DBA GUIDE

SQLServer Database Tuning & Optimization

TABLE OF CONTENTS

Introduction 1

Slow running queries and hanging applications 2

Optimization Tip #1: Check you Index fragmentation level! 3

Optimization Tip #2: Find the missing indexes 6

Optimization Tip #3: Check for blocking sessions 8

What is blocking? 8

Optimization Tip #4: Check your Logical Reads 14

Alert! Alert! Alert! 16

Conclusion 20

About the Author 21

About dbWatch 22

This document is intended for DBAs who are working on tuning and optimizing their databases.

INTRODUCTION

Being a *DBA* (database administrator) is a tough job; as a DBA, you oversee every aspect of database monitoring, management,
data manipulation and transformation, ensuring the integrity and accessibility of your data and securing your data from unauthorized access.

Not only that, but maybe you are also responsible for the database infrastructure planning, high availability planning, cluster configurations, database server configurations, patch management, backup, and maintenance management. You also work on data migration, database optimization, and database tuning.

Sometimes you are also involved with developing queries to support the application team; you have so many tasks to do yet so little time.

To be an efficient and effective DBA, you need to know what to prioritize, and a good DBA ensures that he/she has complete control of all the databases he/she is managing.

As a DBA, you will greatly benefit from specialized tools to get your job done efficiently, assist you in automating and simplifying your daily routine tasks, and help you focus on the more important and demanding tasks. The more instances you manage, the more you will need good tools to monitor status and health, automate routine tasks and streamline your workflow.

This e-book will focus on optimizing your databases and the key areas you should concentrate on when tuning your queries.

dbWatch Control Center is the newest database farm monitoring and management solution offering by dbWatch – designed for efficient proactive monitoring of database farms in medium and large enterprises.
dbWatch Enterprise Manager is the older product from dbWatch which focuses more on an instance centric approach when monitoring databases while *Control Center* focuses on a farm centric approach to monitoring the database farm as a whole, across on-premise, cloud or hybrid and cross-platform environments.

In this document, the author will be using both *dbWatch Enterprise Manager* 12.8.3 and *dbWatch Control Center* in the different examples. When we mention *dbWatch* only, it refers to functionality found in both products.

Slow running queries and hanging applications

When an application problem unexpectedly takes you by surprise, as a DBA, what do you usually do?

a. *Blame the application developers on their inefficient/unoptimized code*
b. *Raise your voice and tell your manager that it is not your fault and walk out of the room*
c. *Investigate the stored procedure, function, query that is being executed by the application and provide a solution on how to improve and fix it so that it won't happen again*

If you answered *a)* you may be right (developers does not always write efficient queries), but it is still your job to fix it, or b) I might suggest that you rethink your life decisions carefully as your rash decisions may result in you being in great trouble.

Kidding aside, when you hear this from your manager, you would first investigate by opening your management studio for SQL Server; you use your Dynamic Management Views (DMV's) to diagnose performance issues and investigate the execution plan to verify whether the query is executing optimally.

When you are optimizing query performance such as this, you need to ask yourself the following:

- Is the query executed efficiently?
- Does the query have any missing indexes?
- Does the query select unnecessary data?
- Does the query select columns that are not needed?
- Are there any blocking sessions that made the application hang?
- Are the indexes heavily fragmented? As this may cause queries to execute slowly

Those are some of the questions you should focus on when identifying and pinpointing the main culprit of the reported issue.

Optimization Tip # 1: Check your Index Fragmentation Level!

Heavily fragmented indexes can degrade query performance because additional I/O is required to locate data to which the index points. As a rule of thumb, more I/O causes your application to respond more slowly, especially when index or table scan operations are involved.

In the examples below, you will see how to determine which instances within your databases are heavily fragmented and need an index rebuild or reorganize.

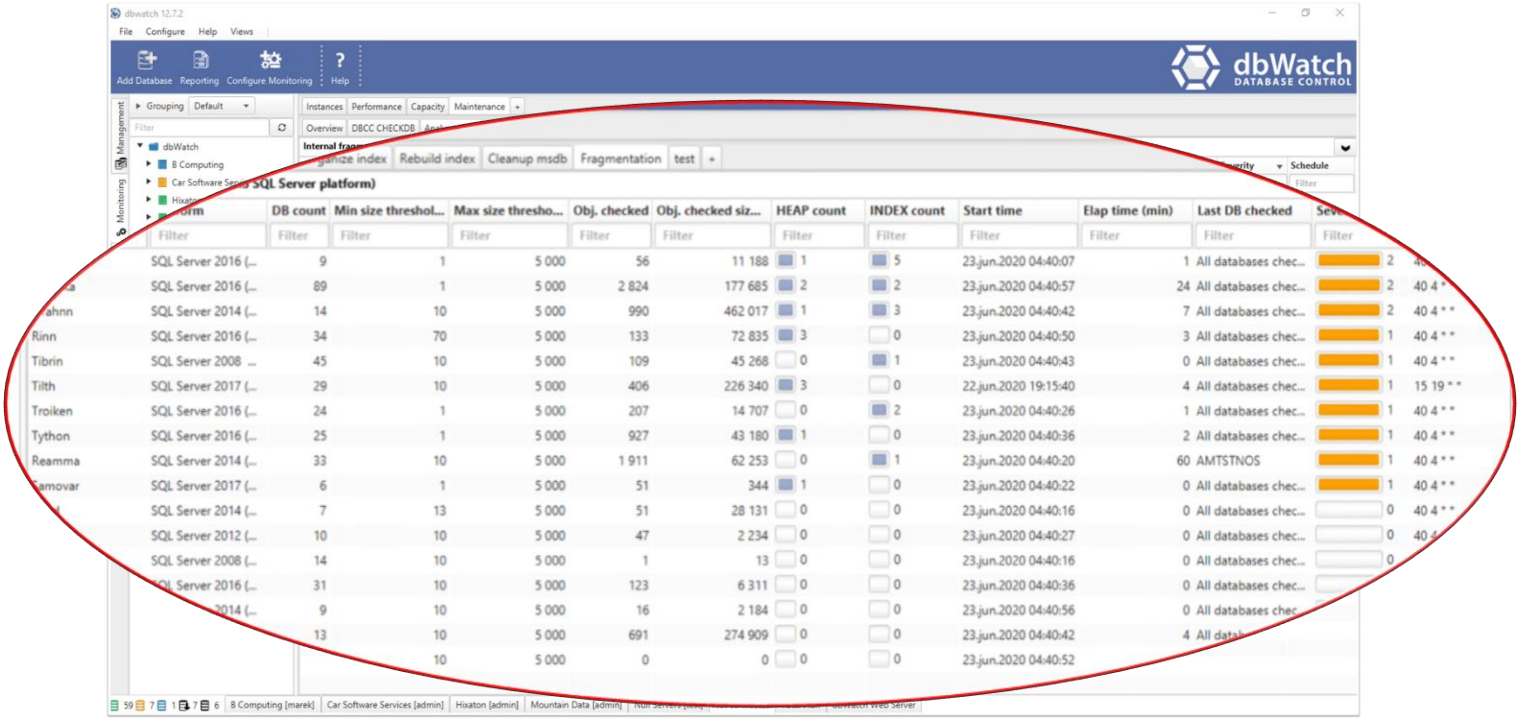

Figure 1. dbWatch Enterprise Manager instance overview

Using *dbWatch Enterprise Manager*, select the *Monitoring* tab on the left. Choose *Maintenance* and click the *Fragmentation* tab.

This overview will see the internal fragmentation statistics, database count, # of the index, the last DB checked, and severity level.
Internal fragmentation statistics show your instance name's information, while the last DB checked refers to the previously checked database within that instance.

Another example is using *dbWatch Control Center*, selecting the management module, and deep dive into the database where you encountered an issue; Right-click on *Indexes* and choose *Show fragmented indexes*.

This overview shows in-depth information on each fragmented index inside your database. In addition, it shows you the fragmentation percentage of each index in this specific database. The summary also indicates what type of index it is, whether it is a clustered or a non-clustered index.

