An Introduction to
Workload Tuning

A White Paper by Ami Levin
September, 2010

"The concept is interesting and well-formed, but in order to earn better than a 'C', the idea must be feasible."

-- A Yale University management professor in response to Fred Smith's paper proposing a reliable overnight delivery service. Smith went on to found FedEx.

> www.dbsophic.com

Contents

Introduction

Workload Tuning is just what it sounds like: tuning all processes that comprise a database workload in one go, balancing the effect of the tuning across all processes.

Until recently, this was not possible. Now it is.

The Workload Tuning system has already been used to optimize over 200 production database workloads worldwide. It has achieved an average workload-wide performance improvement of over 60%, including databases that had already been finely tuned using standard tuning techniques and tools. Moreover, the precise size of the improvement on any organization's workload can be accurately predicted in advance of the changes being migrated to production. This is possible because Workload Tuning provides highly accurate before-and-after benchmark measurements for each and every process in a controlled, non-production environment.

This paper will describe how Workload Tuning works in detail, based on Qure™ for SQL Server, the first commercially available Workload Tuning tool. This paper will also compare Workload Tuning to the traditional Query-by-Query Tuning method.

Database Performance Management

Performance of Information Technology systems has long been a critical issue for most organizations. In the last few decades, IT has become the life blood of virtually all organizations, enabling process efficiencies and service levels that were practically impossible without it. This has resulted in ever growing demands on IT to deliver data in increased volumes, with increased relational complexity, and with increased speed, all while living within decreased budgets.

One natural outgrowth of these challenges is the evolution of automated software solutions designed to assist the expert IT staff. Workload Tuning is possible precisely because of such advances in automation, enabling DBAs to deliver even higher levels of database performance, while reaping significant IT savings.

Workload Tuning will be described shortly. But first, we should review the characteristics of traditional Query Tuning. This will provide the background for why the arrival of Workload Tuning is now so imperative.

Query Tuning – the Trustworthy Standard

> ***"A computer once beat me at chess,***
> ***but it was no match for me at kick boxing."***
>
> *-- Emo Philips*

Database applications start their life cycle in a highly optimized state. Often, over time, they lose that razor's edge. This can be due to any of a number of natural evolutionary changes in the application, the business or the data. Enhancements are made to the application, placing unbalanced strains on existing data structures. The volume mixture of data and queries shifts as business needs shift, adding more contention to the workload. Large data ETL jobs are scheduled to earlier or later times. New reporting needs arise. You can fill in the rest. There is always some new influence that alters the nature of the workload and creates new bottlenecks. When this happens, some means of performance diagnosis and repair is necessary.

Traditionally, the challenge of improving the ever-shifting performance of database applications has been addressed primarily by identifying and optimizing the top resource-consuming processes and queries (affectionately known as hogs). In this section, we'll review the mechanics of this approach, and then we'll consider its pros and cons.

How Query Tuning Works

As you are well aware, the traditional Query Tuning process looks something like this:

Figure 1 - Traditional Tuning Process

1. **Continually Monitor Production**
   Monitoring may be performed actively or passively. If performed actively, a monitoring tool continually collects metrics from the production server and compares them against pre-defined performance thresholds. When any threshold is exceeded, an automated alert notifies the support staff. If performed passively, we let our customers serve as an uncompensated monitoring crew, and they return the favor by leaving irate phone messages when they discover a performance problem.

2. **Identify Bottlenecks**
   Once a problem has surfaced, the first thing we do is use a trusted resource management tool to identify the physical resource (I/O, CPU, network or memory) that is experiencing the bottleneck. That part's easy enough. Then we put on our magician's hat and attempt to make the cause of the bottleneck become visible. This part may or may not be easy. The closer our physical resources are to full capacity, and the more processes are simultaneously contending for the same resources, the harder it is to isolate the one or few main offending processes.

