White Paper presented by

Analytic Data Solutions, LLC

Title: An End To End Perspective On Managing Data Warehouse Performance

Author: By John Majarwitz

Date Written: 3/14/2011

Date Posted (to website): 12/19/2011

Brief Description:

This paper describes a proactive approach to prevent performance problems from arising in the first place and a diagnostic approach for determining the root cause of performance problems and the corrective actions to undertake when performance problems do arise.

An End To End Perspective On Managing Data Warehouse Performance

By John Majarwitz, Principal Consultant, End To End Information Solutions, Inc.

3/14/2011

An End To End Perspective On Managing Data Warehouse Performance

Overview

All too often, in the face of database performance problems for large scale data warehouses, the solution is to throw more hardware at the problem. The simple truth is this is just one option that should be available, and justifiable, only when other options have been systematically ruled out. More often than not, data warehouse performance problems are caused by factors other than having undersized system resources. In many cases, capacity planning is reasonable for the initial deployment of system resources for data warehouses. However, despite adequate system resources, there can still be performance problems. This paper describes a proactive approach to prevent performance problems from arising in the first place and a diagnostic approach for determining the root cause of performance problems and the corrective actions to undertake when performance problems do arise.

Causes of Performance Problems

Before discussing the preventative approach, let's first identify some basic causes for performance problems. The best way to fix a problem is to prevent it from occurring in the first place and so a better understanding of what causes problems may help you to create that ounce of prevention. Some common problem areas include:

Project Organization - Building a data warehouse environment is a complex organizational undertaking. In addition to the business organizations driving the requirements and the PMO organization managing the work, there are typically three primary development organizations: the ETL organization, responsible for putting data into the warehouse; the data design organization, responsible for the data models and databases for the warehouse; and the BI organization, responsible for delivering usable data from the warehouse. Across this organizational setting, there is a need to balance performance between front end query access and the back end ETL. There is also a need to balance performance between short term quick delivery and long term scalability as data volumes increase. Since ETL work is usually begun first, there can be an imbalance of focus on optimizing performance for loading data at the expense of optimizing data structures for query access performance. A classic example is a warehouse that has too many table structures organized by "silos" of source systems, which is easy to load, but puts a high burden on front end queries to handle the complexity of joining, conforming, and integrating data across sources. In summary, it is normal for organizations to be best at focusing inward on their responsibilities. This makes it hard for cross-cutting concerns, such as performance, to be addressed.

Time Pressure – At times, even with adequate insight, poor design decisions are made, especially in the data design. To be fair, other priorities can complicate the decision making process. For example the urgency to acquire and load more sources of data or to produce more types of reports may be given

higher priority than time spent on designing with performance in mind. If the project falls behind schedule, which is often the case, then the pressure to deliver becomes even greater and shortcuts are taken, meaning that performance testing is often reduced or even eliminated.

Vague Requirements - Because a data warehouse serves multiple business functions and because it is more difficult to get a group of people to agree, there is frequently a lack of quantification of performance service levels established in the requirements. In other words, the performance goals are too vague to be actionable. For example, performance requirements stated as qualitative objectives such as, "reporting run times must meet user's expectations" or, "the ETL cycle must complete overnight" do not provide a clear and measurable definition of the expected performance.

Design By Committee – While many have a stake in the data design, it does not mean that the data design process should be all inclusive. A good data model is highly patterned and standardized. There is also a structured evaluation sequence that should be applied to determine the entities, attributes, and relationships. Finally, experience is a key asset. All of these factors lead toward creating a small, core, modeling team, maybe even as small as two or three individuals depending on the amount of work involved. An extended review team should be in place comprised of key stakeholders. There should also be transparency in the design process meaning the designs are readily available, even as they are being worked, and the decision making rationale is explained. There are many symptoms of data design processes that are done by committee or are passed from one individual to another over time. These include the presence of indicative attributes in fact tables, measures in dimensions, lack of aggregate structures, too many dimensions, too many measures, too many dimensional attributes, poor naming conventions, redundancy, lack of conformance, overloaded attributes, and so forth. Altogether, this can lead to performance problems due to additional query complexity and due to confusion because the design is hard to understand.