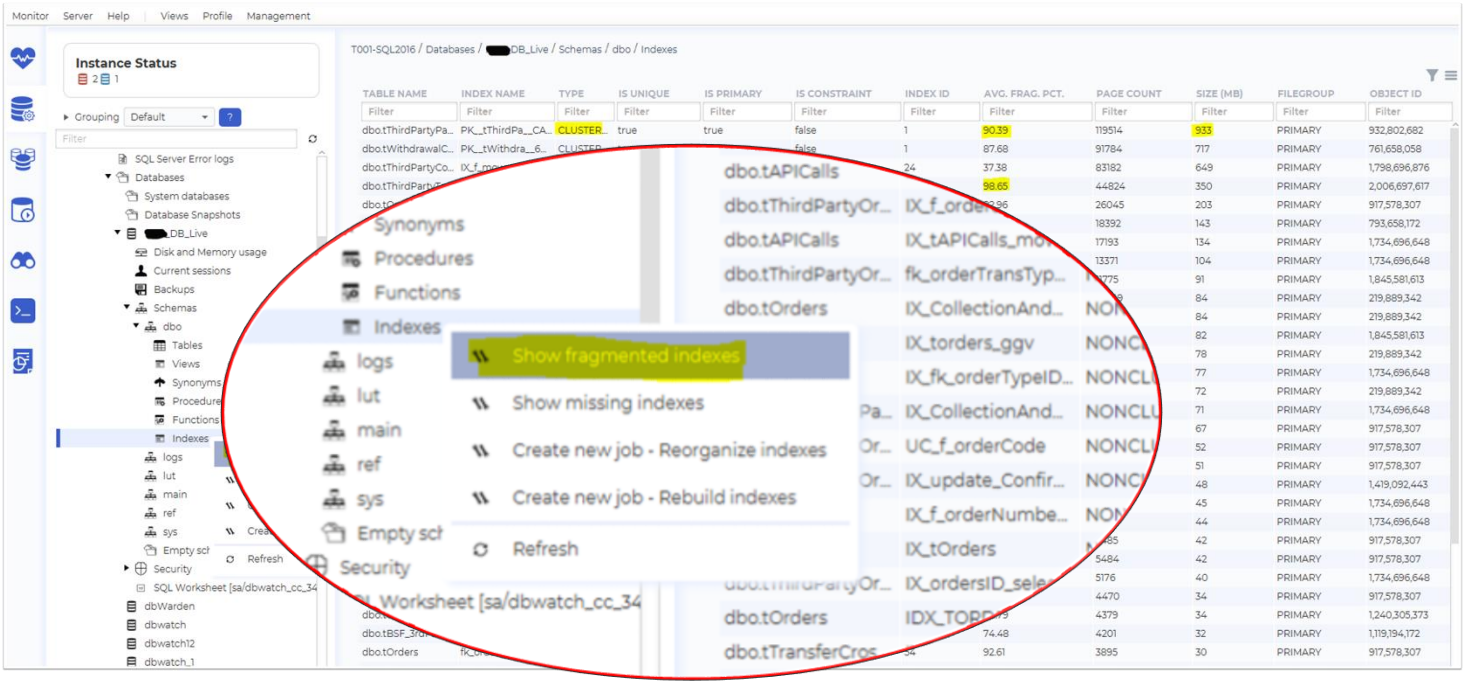

Figure 2. dbWatch Control Center database Fragmentation level overview

If you right-click on an *index*, you can perform different operations such as drop index, move index, index rebuild/reorganize, recreate the index, update the stale statistics, view the index definition, and more information.

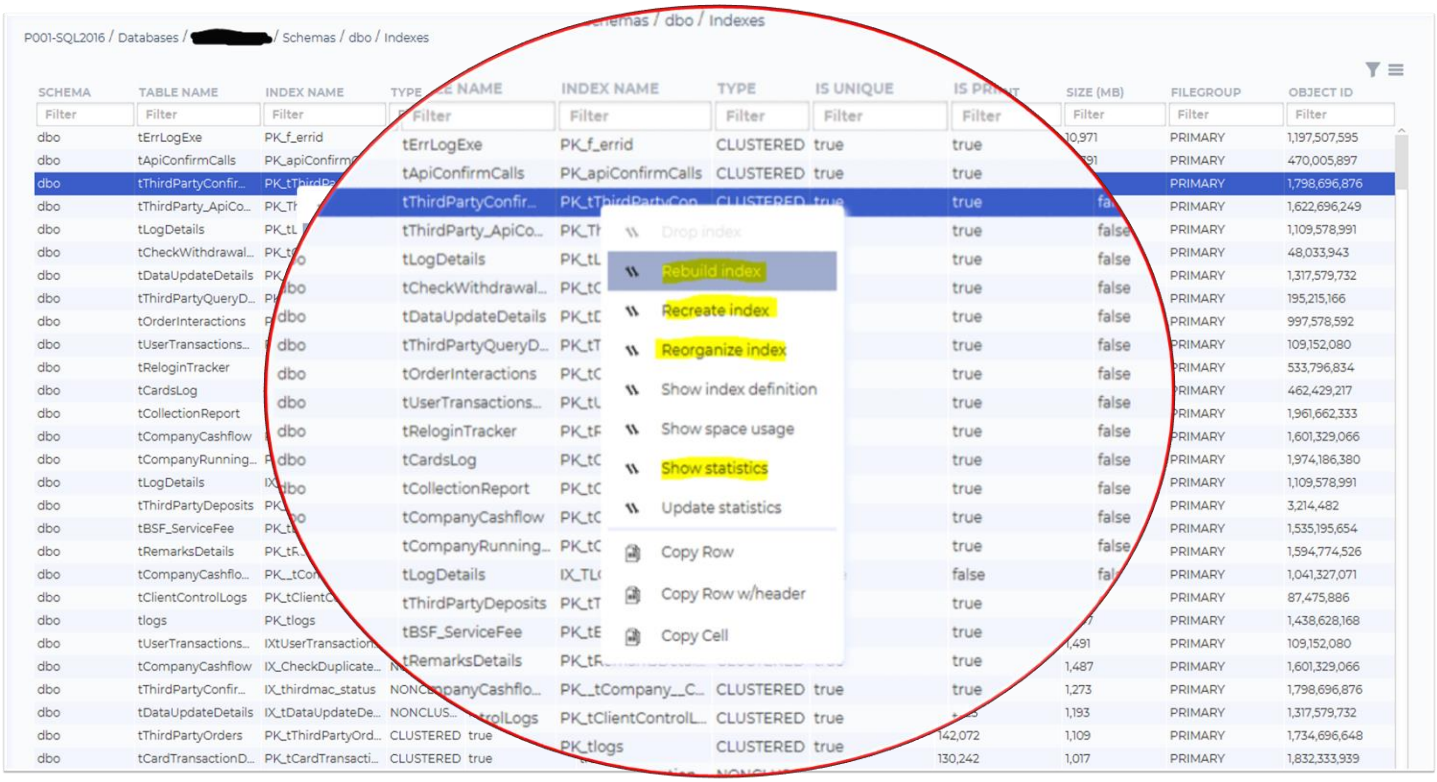

Figure 3. dbWatch Control Center database Fragmentation level overview

To configure them, you can go to either the monitoring module or farm module and select the maintenance job you want to configure. Right-click on it and choose “Configure”. You will see the parameters for the maintenance job as seen in Figure 3.

Optimization Tip # 2: Find the missing indexes

In Microsoft SQL Server's execution plan, when you execute a query, it assists you in identifying if there is a missing index to help improve your T-sql query performance.

An example below shows the management studio's execution plan suggesting creating this missing index.

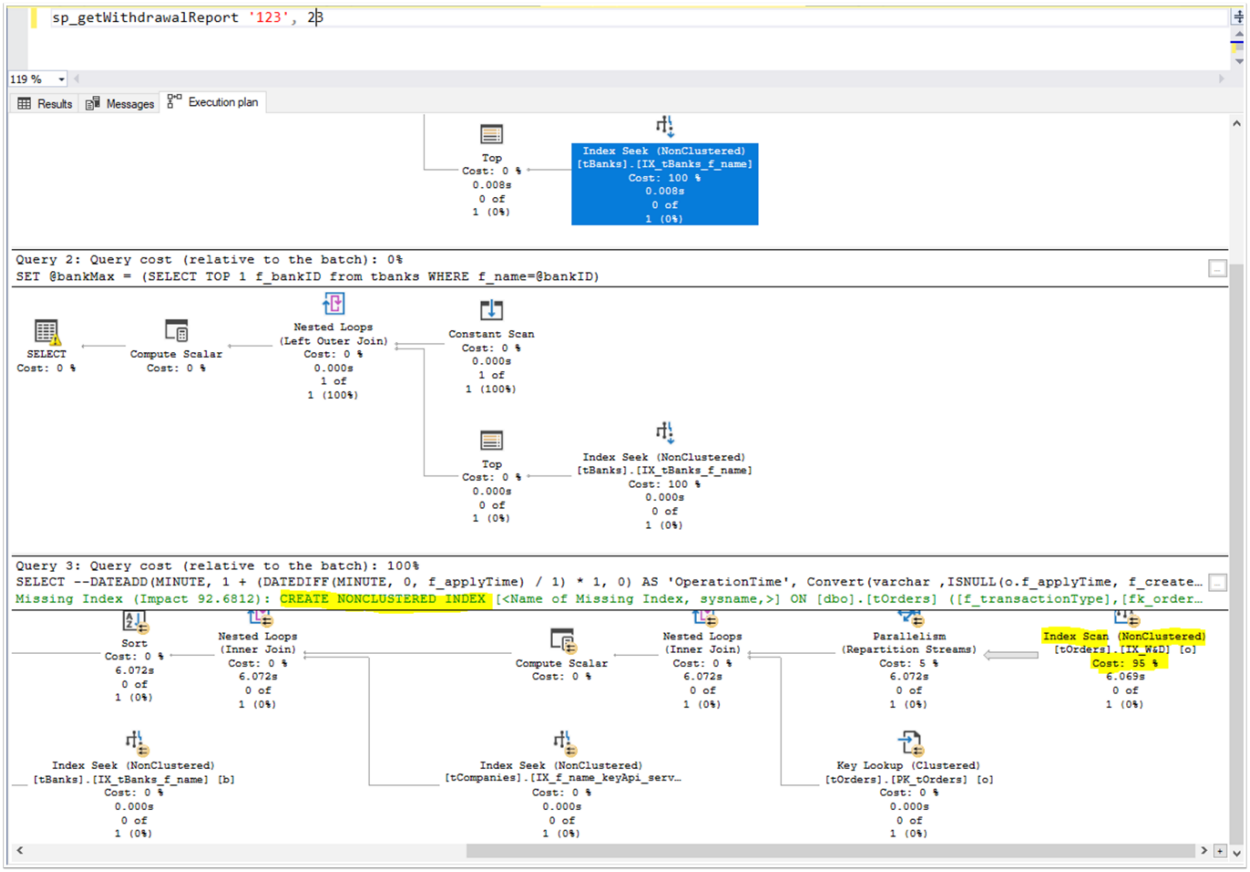

Figure 4. SQL Server Management Studio's Execution Plan view

In creating these suggested missing indexes, you should be careful of the tradeoffs as they might affect insert and update queries. You must be familiar with the workload you are running in this database. You need to assess if creating these suggested indexes will affect other queries as the indexes may slow them down.

