**How Database Monitoring Can Boost Your Performance**

Anyone who is responsible for **database performance** knows how demanding and challenging database performance tuning is when managing a database. One of the critical functions of this process – database monitoring – is often overlooked. Database monitoring includes identifying the right SQL for tuning, determining right way to tune and whether SQL is right thing to tune.

## Database Tuning and Database Monitoring Go Hand in Hand

If you check your **database management software** dashboard and emails one morning and realize that your database applications have slowed down, do you think it will be the right time to do database monitoring? No. Instead, you need to tune the performance. But without monitoring the applications for a time period, how can we tune?

By monitoring the performance, you get historical data that helps you identify what you need to tune and how to correctly tune it. Not only is tuning easier, it’s also effective when you do regular **database monitoring** which gives a comprehensive overview of database metrics. Furthermore, it helps avert problems before they impact your business.

Performance management has three parts:

**Monitoring** – tools for collecting performance data are put in place from each area of database environment.

**Analysis** – accumulated performance data is analyzed regularly to check patterns in workloads, resource consumption and business cycles.

**Tuning** – this involves making changes whenever there is a need to, instead of doing it when you think you have to.

## Finding Right Tuning Area

**Database monitoring** is multi-layered, and requires you to monitor on four levels i.e. the SQL level, infrastructure level, the database/instance level and the user/session level. At each of these levels, a problem can arise. Hence, for each of these four levels, performance tuning is different.

For instance, by adding storage capacity, you may be able to address issues in **configuration management database** but they may still exist at the SQL level. Similarly, by creating the correct non-clustered index, you can address a problem at SQL level, but if it exists at infrastructure level, it may not get resolved.

For effective smart **database monitoring**, it is best to build up a series of snapshots of performance over a period of time. It is advisable to write scripts creating monitoring table and collect timely statistics about the database, to further attach them to the table.

Database monitoring involves using numerous scripts for capturing and storing **wait time data** for further analysis. We can also fill a monitoring table with metrics such as wait events, logical reads, row counts, and locked objects. Applying these metrics can allow you to set alerts about trends like low memory and insufficient disk space.

Once the statistics collection process is automated and the tables have aged, it’s a good time to analyze their historical performance. Although it may seem untimely to check for problems in the **log database** when it’s performing smoothly, still, the best time to do that is before a database can start having issues.

The reason for this is that it saves you a lot of hassle which you will face otherwise. When your **database is up and running**, you have all the time to analyze and tune it without rebooting the server or having to take the database offline.

## Requirements of Database Monitoring

In order to use database monitoring to boost your performance, consideration of these requirements is very important.

**Transactional Workload**

Applications are linked to the database by transactions. In order to report the performance of the database, the only way is to measure the database **transactional workload** (which includes batch jobs, the number of users, the automated jobs, updates, inserts, deletes, small result-set selects, etc.)

**Coverage**

Performance of a database depends on the operating system resources, storage system and virtual machine. If one has not been **monitoring and analyzing all these levels** and the problems are existing there, it is likely for them to begin their troubleshooting in the wrong place.

**Time Trend Analysis**

The more time frame you have for **analyzing trends**, the better insights you get. A long horizon of data tells us what is normal for the environment, which applications are least/most active, when are the instances at higher volume and when they are backed up.

**Overheads**

Studying a database fundamentally affects its behavior. For a better performance, keep the overheads low to make sure that your collection cost does not have any affect on the conclusions. It is also advisable so that it doesn’t exceed the value of the collected data. To avoid overheads, create a backup **SQL backup database** script.

**Granularity**

The rate of daily, weekly or hourly collection determines the **granularity level** with which we can determine potential threats to the performance. When the volume of transaction changes or when you are looking into a problem with SQL server use database, it is best to plan on increasing the sampling frequency.

**Workable Outputs**

Some common database performance issues include long wait times, logical reads, too many/few indexes, and index fragmentation. SQL tuning often breaks these **log database** jams. However, database monitoring will also ensure whether you need to tune SQL in the first place or not, and whether there is a need to backup SQL database server.

One needs to ask questions like, is the operating system patched? Are the virtual machines under the database functioning properly? Is maintenance required by the **storage subsystems**? The outputs you receive from the analysis should help determine not only the problem but also which part needs to be addressed.

**Changing Alerts**

Alerts and warnings must be able to change according to the **transaction volume**, time, resource capacity and the conditions under which the business functions. If the threshold on alerts and warnings does not change, the monitoring tools will not be as useful as they should be as you create SQL database.

**Virtual Metric’s Database Monitoring**

Today, organizations depend on intuitive database monitoring to optimize their applications that are critical to the business. Whereas most **database monitoring tools** notify when there is a performance issue, an inclusive monitoring tool also provides detailed insight into the root causes and troubleshoots them.

Virtual Metric’s [database monitor](https://www.virtualmetric.com/database-monitoring) is preferred by thousands of admins for an overall **database performance management** and for ensuring uninterrupted service delivery. It provides powerful Microsoft SQL server and backup database SQL server monitoring with proactive alerts, log analysis and advanced database analytics.