maximum capacity or let a DB2 transaction log fail to expand and see how a dynamic transaction processing system reacts. Fortunately, database vendors have provided a number of new capabilities that address availability issues. All popular databases now sport an "auto-extend" feature allowing their underlying operating system files to automatically grow to meet the demand of increasing data volumes. DB2 automatic storage and database and transaction log files have the capability to expand when necessary. Still, such features are not a 100% guarantee against failure. A DB2 database file could meet an imposed growth limit or an Oracle temporary table space tempfile could be denied expansion at the operating system level if a disk drive runs out of space. Therefore, a DBA must still monitor storage to ensure availability.

A storage problem may also have a major impact on database performance. Fragmentation—both globally and at the object level—can result in significant unnecessary I/O for a database. Wasted space in tables and indexes can also cause problems and contribute to excessive table and index scan times.

In addition, object storage problems can create major performance roadblocks in a database. A DBA must understand how to detect and eliminate these issues as quickly as possible.

Regardless of platform, object storage problems generally include four components:

- **Wasted Space** – While it may not seem like a problem, a table or index with more allocated space than it needs can hamper performance. For example, if the pages of a DB2 object contained in a data warehouse are not 100% utilized, the object will examine more pages than necessary during table scans. This increases response time for user queries. The same problem arises with Oracle tables that have high-water marks above the areas where data resides.
- **Poor Extent Proximity** – A database engine's read-ahead mechanisms work more efficiently when objects are contained in extents that are close to one another. For example, DB2's sequential prefetch can read much larger groups of data when its target object has extents and pages that are contiguous. Sybase and SQL Server have similar facilities. Conversely, when an object has poor extent or page proximity, scan times can increase.

- Out of Sync Data Order – When indexes have a logical order that does not match the actual physical order of data stored in the database, index access performance can be affected. If the logical and physical orders are in sync, the disk head can scan in one direction instead of moving back and forth to obtain the needed information. Otherwise, the disk head will skip across the disk many times. In DB2, for example, index fragmentation is one of the more common culprits when determining root cause for performance issues.

- Overflow Rows – Each row in a table must fit on a single page. However, if a row expands because of the growth of variable length columns, the database may either relocate the row to another page and leave behind a pointer to the new page, or relocate the row to a new, less optimal, page. Neither situation is preferred as they increase the amount of I/O necessary to obtain the row. This is seen in Oracle as chained or migrated rows and in SQL Server and Sybase as forwarded rows.

So, how does a DB2 DBA detect and diagnose these object storage problems in less familiar databases? After all, each engine has its own set of diagnostics. Moreover, how can you ensure that no downtime is attributed to space outages across many servers? As a high performance DBA, you should establish an around-the-clock, multi-platform, fully-automated database monitoring solution that allows you to configure storage thresholds and proactive notifications across all your critical databases before you develop storage-related problems. Such a plan should take into account the multiple types of databases in their environment and also focus on being completely automated 24/7, at least for critical databases.

What about diagnosing and remedying database and object storage problems that hurt performance? This can be a challenge for the DBA tasked with taking on additional platforms. The fact is there are vast differences in diagnostic and treatment methods that exist across diverse database platforms.

The solution is to design an advanced storage diagnostic and management system that incorporates the following characteristics:

- Eliminates all the guesswork and labor from handling database storage problems in heterogeneous environments
- Allows you to handle complex storage dilemmas even if you are unfamiliar with the underlying database platform
- Alerts you when storage issues in the database objects are draining performance by visually identifying storage problems and pinpointing the objects that require attention

Because storage management can be complex, especially if you work on multiple database platforms, you must be proactive in your approach and automate as much of the management as possible. A two-phase approach to setting threshold alerts starts with standard thresholds designed to uncover space issues threatening overall system performance, then customizes them to meet the unique needs of your environment. For DB2, target your data and log file groups that are experiencing high growth rates so you have plenty of notice to solve potential problems.

A best practice is to rebuild your database objects for maximum performance at regular intervals. For example, create a find-and-fix job that searches the entire database (or a particular set of objects) for objects that violate your specified set of thresholds. Then reorganize them for better performance.

Once these jobs are set up and scheduled, you will never have to manually reorganize your databases again.

PERFORMANCE MANAGEMENT

DBAs want their databases to run as quickly as possible. The same can be said for every person who uses a system that connects to those databases—they want their queries and processes to have the shortest possible response times. Again, the situation is complicated when you have more than one database platform. As with storage management, the variables and remedies differ from one platform to the next. As a result, you need to put together a platform-neutral roadmap that applies to each of the database engines you manage.