In dbWatch, you can see all the missing indexes in your selected database. dbWatch helps you determine which index recommendation is suitable for your case. The overview displays the average impact improvement in percentage when you create the suggested index, the table name, the total size, the number of rows, the create statement for the index, and the proposed index name.

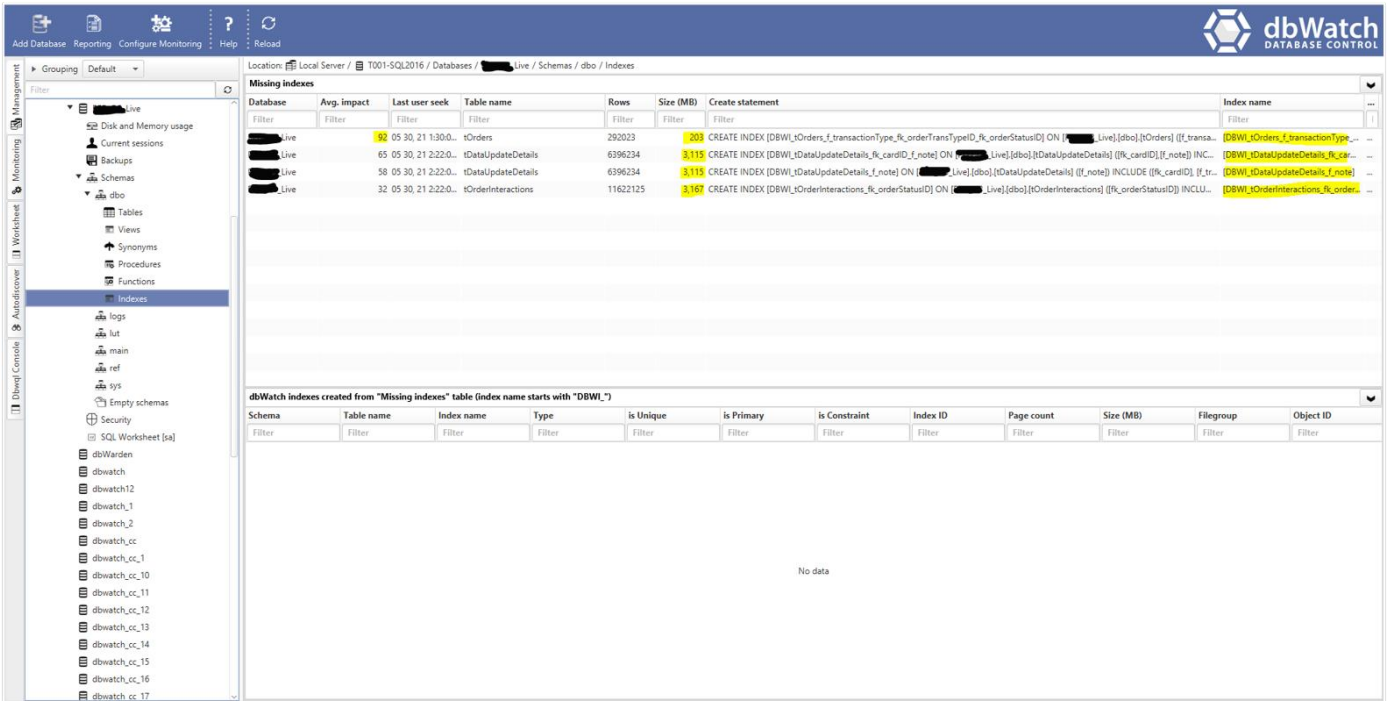

Figure 5. SQL Server Management Studio's Execution Plan view

In the example below, if you right-click on an *index*, you can view the whole query text or execute the create index on the table.

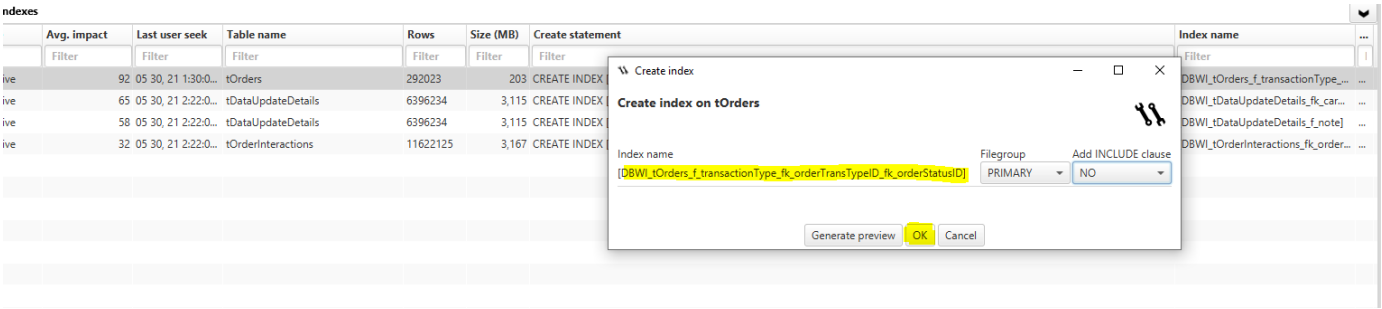

Figure 6. SQL Server Management Studio's Execution Plan view

Having good overviews will help you assess the performance situation of your database servers.

As a DBA, it is up to you to create the recommended indexes for your databases; you need to ensure that the correct indexes are in place.
Ensure that the recommended indexes are based on the workload running in your databases without affecting other important/critical queries.

Optimization Tip # 3: Check for blocking sessions

What is blocking?

As per Microsoft, "*Blocking is an unavoidable and by-design characteristic of any relational database management system (RDBMS) with lock-based concurrency. In SQL Server, Blocking occurs when one session holds a lock on a specific resource and a second session attempts to acquire a conflicting lock type on the same resource.*

Typically, the time frame for which the first SPID locks the resource is small. When the owning session releases the lock, the second connection is then free to acquire its own lock on the resource and continue processing. Blocking as described here is normal behavior and may happen many times throughout the course of a day with no noticeable effect on system performance."

Blocking and Locking is a mechanism designed to maintain the integrity and consistency of your data in a database. The goal of the blocking mechanism is to prevent dirty reads of data in a database.

When is blocking/locking a problem?

In an enterprise with multiple users/transactions from left and right, users try to access a busy database to extract data for their tasks.
Given an example:

User 1 tries to update a column in table A with millions of rows; in this scenario, User 1 locks the resource table A. At the same time, User 2 is executing a stored procedure used to select data from table A and generate a report. User 1 is performing an update transaction, and it is taking a while to update all the records; the lock is still on, and User 2 keeps on waiting for User 1's transaction to finish and release the lock. Finally, after a long time of waiting User 2 reaches its limit resulting in an application time out.

While more users try to access and select data from resource table A, User 1 update transaction is still not finished, resulting in multiple blocking chains.
It affects your database performance and results in application timeouts that are not good and become a massive business problem.

A DBA should be proactive when handling these kinds of blocking problems; you need to put the proper alerts in place to be aware of what's happening in your database at any given moment.

It would be best if you were informed which queries are blocking one another for you to implement a plan that will reduce the long-running execution of those queries blocking one another to avoid it from happening again.

In dbWatch Control Center's management module, if you go to select your database instance> expand the instance > click on *performance* under blocking sessions. You will be able to see all blocking sessions and the blocked sessions inside your database.