Poor Planning – Identifying potential data access challenges early on and prototyping design approaches can go a long way to eliminating performance problems down the road. In the planning phase there is often a lack of concrete information regarding amounts of data, types of data, the largest queries that will be run, etc. This often leads to leaving requirements blank rather than putting a stake in the ground. This is a bad approach to take as it relates to performance because it prevents developing early thoughts and approaches on how to begin addressing performance needs. A better approach would be to put a stake in the ground by creating initial high level requirements as a proxy until such time that the true business stakeholder can confirm the actual requirements. Even if the assumptions in the proxy requirements are off target, it is difficult to imagine this creating more harm than having blank requirements and continuing to move forward in the project planning activities. As it relates to performance, this approach would be beneficial because it is not necessary to have complete requirements before performance related prototyping can begin. In fact, there are usually a small number of performance critical functions in comparison to the total number of functions the system will be required to perform. These might be one or two processing intensive ETL tasks and several query access patterns. Identifying these functions early and proceeding to design them with performance in

mind will help prevent performance issues down the road. Although the detailed requirements may not be known, the basic performance characteristics can still be investigated while an iterative process is established to refine and deepen the design as more requirements detail emerges.

Problem Deferral – It is easy to ignore performance concerns because it cannot be proven that there will be a performance problem until it actually arises. So sometimes there can be an attitude that "performance is not an issue" until an actual performance problem exists. In fact, it is difficult to quantify the value of any work to mitigate potential performance issues before they arise, because, by definition, the benefit is the absence of a problem (and the absence of costs associated with the problem). This can lead to an approach where it is assumed that performance will not be a problem until there is proof that it is a problem. But by then it is too late. The irony is that once a performance problem exists, it is more difficult to address because of the extra pressure of urgency and customer exposure as well as the general unavailability of resources available to focus on the problem. In a sense, by wearing "blinders" to potential performance issues throughout the project the organization has created a large amount of "technical debt" related to performance engineering which is now coming home to roost. It is better to operate under the assumption that there will be performance problems. This means establishing work processes to explicitly demonstrate that performance is not a problem. Only when proof of acceptable performance is demonstrated should the system then be considered to not have a performance problem. This is a case where guilty until proven innocent is better than innocent until proven guilty because of the high cost of suffering from performance problems once they arise.

Scope Creep – Changes in requirements that are not addressed through a disciplined change management process can result in development organizations taking on extra responsibility without being fully cognizant of all the risk factors. In this situation, performance consequences can arise because the increase in scope does not undergo the same rigor as occurred in the initial capacity planning and performance engineering work. For example, additional sources of data may be included, processing frequency may be increased, or additional groups of users may be added, which can cause an impact to performance. Being accommodative to change is important, but evaluating the impact of changes, including the impact to performance, is an important aspect of a mature change management program. Caution must be exercised to avoid "wishful thinking" that performance won't be a problem when capacity related changes are made to the system.

~~~

As we can see there are many potential causes for performance problems. Given these, it is fair to ask "What can be done to help prevent performance problems from occurring in the first place?" Just as importantly, we should also ask, "If I do have a performance problem, how do I go about diagnosing and correcting it?" The next two sections will address these questions.
~~~

Preventing Performance Problems From Occurring

It is interesting to observe that the act of preventing a problem from occurring means that you have just spent resources that are difficult to justify because it is not possible to prove that a performance problem would have occurred had you not have taken the preventative steps. This makes it easy to stop investing in proactive performance efforts once the performance problems go away. It is also interesting to observe that it is less "heroic" to take preventative steps than it is to step in and fix a performance problem once it is visible and in the critical path of the company's ability to function. This means that there needs to be strong organizational conviction that proactively addressing performance is important and strong organizational discipline to continue to do this over time.