The starting point is to define the different methods of analysis that will be used across all platforms and then apply a set of diagnostics and actions that can be used for each platform under each method. These methods of analysis include:

- Bottleneck/Response Time Analysis
- Workload Analysis
- Ratio Analysis

The following examines how each method can be used to manage the performance of any database.

BOTTLENECK/RESPONSE TIME ANALYSIS

Regardless of the platform, when a database is running, every connected process is either working or waiting to perform work. A process that is waiting may mean nothing, or it can be an indicator that a database bottleneck exists. DBAs use bottleneck or response time analysis to determine whether perceived bottlenecks are hurting performance.

Bottleneck analysis is an essential method of measuring performance because it helps you track where a database has been spending its time. In DB2, looking at Buffer Hit Ratios, number of times specific queries are run and how long they run, along with disk queues and other activity counters will help point you in the direction of a poorly performing query or hardware issue.

If DB2 table-scan activity or heavy Oracle latch contention has been dragging a database's performance, you can use bottleneck analysis to pinpoint the root cause. Once one or more of the potential sources has been identified, drill down to detail about the sessions and objects that are causing the problem.

This methodology is the strategy of choice for top performance analysts in the industry. This being the case, nearly every database vendor has tailored edits engines to report metrics to analyze bottlenecks and response times. DB2 uses table functions and administrative views to provide information to the DBA. Oracle 10g introduced new metrics in its V$ performance views that enable DBAs to understand bottlenecks and response times at global and session levels. Microsoft SQL Server made wait events available via an undocumented DBCC command (DBCC SQLPERF(WAITSTATS) and now exposes that data via dynamic management views ("DMVs"), and Sybase offers new monitoring views in engine versions 12.5.03 and above.

WORKLOAD ANALYSIS

Workload analysis involves the investigation of two critical areas of database performance:

- Session resource consumption and activity
- SQL execution analysis

When performance on a database drops suddenly, one or two sessions often generate the bulk of the workload. The issue is system balance. In a well-balanced system, no single session should consume the bulk of resources for an extended period of time, with the exception of batch job processes that are run safely in non-peak hours. If individual sessions are using a majority of system resources, the DBA should examine each problem session in detail to uncover session activity.

Normally, this can be easily accomplished by viewing session metadata coupled with resource consumption and statistical execution statistics. Most database engines provide vast details regarding a session's current and historical activities. SQL Server, for instance, provides this data via several dynamic management views, one being dm_exec_session.

Understanding current and historical SQL execution patterns will enable a database analyst to have the second set of data points needed to properly perform workload analysis. Optimizing SQL code will produce the second-best performance-enhancing boost available for a database—with proper physical design being the first. But what set of metrics should you use to evaluate 'good' versus 'bad' SQL? Naturally, factors such as overall elapsed time, CPU time, and I/O activity play a part. But other, more subtle factors can make a difference—like the number of times a statement is executed, or whether sorts are done in memory or on disk. As we saw on page 6, this is where SQL analysis becomes more of an art form than a science.

All the database vendors offer a window into SQL analysis. Oracle 10g has the most complete set of metrics offered by any vendor. DB2 offers its own SQL tracing capabilities. Microsoft SQL Server offers code profiling utilities that can be used to collect and measure SQL and stored procedure executions. Sybase has made a good start with its SQL execution views in versions 12.5.03 and above.

RATIO ANALYSIS

Ratio-based analysis was for many years the only technique DBAs had to diagnose the cause of a database slowdown. You are probably all too familiar with the DB2 buffer hit ratio, Oracle buffer cache hit ratio, Microsoft procedure plan hit ratio, Sybase data cache hit ratio, and so forth. Many performance experts advise that such ratios are now worthless and misleading because of advances in bottleneck/response time analysis. There is an element of truth to these claims, but some ratios are still quite valuable, such as the Oracle library cache hit ratio and the Microsoft and Sybase procedure plan hit ratios.

When working through performance issues, buffer hit ratios in DB2 will help define utilization on the server. In addition, by looking into the queries and query execution plans, you can see whether and how often indexes are used and whether the indexes have been created on the most beneficial columns in the queried tables. Then you can review fragmentation ratios to determine which indexes may need to be rebuilt or reorganized.

Ratios can be helpful, but the information is being muddied in many installations, by factors such as feature sets, server functionalities and other issues outside the database engine's realm of control

and reporting. For example, the use of a SAN disk subsystem can confuse disk access reporting to mask issues of data access, making it difficult to determine whether a slow disk system may be the root cause of performance issues.

More obscure ratios can also be useful. For example, if you were told that a DB2 table had 100 overflow rows, would you consider this good or bad? Of course, you cannot answer this question unless you know how many total rows are in the table. But if you knew that 20% of all row accesses in your DB2 table overflowed, this is enough information to make a reorganization decision. In these types of situations, the application of proper ratio analysis can be advantageous.