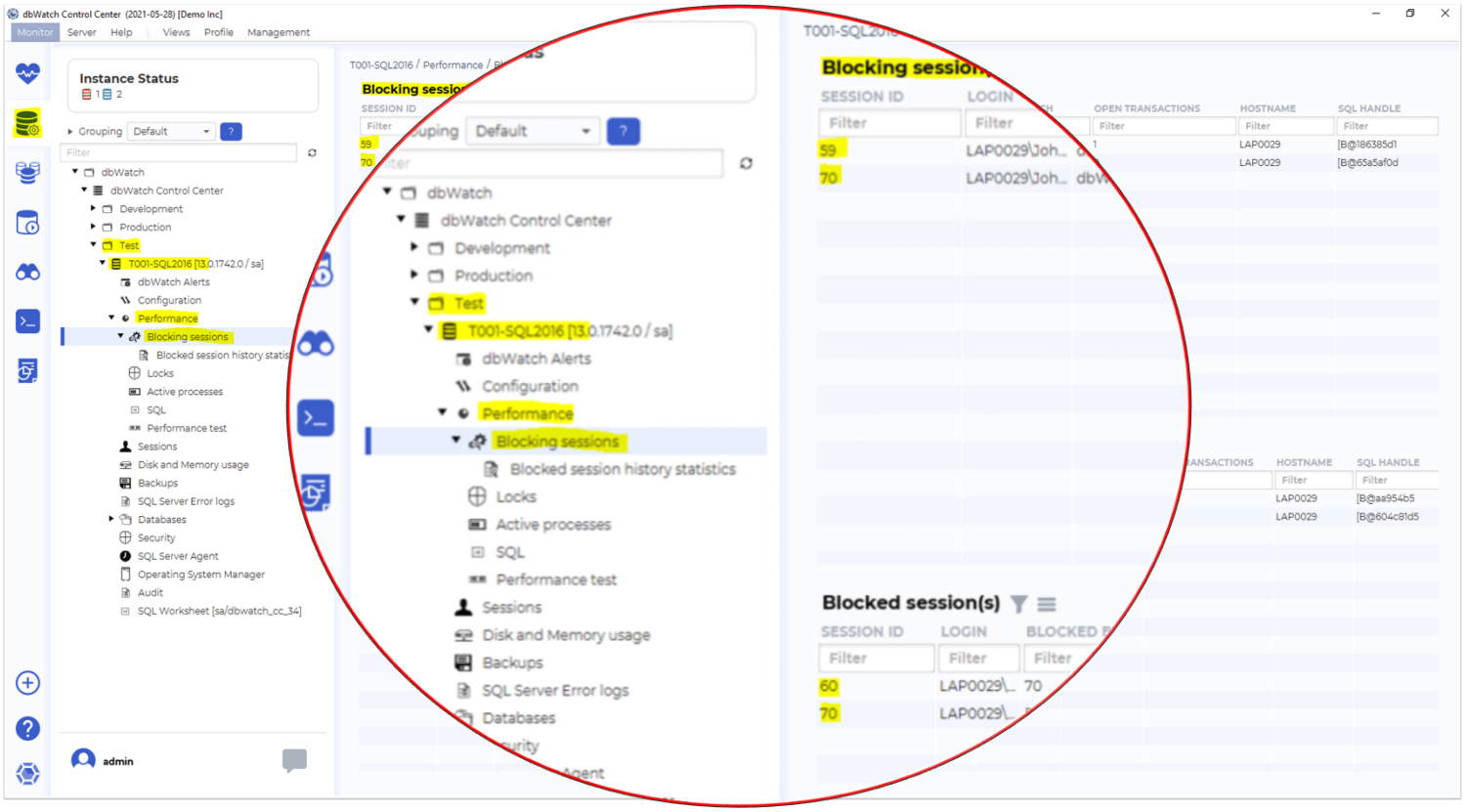

Figure 7. dbWatch Control Center blocking sessions overview

If you right-click on a *session*, you have an option to kill the session or display the sessions' SQL query.

This view helps a lot when it comes to troubleshooting blocking scenarios; it identifies which resources are being locked now by a specific session.

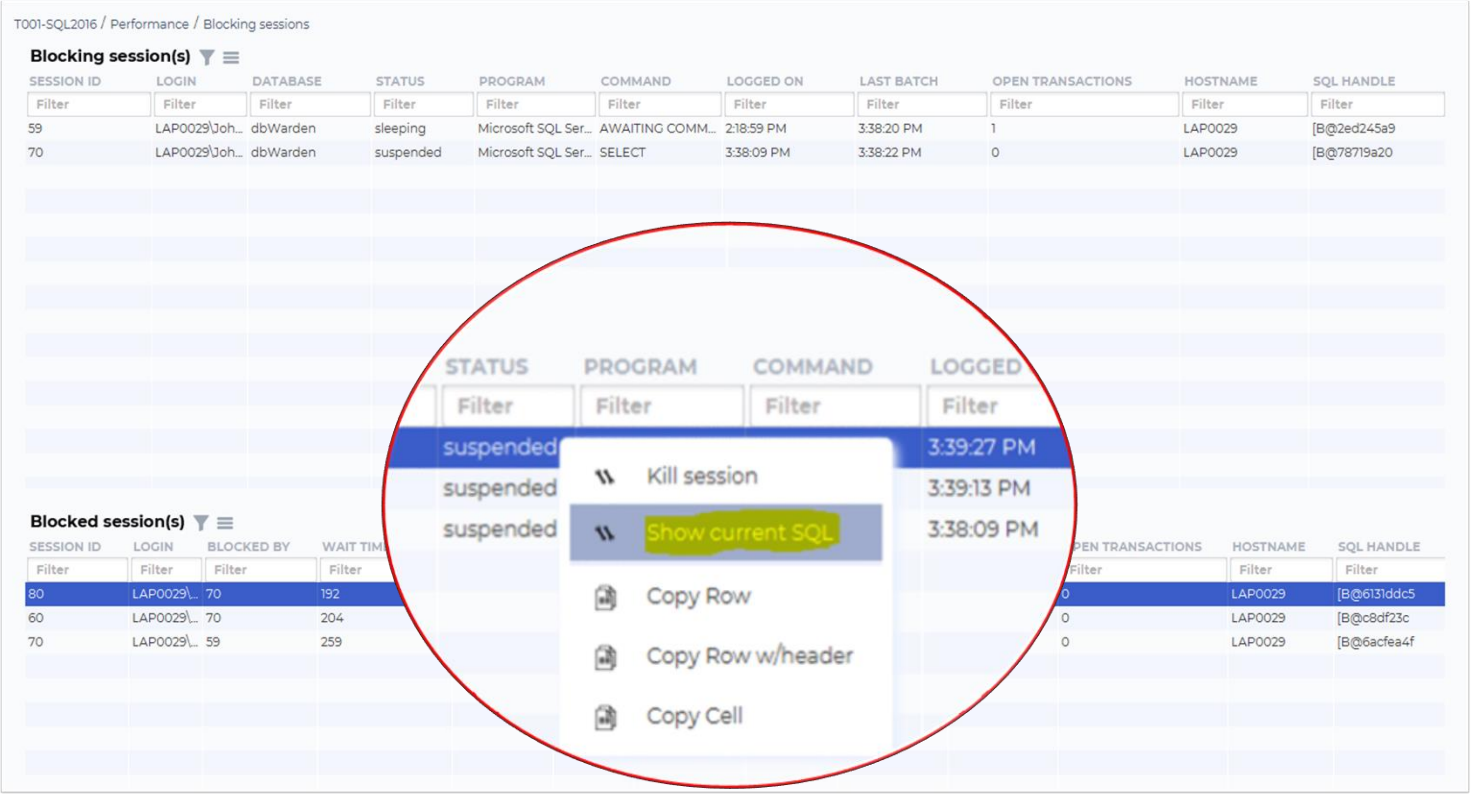

Figure 7.1 dbWatch Control Center blocking sessions overview

The view below shows the SQL query that is currently blocked. Enables you to fix and optimize your code to avoid these concurrent blocking sessions that affect your database performance from happening again.

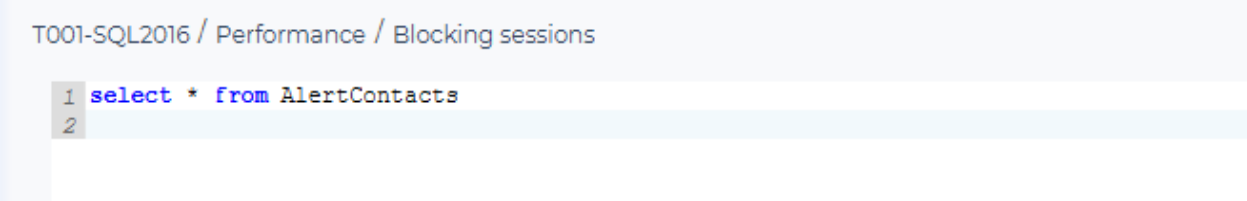

Figure 7.2 dbWatch Control Center the blocked session current SQL query

In the next view, you can see a job named blocking statistics reporting an ongoing blocking session in the *Control Center* monitoring dashboard, which you can see in the *detail's* column.

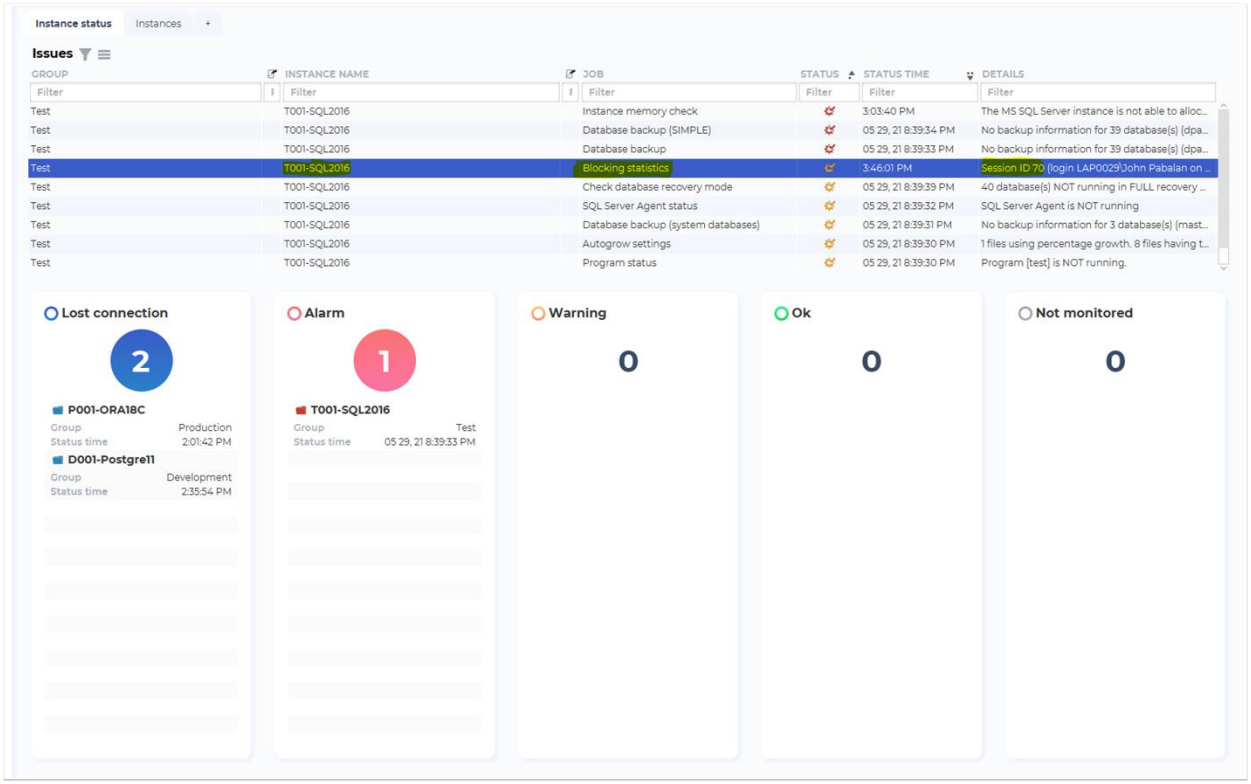

Figure 8. dbWatch Control Center Monitoring dashboard alerts an ongoing blocking session

If you right-click on the *job*, you have options to execute it to retrieve the latest statistics and details to view more information about it. A *configure* option is available to set an alarm threshold when you wish for dbWatch to notify you if the blocking threshold. Choose *Details* to find out more information about the alert.

