DBG Software

Real Performance…It's About Time

SQL Performance

Performance Modes

Background

SQL performance can be performed in two modes; proactive and reactive.

In a proactive mode, the objective is to take action before users or resources feel the impact of poor performance. Proactive SQL performance analysis can yield a **profound return on investment**.

In a reactive mode, either a user is complaining or resources are being compromised as a result of poor performance and the objective of performance analysis is to **mitigate the problem as soon as possible**. Often these scenarios can present an opportunity for someone to "hit a home run" when fixing the issue, but this creates a false sense of accomplishment in a sense. A shop should never be in a constant state of being reactive ("don't fix it until it's broke!!"). This can have **disastrous effects** on end user perception, hardware resource usage and accurate capacity forecasting.

Before going into specifics regarding reactive and proactive analysis, there is one important point to stress. That is, to a large degree, the efficiency of analysis in either mode is dependent on tools to **identify the top SQL resource consumers** along with intuitive performance demographics (CPU, Wait times, Locking, etc.).

Mode Descriptions

It's important to emphasize that **proactive SQL performance analysis** can occur in the development, system test (sometimes called benchmark), and production phases.

In the **development phase** proactive mode, the responsibilities of today's DBA are so extensive, that regular one-on-one SQL reviews during development are a thing of the distant past. However, analysis in the development phase can be very effective if the developer(s) concentrate the execution of their process in an isolated database. This should not take place in a database where there is routine development, but rather in a database after the final functional form of the SQL has been created. This way, **an SQL performance reporting tool** can report on the most resource intensive SQL's executed in that isolated database during a given timeframe and analysis would occur in a top-down fashion. This **development phase performance strategy** not only makes the best use of human resources, but also ensures measures are taken to ensure optimal SQL performance before code is released to system test or production. The usual limitation in the development phase is the analysis is at an adhoc level and no indication of how that SQL will perform under load can be observed.

In the **system test phase (benchmark)** proactive mode, stressing the SQL with load is the key. An SQL performance reporting tool can report resource intensive SQL's in the same fashion as in the development phase. Often, SQL that performed just fine in development **will not perform well under load in benchmark**. Also, ensure the benchmark test has a transactional load similar to that of production. SQL performance recommendations can be tested in an iterative fashion in benchmark until end user time Service Level Agreement (SLA) thresholds are met, resource utilizations are acceptable, etc. Remember, this is likely the last true performance analysis that will occur before migration to production, so be diligent about performing sufficient tests to assess the incremental benefit of SQL performance tactics so that end user times and resource utilizations are the best possible (not just meeting expectations).

In the **production phase** proactive mode, mining for SQL performance opportunities and resolving them before end users or hardware resources are noticeably affected is the objective. A regular (usually weekly) report should be generated from an SQL performance monitoring tool that reports resource intensive SQL's in the same fashion as in the development and system test phases. The difference would be the tool should also be proficient at alerting on database SLA breaches, alerting on mission critical SQL execution plan changes, and trending SQL performance metrics.

True **reactive SQL performance analysis** can really only occur in production. An end user complains or an alert is triggered because CPU is elevated on a resource are examples. Having an SQL performance reporting tool that can monitor 24/7 and quickly isolate the issue is critical.

Success Factors

Regardless of the SQL life cycle phase or analysis mode, there are certain common success factors to keep in mind:

- Notice in all phases and modes above, a tool was used for SQL offender identification. 3rd party tools generally do a much better job than database vendor options.
- Always focus on the best execution plan during SQL analysis, not the performance metrics. The most optimal plan will always yield the best performance metrics.
- Have a good method of tracking performance recommendations, usually by assigning them a performance ID and keeping on top of their implementation status
- For each analysis effort, keep an analysis document and a report. Retaining the analysis is good for reference and shows all experiments tested to arrive at the final solution. Also good for knowledge transfer.
- Create and maintain a database specifically for tuning.
- Keep this skill set visible within your company. Create a regular (usually monthly) high level report showing value to upper management.

© DBG 2010 Unauthorized reproduction or distribution of this document, or any portion of it, may result in severe civil and criminal penalties