Similar measures exist for DB2 and will be helpful based on the specific situations you're trying to address. There are a host of counters available from Performance Monitor at the operating system level. These counters will let you explore and record information for everything from the disk subsystem to the instance performance. Further, with the administrative views available for query performance and index analysis, you can gain additional insights into suggestions for investigations.

CAPACITY MANAGEMENT

Many DBAs find it difficult or almost impossible to implement capacity management, as their day-to-day activities simply leave no room for historical and proactive analysis. This is unfortunate because proper capacity planning can help both the DBA and management, answering important questions such as, "How much more storage will this database need in six months?" and "Is this database server currently underutilized?"

Capacity planning generally involves three processes:

- Collection of key database metrics
- Historical analysis of collected metrics
- Forecasts of future needs

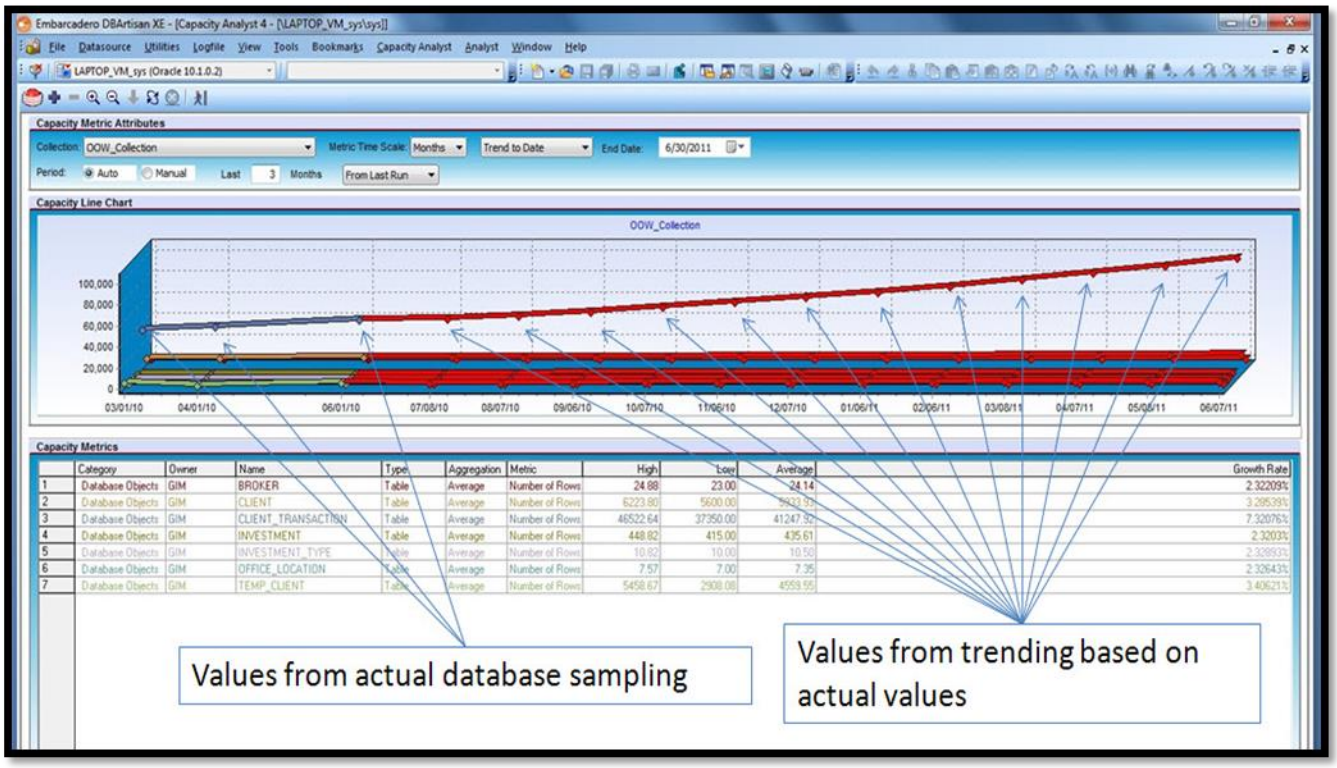
Every organization will have different needs, but in general, most should collect the following information:

- Snapshots of global storage usage, including Sybase device metrics, DB2 database statistics, Oracle tablespace data, and database object statistics that concern space usage. Elements such as total allocated space, used space, and free space should all be collected as well as other performance-related items like number of chained rows
- Snapshots of performance metrics, including key database statistics like I/O performance and wait events
- Snapshots of resource consumption, including metrics such as CPU usage, memory usage, and user traffic

While some DBAs have been able to get by using scripts and manual processes to manage storage and performance, the burden in using such techniques for capacity planning is usually too high. These tasks are complicated greatly by multi-platform database environments, as well as the need to build and maintain a repository that holds all the necessary statistics. In addition, DBAs are faced with challenges in the time it takes to manually weave together collection scripts and scheduled jobs as well as analyze and manage the generation of complex historical and forecasting reports. With all these complications, it is obvious that manual maintenance is not practical for the high performance DB2 DBA—especially when moving into other DBMS platforms.