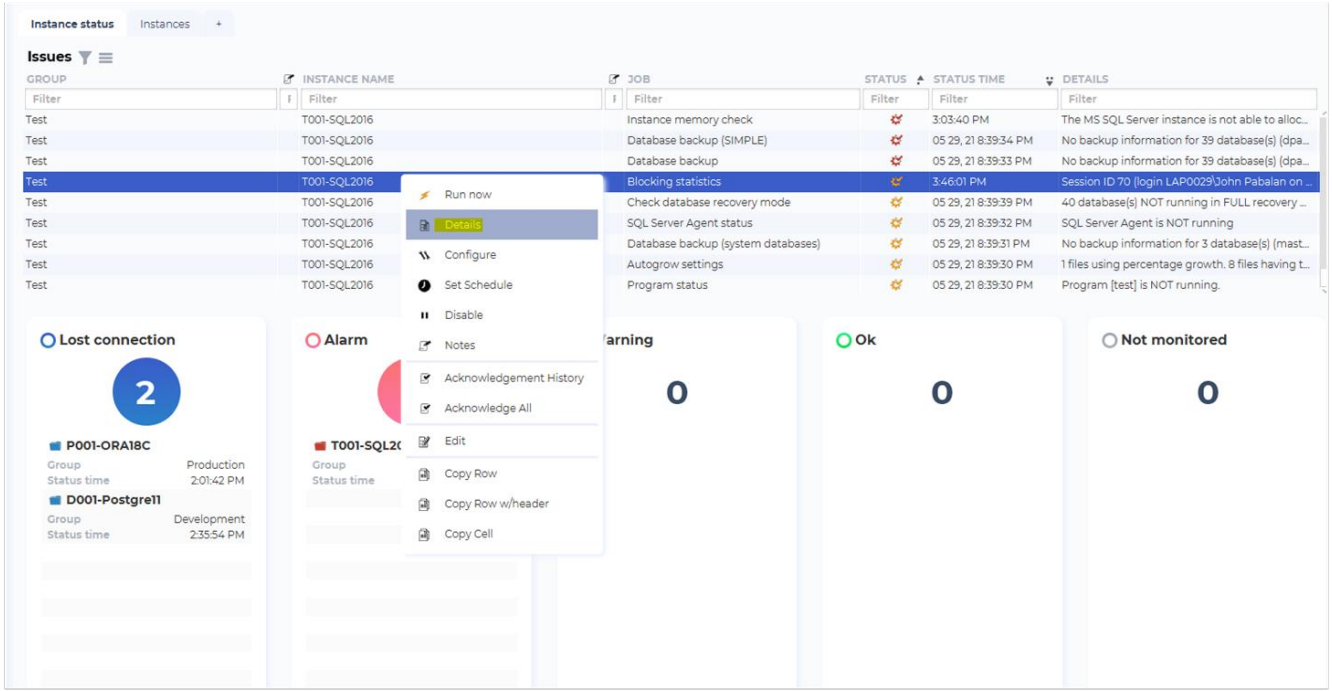

Figure 8.1 dbWatch Control Center Monitoring dashboard alerts an ongoing blocking session

In figure 8.2 and 8.3, you can see more information about the session id, which database is the blocking session happening, which user session is causing it, and many more information.

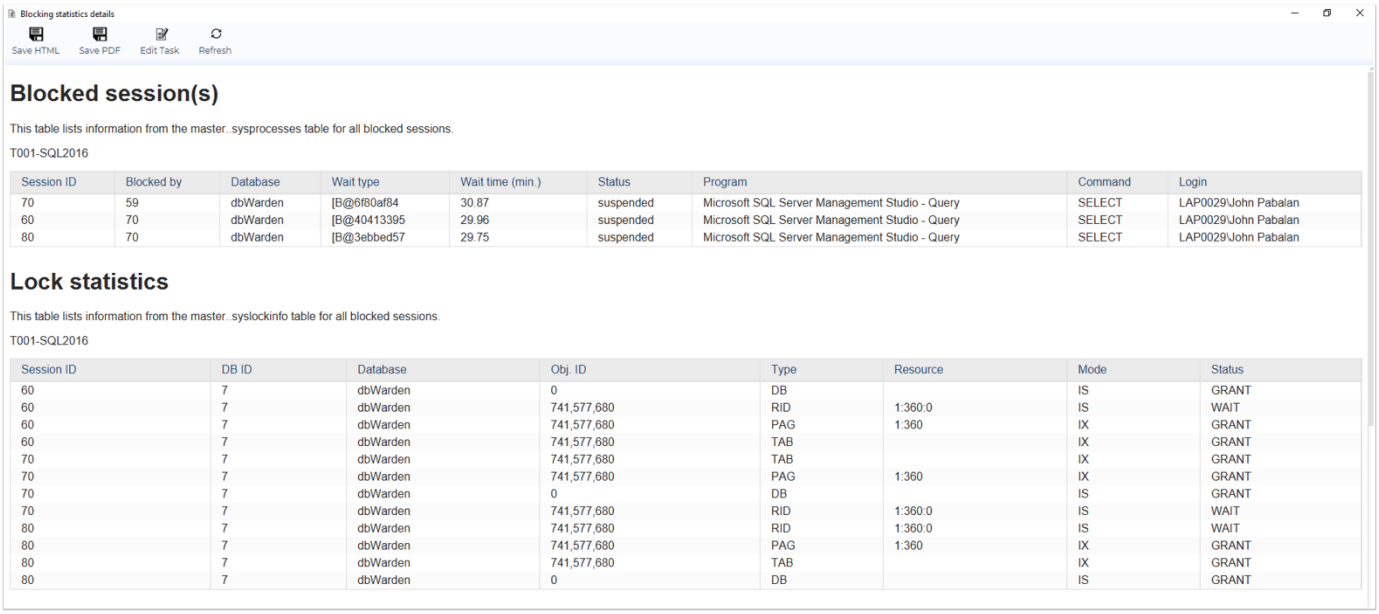

Figure 8.2 dbWatch Control Center blocking detailed report

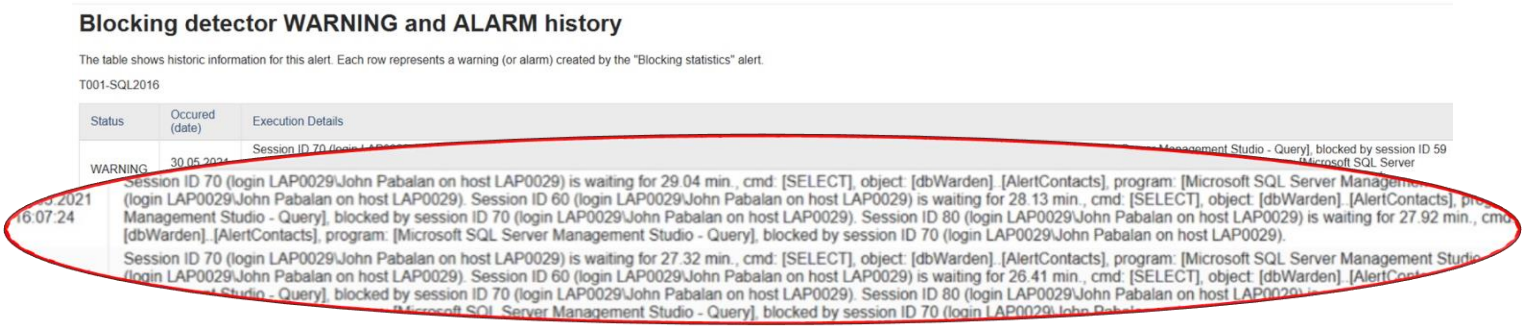

Figure 8.3 dbWatch Control Center blocking detailed report

Upon opening the history details, you will see a detailed blocking report. For example, in Figure 8.3, it shows that Session ID 70 – name LAP0029 / Chad Pablan is waiting for approximately 29.04 minutes. Simply put, the login credential's query *"SELECT * FROM [dbWarden].[AlertContacts]"* in MS SQL is being blocked by session ID 50 of the same name. What's interesting here is the detailed history is separated by entries scheduled for every 2 minutes. In each cell, you can see the updates for every minute.

Having a complete overview and insight into the performance of your database is critical. It is a best practice for you as a DBA to be notified when a blocking turns into multiple blocking chains that degrade your database performance.

Optimization Tip # 4: Check your Logical Reads

Logical reads is one of the most critical performance metrics. Logical read occurs when a database engine requests a page from the buffer cache(memory). In the overview below, you can see your instance name and the key columns like current logical reads and HR% Avg (Average hit ration). Ideally, you want this value to be as close to 100 as possible).