3. **Optimize**
   Once the root cause of the bottleneck has been identified, one or more solutions are proposed to optimize its performance in order to ease the resource pressure and relieve the bottleneck. For the purposes of this paper, we will skip over the short term means of relief and focus on the lasting remedies. Often, indexes are adjusted or added to help queries

respond faster. Alternative query syntax may be tested in an effort to find a more efficient data access path. In dire circumstances, hardware is upgraded or portions of the application are redesigned.
Each proposed long-term solution to the identified bottleneck is tested in a non-production environment to determine which solution provides the most consistently dependable reduction in resource use, when compared to the "before fix" baseline performance measurement.

4. **Apply to Production**
After proper QA testing (or not), the fixes are applied into production. Performance monitoring resumes, in an attempt to confirm that the fix worked as expected, and in an attempt to catch the next performance headache that may be lurking around the bend.

The Pros and Cons of Query Tuning

The mechanics of Query Tuning are quite familiar. So much so that we may find ourselves talking about our last tuning adventure to our family members or neighbors for a few minutes until we notice the glazed looks and the facial expression that says "Get me out of here!"

The pros and cons of Query Tuning may, however, not be at the forefront of our minds. It's human nature not to evaluate a solution when historically it has been the only solution for such a long time. When we do take time to reflect, these considerations jump out:

Query Tuning is, by nature, reactive. Query Tuning is used only after a problem has become apparent. We just pray that our monitoring system notices the problem before our customers do.

Because Query Tuning is driven by DBAs and developers, manually focusing on only the top processes highlighted as the sources of our current bottleneck, it is by nature only able to optimize a few processes at a time. It is understood that quality solutions take time, and so we accept that it is just not feasible to be able to provide this kind of focused attention on more than a few problem areas. This practice is seen as normal and acceptable, but that is only because the costs of optimizing a large percentage of production processes have until now been prohibitive.

Behind most performance crises are usually only a small number of offending queries, and DBAs are very adept at troubleshooting and coming up with workable solutions for them. Query Tuning, whose sweet spot is to tune small numbers of queries, is thus perfectly suited for solving a crisis.

In a crisis situation, when the system is at or near complete halt, the topmost priority is to get the system back to operational status, **now**. This often invites shortcuts. We are praised for the speed, and not necessarily for the quality of our repair job. Sometimes our action to take the pressure off one offending process may unintentionally shift the pressure to some other portion of the database workload. The less time we have to perform realistic workload tests on the co-incident, resource-sharing processes, the less we know about the potential collateral damage. Usually, our joy at having made it home before midnight is enough to blot out such worries!

Query Tuning has the strong advantage of being familiar. Many of us DBAs and software engineers have a large amount of experience with Query Tuning, and so it's a comfortable and predictable method for getting us out of a hole. Furthermore, there are quite a few third party tools that are extremely good at helping us practice this methodology very efficiently.

On the flip side, even with the right performance monitoring tools, the diagnosis of the root cause

becomes much more challenging when several simultaneous processes are exhibiting multiple cross effects in real time. Heaven forbid that the problem should surface when utilization is near 100%, because then the problem is about as easy to spot as one particular snowflake in a blizzard.

Workload Tuning – the Promising Powerhouse

> ***"Being able to fix things does not make you a senior DBA. You become a senior DBA by making certain that things do not break in the first place."***
>
> *-- Thomas LaRock*

Workload Tuning introduces a completely new approach to Database Performance Management, complementing Query Tuning as a means of diagnosing and repairing production problems.

The newness is summarized in two adjectives: holistic and large scale. First, "holistic".

Workload Tuning is to Query Tuning what holistic medicine is to traditional medicine. If a man complains of a back ache, traditional medicine commonly addresses the precise location of the pain by prescribing muscle relaxants and physical therapy to stretch and strengthen the muscles. Relief can be obtained in this way, but the pain may return if the root cause of the back problem has not been addressed. Holistic medicine looks for the root cause. The holistic therapist may notice that the man's gait is uneven, and determine that there is a problem with the left foot that is transferring to the back. By additionally treating the foot problem, the back problem is cured for the long term.