The following techniques to simplify capacity management will save you time and headaches:

- Building a repository to track key database metrics
- Creating statistical collections for everything you want to track across your key databases
- Scheduling to automatically run collections as often as needed
- Implementing tools to accurately predict future needs by depicting upcoming storage, performance, and resource requirements through visual graphs (example below) or well-formatted HTML reports that managers easily understand



CHANGE MANAGEMENT

Today's DB2 DBA sits at the center of a complex system that touches many participants: data modelers, database developers, architects, business analysts, software developers, and more. At any given time, each database is physically instantiated across a number of different environments, each of which may contain any one of several versions of the same database. Furthermore, the design of a particular database is typically stored in several locations using a multiplicity of tools, which may include a data modeling tool, a SQL development tool, a database administration tool, a database change management tool, etc.

Very few organizations look at database change management as a single, unified process. Many assume that database change management is a natural outcome, at the juncture of database management, software configuration management, and data modeling. The first step in developing a robust database change management solution is to recognize it as a standalone process or discipline, with peculiarities that make it quite different from software change management.

As the DBA, you should develop a database change management process that provides a view into the changes occurring across all of your multiple database platforms. You should establish solid baselines and manage change to those baselines. In addition, best practices dictate that you periodically revisit the baseline to ensure that elements are properly versioned and if needed, reset the baseline. Having a consistent view with consistent reports, no matter the database platform, provides confidence that database change is being managed. You can also incorporate database configuration into your change management process. A change to a configuration setting could have as much impact as an object change.

Third party applications present their own unique set of challenges. Though most limit the DBA's control, at the end of the day, you are still responsible for an application's performance. A comprehensive database change management process allows you to understand and predict changes that occur when vendor updates are deployed. With that information, the DBA has a starting point to diagnose problems that may arise when applying patches. Without a process in place, you have to rely solely on the quality of the vendor's updates and their ability to remotely troubleshoot.

Just as important as object level changes are changes made to the security model. With increasing compliance and reporting pressures, DBAs must establish that they have control over the database. Being able to flag erroneous, or even malicious changes to database security, and then act on those changes, can be critical to meeting expected compliance levels.

Without a robust change management solution for your multiple database platforms, you could be putting the work of the other disciplines of administration at risk; a single change could result in a serious database issue or, even worse, could be a direct reflection on your ability to successfully administer the database.

Conclusion

The combination of multiple database types and versions, added database complexity, more data, and lower headcount has become a real challenge for the DB2 DBA who works hard to maintain high database availability and performance. The key to keeping databases running well is to implement a strategy for ensuring good design documentation and high quality SQL. Proper storage, performance, capacity management, and change management are also critical. Such a strategy should be platform-independent and as automatic as possible.

In the end, the responsibility for the performance of an organization's database structure falls on the DBA. And a well-prepared DBA should have all the tools she needs to ensure high database availability and performance.

Embarcadero Database Tools

Embarcadero Technologies provides a complete set of professional-grade database tools for the high performance DB2 DBA to take control of SQL, storage, performance, capacity management, and change management. ER/Studio allows you to reverse engineer the ERD so you understand the task at hand. Embarcadero DB Optimizer XE, DBArtisan XE, and Analyst Series Tools are designed to deliver productivity gains, regardless of your current level of expertise. DB Change Manager XE enables you to simplify, automate, and report database change. In addition, these essential support tools will give you the confidence to carry out even the most complex tasks, which goes a long way to ensuring the success of your databases.

About the Author

Scott Walz has more than 15 years of experience in database development and serves as senior director of product management for Embarcadero Technologies. In this position, Scott oversees the direction of the company's database product family, while focusing on database development and administration products. Prior to joining Embarcadero eight years ago, Scott served as development lead for Louisville Gas & Electric. He holds a bachelor's degree in computer information systems from Western Kentucky University.

Embarcadero Technologies, Inc. is the leading provider of software tools that empower application developers and data management professionals to design, build, and run applications and databases more efficiently in heterogeneous IT environments. More than 90 of the Fortune 100 plus an active community of more than three million users worldwide rely on Embarcadero's award-winning products to optimize costs, streamline compliance, and accelerate development and innovation. Founded in 1993, Embarcadero is headquartered in San Francisco with offices located around the world. Embarcadero is online at www.embarcadero.com.