Afterward, you need to check what queries consume excessive memory as this is a sign of a memory bottleneck.

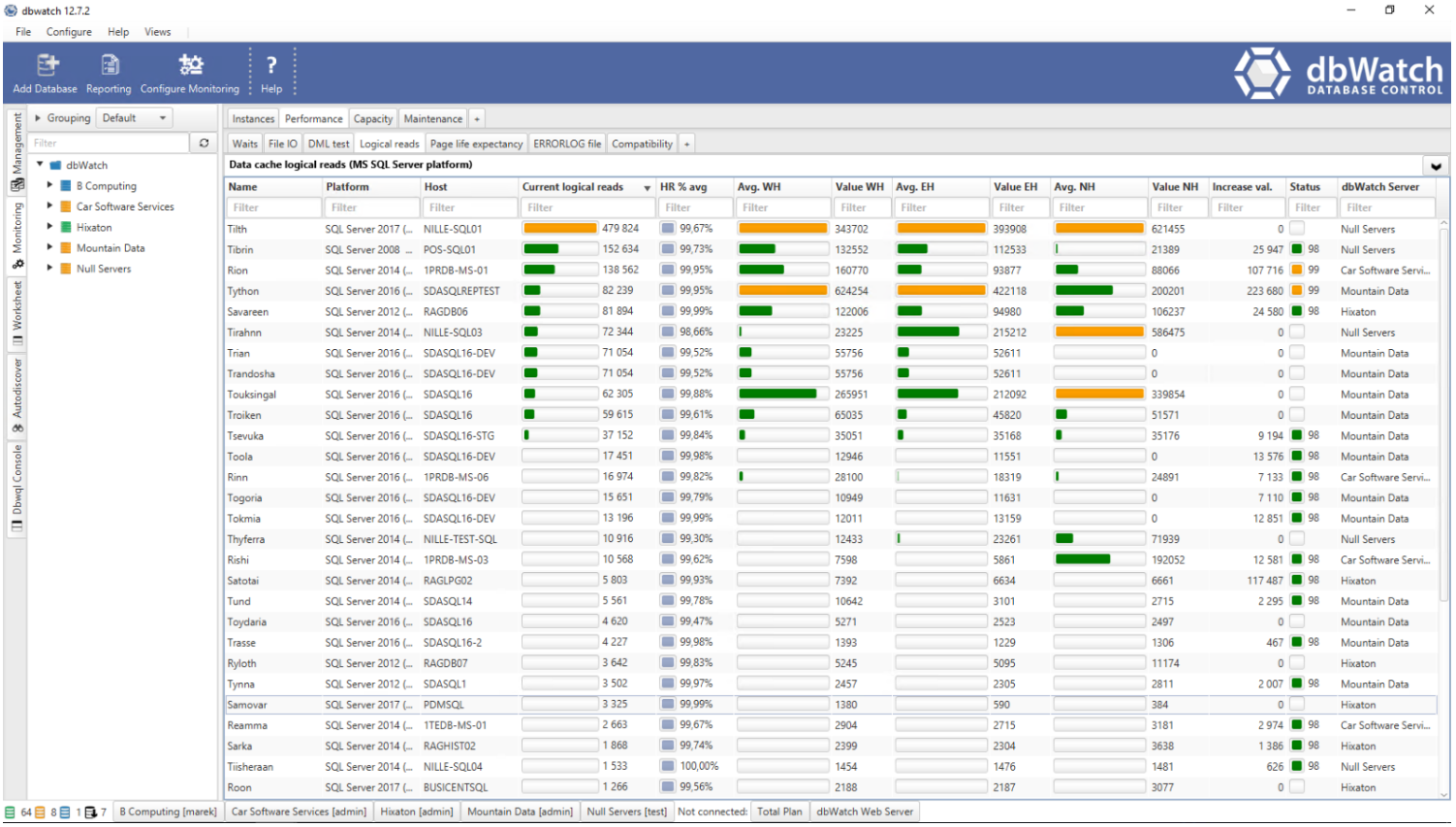

Figure 9. Data cache overview – dbWatch Enterprise Manager

This example displays the logical reads history and the cache hit ratio history.

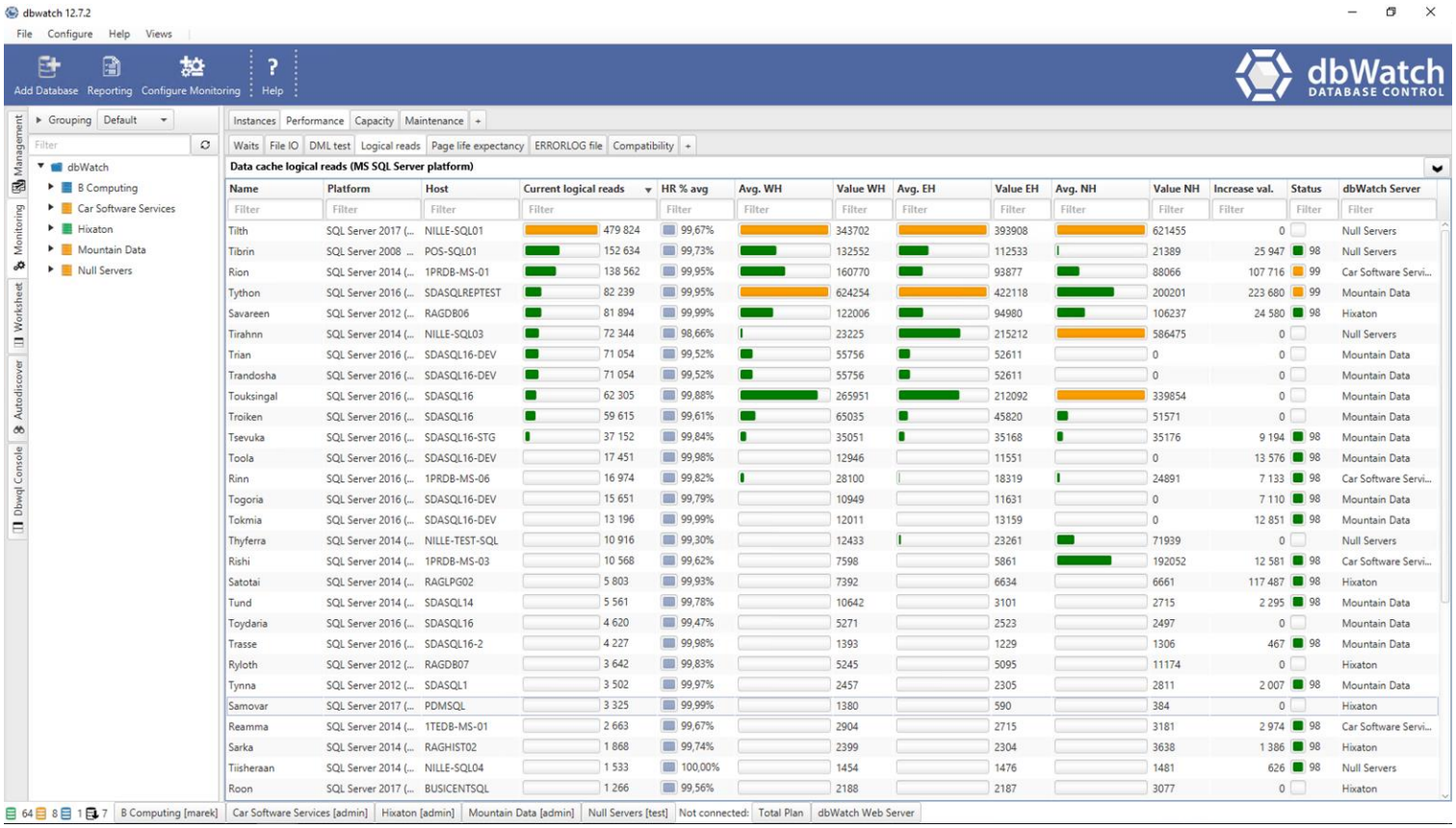

Figure 10. Data cache report

As time progresses, your table data grows. Eventually, it will reach a point in surpassing your memory threshold. To avoid this problem, you need to plan to partition your data so that it will not instantly deplete your machine's memory upon reading huge tables. Avoiding loading unnecessary data into your machine's memory is the primary challenge as this causes high logical reads.

Alert! Alert! Alert!

Have you ever experienced waking up in the middle of the night because a colleague is calling you regarding an issue with the production database? Is this the usual notification system that you expect when a problem arises with your databases?

As DBAs, you need to monitor and manage our database's health. With various processes relying on the databases you manage, you need complete control and the overview necessary to keep track of your database performance and health.

Alerts provide a great way of notifying you about the status of your database environment. dbWatch Enterprise Manager provides a variety of extensions to configure how you want to set up your notification ecosystem.