A Cross Functional Performance Engineering Team

The establishment of a cross functional performance engineering team can go a long way to address and prevent performance problems before they arise. This team should span traditional organizational boundaries. The organizational groups mentioned earlier in this paper are in place for good reasons relative to performing their functional responsibilities, and trying to alter them or create yet another organization to address performance is not likely to be the most effective alternative. Instead, since performance is a cross-cutting concern, the solution should focus on a performance governance function. An analogy can be drawn to data governance programs which provide stewardship, quality improvement, prioritization, and policy setting within an enterprise data warehouse program. The data governance responsibilities are also a cross-cutting concern relative to the organizational structure and the governance team is assembled from key stakeholders and accountable parties amongst the organizations involved. While some data governance groups tend to be more centralized and others tend to be more de-centralized in the means by which they fulfill their charters, the salient point is that this group membership includes representation from the organizations involved in the data warehouse program. So the same approach can be used for creating a cross functional performance engineering team. Such a team can even be a sub group of a larger data governance structure, operating to fulfill its charter and reporting into the program governance steering committee.

The membership of the cross functional performance team needs to include data modeling and design representation, database administration representation, business intelligence representation, and ETL representation, along with other possible constituencies, as needed. There also needs to be a team leader.

During development, the team works pro-actively to identify and mitigate potential performance issues and to balance concerns of performance related to getting data into the warehouse versus getting data out of the warehouse. The team can also balance concerns between short term designs that can cause long term performance issues (for example "kill and fill" for ETL; overly "dimensional-ized" fact tables; lack of aggregation paradigms, etc.). These considerations need to span the major architectural tiers and not just include the database tier design and configuration, but also include the designs and configurations of the ETL tier and the business intelligence (BI) tier (supporting reporting, decision support, and analytics). Networking considerations between the tiers is also an important

consideration. The performance goal should be to achieve satisfactory overall (end to end) performance rather than optimize an individual tier, so the span of consideration needs to be all inclusive.

Once in production, the team remains in place but its composition should evolve to include a liaison to business operations to feed in observed performance issues encountered by end users. If performance issues arise, this team should follow the diagnostic protocol identified in the second part of this paper to determine root cause and correct the problem. Membership in the team can be rotational over time, thereby allowing the organizations to benefit by having a greater number of people exposed to the value of performance engineering.

For more information, refer to the document set *"Cross Functional Performance Engineering Teams For Data Warehouse Environments"* by End To End Information Solutions, which describes a detailed method for creating and managing a performance engineering team including: creating an engineering mindset; defining roles and responsibilities; establishing step by step tasks and deliverables; tips and techniques for best results; success measures; team leadership responsibilities; and team governance.

Diagnosing and Correcting Performance Problems

Once a performance problem exists, determining the root cause and how to correct it can be very time consuming. This is because the "symptoms" are often observed whereas the underlying "causes" are not necessarily evident. The key to efficiently solving problems is to take a structured diagnostic approach that allows you to quickly eliminate possibilities while also quickly identifying areas of likely cause. There are a variety of factors that influence the ability to perform such diagnostics effectively including the types of monitoring tools and historical performance information that is available, the capability of the people available to address the problem, the ability to be able to recreate the problem in an isolated environment, the accessibility of design information, and the topological complexity of the systems and networks involved within the scope of the data warehouse environment, to name a few.

Further complicating the process is that the steps that are necessary to remediate the performance problem can be very different depending on what is determined to be the root cause. Also, during the "journey" to discover the root cause there are a lot of constituents that may have convictions as to how to treat the "symptoms". A classic example of this is the axiom to "throw more hardware" at the problem, which, while in some cases can be needed, will also, in other cases, not improve the situation, depending on the root cause of the issue. In the absence of a disciplined, top-down diagnostic protocol when there are performance problems, there can also be significant "finger pointing" between organizations. My personal favorite is to always blame the network since this tends to garner the support of all the processing tiers (ETL, Database, BI) who otherwise may have to face the discomfort of learning that one of them is truly the root cause of the problem. When not handled effectively, it is possible for organizations to go on for a long time (months, or even longer) with a chronic underlying performance problem that never actually gets resolved.