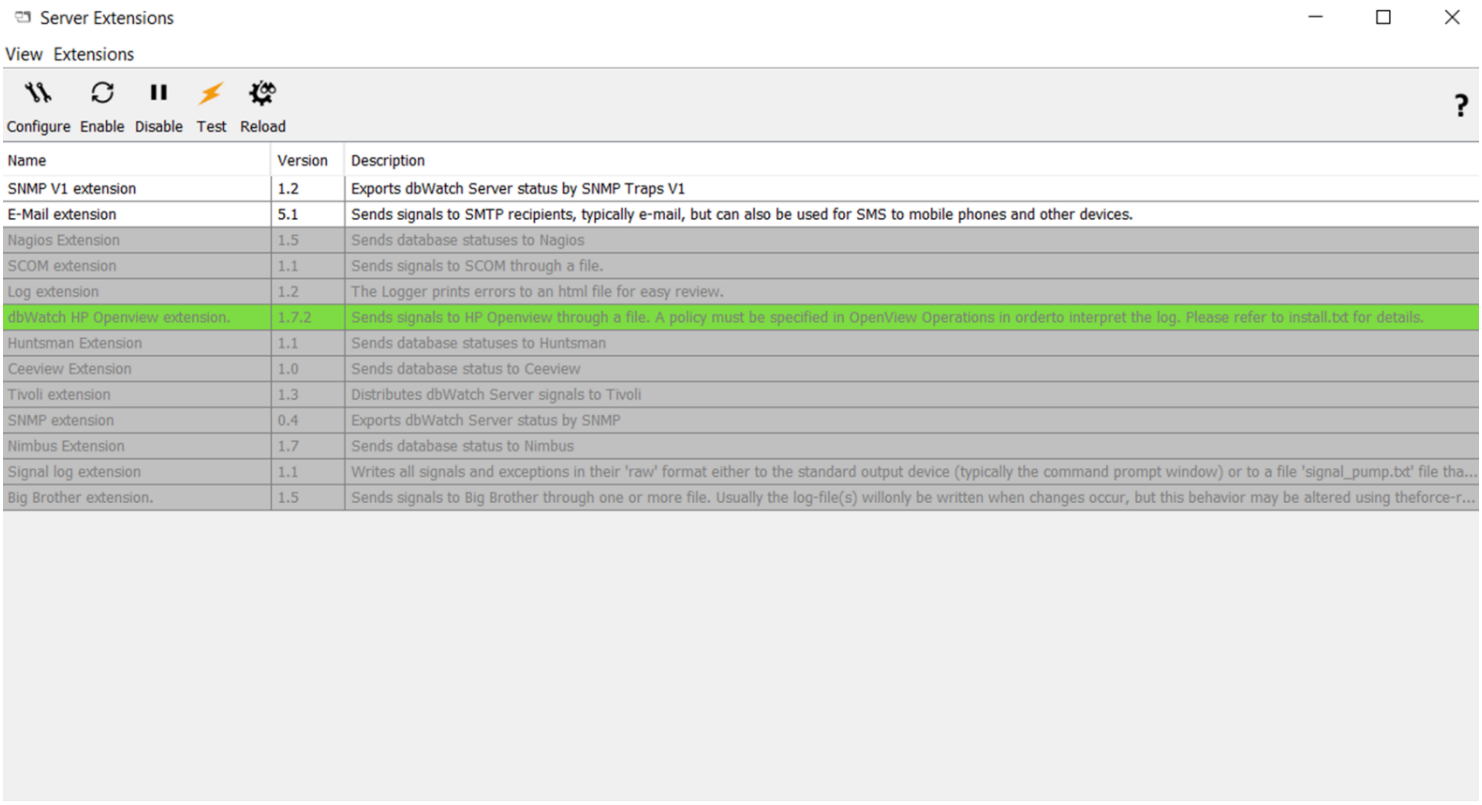

Figure 11. Opening Server Extensions for dbWatch Enterprise Manager 12.8

You just need to enable which alerts you will need by hover to “Configure” and clicking “Extension”. Right click on the extension you will need and select from one of the options presented.

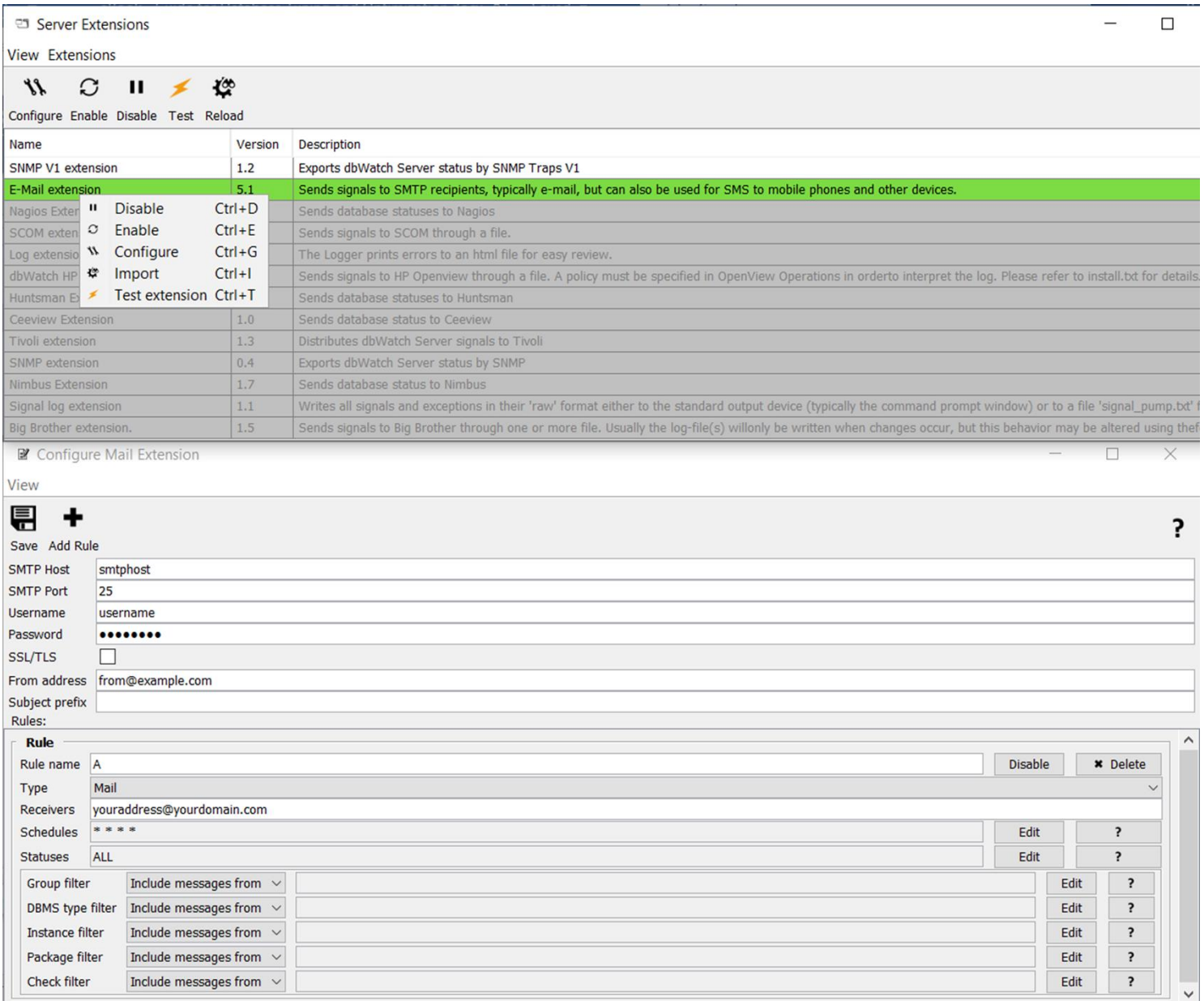

Figure 12. Right clicking on one of the extensions and selecting Configure on Mail Extension

By default, all the extensions are grayed out which means they are disabled upon installation. As an example, you can enable the mail extensions then configure it afterwards to capture blocking sessions.

In the screenshot below, the mail alert is configured to notify the manager when dbWatch detects a blocking session happening in the Production environment of all database platforms and when it reaches the maximum threshold.