The diagnostic protocol described below is based upon a problem resolution system that:

- First, determines the architectural tier that is causing the problem,
- Second, determines the type of problem the tier is experiencing,
- Third, goes through a system of diagnostic procedures for that problem type to determine root cause, and,
- Finally, based on the root cause, identifies the approach needed to correct the problem

By using this type of methodical progression an efficient path is taken to problem resolution. This protocol is reinforced by a structured documentation system that serves as a roadmap to identify the path you are on, the other paths that you have taken, and the paths that you have ruled out. Along each path is a series of diagnostic activities that are performed and the results are recorded. This allows for an integrated body of data to be collected as efficiently as possible and linked together for analysis. Problems that are difficult to diagnose can be more productively serviced through this approach by eliminating duplication of effort and redundant analyses. Problems that are relatively straightforward to diagnose are also easy to resolve using this protocol, with the added benefit of a well-documented resolution path, which can be very helpful for problems that re-occur over time. As this progression takes place, tool and process gaps are also identified to serve as a means of improvement for the organization moving forward.

The remainder of this document provides an overview of the multi-stage diagnostic protocol. For more information, refer to the document set *"A Multi-Stage Diagnostic Method For Correcting Performance Problems in Data Warehouse Environments"* by End To End Information Solutions, which provides the detailed methods, diagnostic tools, and decision framework for addressing and resolving performance problems.

The diagnostic steps to determine which tier(s) has a performance problem are based on allocating the portions of the end to end performance objective onto each tier and then monitoring each tier to determine if the performance requirements are being met. This includes being able to monitor the network interfaces between tiers as well. The three main tiers that are usually considered are:

Database Tier

The database tier is at the core of the warehouse and is a central point of processing. Database tier performance problems are probably the most common and can be the most difficult to fix. While database platforms are intended to be scalable, performance also depends upon the logical data design and physical implementation and the query usage patterns, so the overall complexity is high and it can be difficult to isolate the root cause without a systematic diagnosis process. If the database tier is identified as having a performance problem, then by going through the diagnostic steps for this tier, the problem will fall into one or more of the following categories:

Undersized database server – This can be the problem if initial capacity planning was inadequate or if the actual system load is more than what was planned for. The diagnostic steps to confirm or eliminate this as a problem are centered around establishing several performance benchmarks for defined usage patterns as well looking at individual workloads by themselves to baseline isolated performance.

Diagnostics are defined to measure the various resources that are managed by the server, including cpu, memory, and i/o. Also included are tests to isolate caching and buffering subsystems.

Physical database design / configuration – This can be the problem if the allocated resources in the database are insufficient (for example, logical memory, logical processors, sort areas, etc.) or if the application structures have physical design problems (for example missing indexes, poor partitioning strategy, inefficient file allocation, stale statistics, fragmented tables, etc.). The diagnostic steps to confirm or eliminate this as a problem are largely centered on comparisons to expected structures for similar class systems along with examination of query plans and query profiling.

Logical data model / Physical data model – This can be the problem if the data model is ineffective relative to the business questions the users are trying to answer. For example, poorly integrated or poorly conformed entities, overloaded attributes, mixed levels of grain, non-additive measures, lack of aggregate structures, poorly constructed primary keys, overloaded reference tables, poor choice of index types, etc. The diagnostic steps to confirm or eliminate this as a problem are largely centered on review of the logical and physical data design as compared to expected design patterns for similar type databases.

Query design – This can be the problem if the front end BI platforms and end users are attempting to use the database in a different manner than the business intended it for. The result is usually seen as queries that are much larger and/or much more complex relative to the design intent. Complicating this category could be an issue of changing business requirements over time and the lack of an effective change management system. The diagnostic steps to confirm or eliminate this as a problem are centered on query profiling relative to join patterns, amount of data scanned, amount of data processed, amount of data returned, types of joins, and observed complexity as compared to expected queries for similar types of databases.