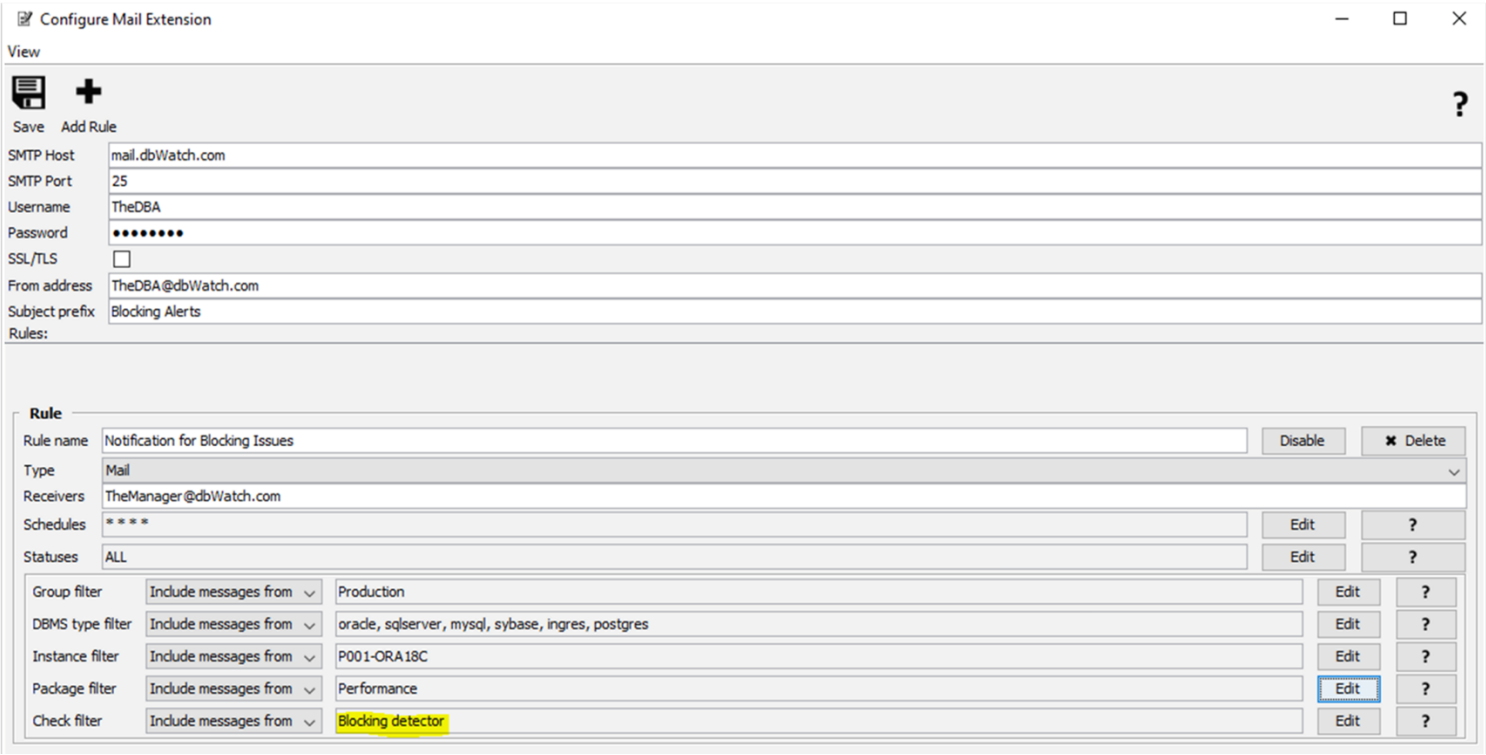

Figure 13. Configure dbWatch Enterprise Manager 12 Mail Extension for Alerts

Alternatively, you can schedule a report by clicking on “Report” and a Report Wizard will help you generate an overview report. Select one of the many templates available below.

Complete List of Report Templates available

Database Availability Statistics	SQL Server Version information	Health check Report for Oracle 18c
Database Backup report	Health check Report for Oracle 10g	Health check Report for Oracle 19c
Database Environment Report	Health check Report for Oracle 11g	Health Check Report for MYSQL
Database Information	Health check Report for Oracle 12c	Health Check Report for Postgres
Uptime Statistics	Health check Report for Oracle 18c	Sybase Report -version 12

Health check report for MS SQL Server	Health check Report for Oracle 19c	Sybase Report -version 15
Health check report for MS SQL Server 2000	Health check Report for Oracle 8i	
Health check report for MS SQL Server 2005	Health check Report for Oracle 9i	
MS SQL Server backups	Health check Report for Oracle 10g	
Oracle global license report	Health check Report for Oracle 11g	
RAC Statistics	Health check Report for Oracle 12c	

Afterwards, schedule your report as seen in Figure 15. Your scheduling preference can either be “Cron schedule” or “Interval schedule”. “Cron scheduling” is handy when you want to generate a report at a specific schedule. For example, you want to receive the report every Friday at 1500H. While “Interval schedule” is runs on a set interval such as every 15 minutes. When all is ready, input the recipient’s email addresses and email subject, and provide the report information you want – PDF or HTML in either landscape or portrait orientation. You are done configuring your report.

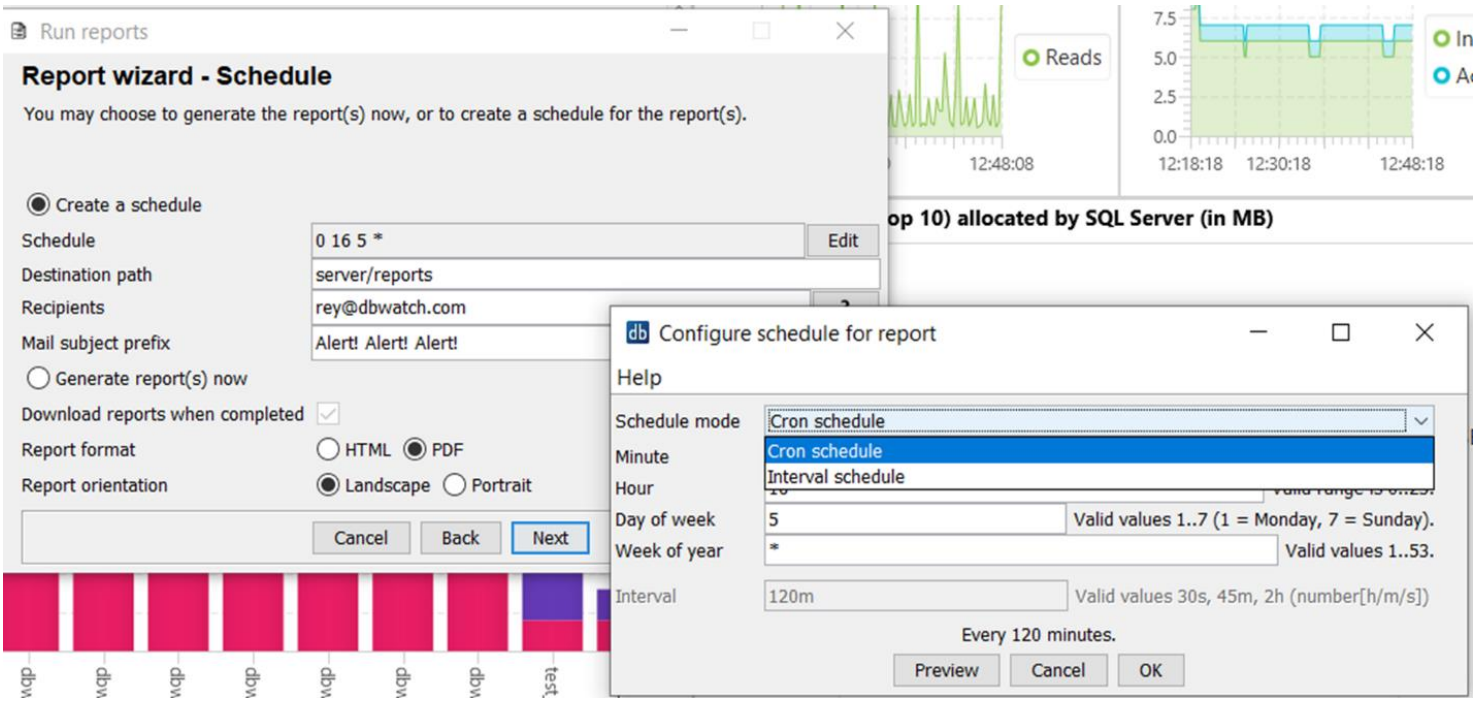

Figure 14. Scheduling a Report with dbWatch Enterprise Manager 12.8

As a DBA, you must be proactive with you approach. The best way to be a proactive DBA is to set up good alerting mechanisms, so we are notified early to any issues that need your attention before it becomes a critical problem.

CONCLUSION

In this e-book, we demonstrated a few optimizations tips for you to be effective and efficient in optimizing your query performance; you need to remember these things:

- **Check your Index fragmentation level!**
  - You need to identify which indexes are heavily fragmented as it will affect your query performance

- **Find the missing indexes!**
  - Identify any missing indexes that may improve your query performance. But beware, you should create those recommended indexes based on your workload

- **Check for Blocking sessions!**
  - Always remember to set up alerts for you to be notified if a blocking chain is building up

- **Check your logical reads.**
  - As per Microsoft, "a logical read is when the query engine needs to read data. First, it looks in memory. If the page is already in SQL Server's memory, then it uses that. If it can't find it in memory, that triggers a physical read, and the data page is read from disk. A logical read without a subsequent physical read is a "cache hit," basically."

  So, you need to keep track of your queries having high physical reads rather than logical reads.
  Ideally, you would like your cache hit ratio to be around 80 to 100%, which means most or all your data will be read from memory, which is much faster than being read directly from physical disks.

- **Perform Routine Checks in your database**
  - With dbWatch's automated management tools, you can perform database analysis and database checks to both your hardware and software. With just a click of a button, rebuild, reorganize, analyze indexes, and check databases in your database instance.
  - dbWatch also offers safeguards for automatic performance optimization. It prevents users and the system from running these resource-intensive procedures during your work hours.
  - Take advantage of these performance tuning features for preventive maintenance,

- **Don't forget the Alerts!**
  - It is a best practice to put the right alerts in place for every critical performance metric that you wish to keep an eye on

ABOUT THE AUTHOR

Chad Pabalan is a Pre-Sales Engineer for dbWatch and a DBA specializing in SQL Server high availability setups and disaster recovery planning and configurations. He is an AWS Certified Solutions Architect Professional, a cloud enthusiast specializing in architecture and designing scalable, high available, and fault-tolerant systems on AWS Cloud.

If you have any questions, comments, and suggestions or you would like to know more on how dbWatch can assist you in your current enterprise database monitoring and management situation, feel free to contact: **sales@dbwatch.com**
For more information, visit **www.dbWatch.com** or the **dbWatch wiki pages**

ABOUT dbWatch

dbWatch was founded in 2001 by leading database experts in Oslo, Norway.

From the beginning, we have developed and delivered our database monitoring solution to customers in Scandinavia, Europe and North America. Combining deep DBA experience with top software developer talent has allowed us to create a complete and optimal solution for monitoring and managing large enterprise database server farms.

If you have any questions regarding our solutions, contact:

Andreas Hope
andreas@dbwatch.com

dbWatch AS
Kongens Gat 15 0152 Oslo, Norway
+47 22331420 (OSL Office)
www.dbwatch.com

© 2021 dbWatch AS. ALL RIGHTS RESERVED.

This guide contains proprietary information protected by copyright. The software described in this guide is furnished under a software license or nondisclosure agreement. This software may be used or copied only in accordance with the terms of the applicable agreement. No part of this guide may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording without the written permission of dbWatch AS.