Equipment malfunction – This is a special class of problem that is really a separate concern but needs to be eliminated as a possibility. If a redundant subsystem is impaired (for example a disk in an array, a network interface card, a disk controller, a cpu board, etc.) then the system performance may be significantly impaired and the solution is obvious. While operations monitoring is usually in place to detect and correct these problems right away, it is important to confirm the system is operating at full capability, otherwise a lot of time can be spent diagnosing the categories listed above when a piece of hardware simply needed to be replaced.

Business Intelligence (BI) Tier

The BI tier can vary greatly in terms of degree of sophistication and capability. Advanced capabilities could include significant data storage mechanisms and data processing features that are higher level integration points above the database tier. These can even federate across multiple databases. A performance issue observed by end users accessing a BI tier needs to first be diagnosed in terms of tier isolation, meaning is the problem below the BI tier and observable at the database tier or is the problem within the BI tier. Even if the problem is observed in the database tier, the cause could be related to the

design of the BI tier and the means by which it queries the database. If the BI tier is identified as having a performance problem, then by going through the diagnostic steps for this tier, the problem will fall into one or more of the following categories:

BI Platform capacity / configuration – This covers issues of server sizes, resource allocations for the supported applications, and capacity management.

BI meta-model design – This covers the BI meta-model that serves as an abstraction of the database to provide ease of use to report developers and end users performing analytics.

Report design – This covers the logical and physical report design approach focusing on the "alignment" between the goals of the report relative to the design of the BI model and underlying database tier.

Networking – This covers the network capacity and throughput between the BI tier and the database tier as well as the network capacity and throughput from the BI tier to the presentation tier or user desktop.

Equipment Malfunction – Same as described in the database tier section above.

ETL Tier

If the performance problem is related to loading data into the warehouse, then the ETL tier comes into consideration. ETL platforms and processing can also vary widely in their complexity and sophistication and can include various processing modules that provide specialized capabilities for cleansing, matching, de-duplicating, and relating data elements. If the ETL tier is identified as having a performance problem, then by going through the diagnostic steps for this tier, the problem will fall into one or more of the following categories:

ETL Platform capacity / configuration – This covers issues of server sizes, resource allocations for the supported applications, and capacity management.

ETL processing design – This covers the processing steps within the ETL jobs. It addresses concurrency, network i/o, task dependency, job partitioning, task allocation to tier, processing algorithms, upstream dependencies, etc.

Networking – This covers the network capacity and throughput between the ETL tier and the database tier as well as the network capacity and throughput from the ETL tier to the sources of data.

Equipment Malfunction – Same as described in the database tier section above.

~~~

In all three tiers, for each of the categories identified above there are specific diagnostic procedures that are followed to help pinpoint the root cause of the problem. These procedures use a variety of tests, tools, decision paths, patterns, prior knowledge bases, and analytical techniques. Once the problem is identified, there are best practices identified that suggest the approach to take to correct it. A
~~~

documentation record is also built into the process to serve as an ongoing reference should problems re-occur or other similar problems present themselves.

Conclusion

Since data warehouse performance problems can be complicated, it can be time consuming and expensive to identify the root cause and correct the problem. The best approach is to prevent performance problems from occurring in the first place. If they do occur, an efficient method for diagnosing and correcting the problems can be extremely valuable. This paper outlines an approach for preventing performance problems based on establishing a cross functional performance engineering team which operates during the development phase of the data warehouse build out and then continues to operate once the warehouse is in production. This paper also outlines an approach for diagnosing and correcting performance problems if they do occur, based on a systematic diagnostic protocol that can be implemented by the performance engineering team. Detailed document sets are available from End To End Information Solutions for both topics, entitled:

"Cross Functional Performance Engineering Teams For Data Warehouse Environments"

"A Multi-Stage Diagnostic Method For Correcting Performance Problems in Data Warehouse Environments"

About End To End Information Solutions

End To End Information Solutions (E2E) specializes in information technology solutions, architecture and design for data warehouses and distributed systems including large scale data management and service oriented architecture projects, application and project assessments, project planning, solution architecture, data architecture, performance tuning and project execution and delivery. Contact us at 904 612 1124.