The second new characteristic is "large scale".

Workload Tuning is to Query Tuning what production auto manufacturing is to custom car manufacturing. After years of doing all work piecemeal, and through diligent study of the process of manufacture, an automated approach was invented that can achieve a consistently high level of quality in each of thousands of units. A new expertise arose: that of designing and supporting the large-scale operation.

A similar advancement has arrived with regard to database performance tuning. For the first time, it is cost effective to tune every single SQL statement in a workload.

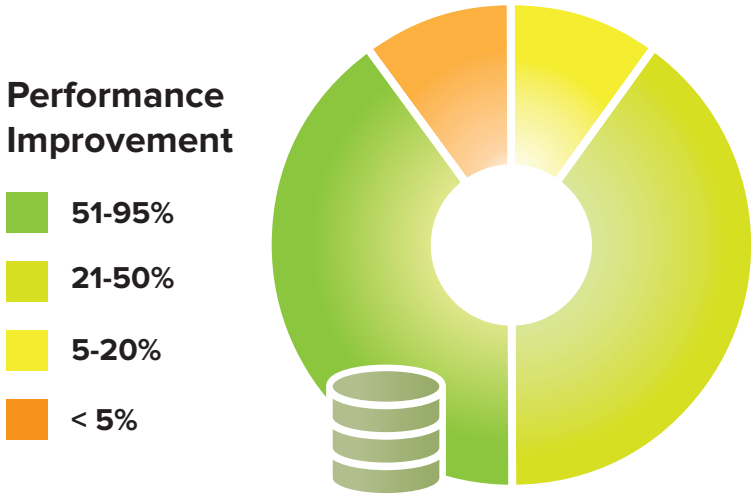
In over 200 production workloads and databases so far analyzed and optimized by Qure (see Figure 2), 80% have experienced a workload-wide average performance improvement of between 20% and 95%. To understand the 95% figure, it means that after applying Qure's recommended changes, the customers found that 95% fewer resources (CPU, query duration, reads, writes) were required to execute the same production workload. That is, the entire workload averaged a 20-fold improvement in query speed, and used 1/20th the CPU time, reads and writes.

That point deserves to be underscored. These performance improvements are not just for a handful of processes or on one of the measured resources. They reflect the average improvement across the thousands or millions of processes that make up your workload, and across all resources.

Specific examples of databases and workloads analyzed, and their results, can be found in the case studies page on the DBSophic website.

To shed some light on the nature of this new approach to tuning, we will first outline the mechanics of how it works. We'll then follow with its pros and cons.

Figure 2 - Workload Improvement Distribution



Average, workload-wide improvement in Duration, CPU and IO
Based on ~200 analyzed production workloads

How Workload Tuning Works

"Make everything as simple as possible, but not simpler."
-- Albert Einstein

Workload Tuning, as implemented by Qure for SQL Server, works as follows:

1. **Prepare a Dedicated Offline Analysis Environment**
   Qure was designed to work completely offline on a copy of the production database, set up on a dedicated analysis environment. Working offline is important for two reasons:

   - It enables a thorough analytical diagnosis which is practically impossible to perform in a production environment.
   - It allows Qure to automatically modify objects and code in the database copy in order to automatically empirically validate its recommended changes. That cannot be done in production.

   The copy of the production database includes everything (all tables, indexes, stored procedures, functions, views, triggers and data). This gives Qure the opportunity to fully understand all aspects of the database, and feed all this information into heuristic algorithms that can holistically optimize the entire workload.

2. **Create a Trace Log of the Production Workload to be Analyzed**
   The second input to Qure's analysis of your database is a comprehensive workload trace. The workload that is provided for the analysis should faithfully represent the full range of

activities that the application and databases perform. Within the trace is evidence of the largest bottlenecks, stress on the CPU, frequency of each type of SQL statement, variation in the parameters used for each type of SQL statement, and all the various other clues Qure will analyze in the process of solving the performance mystery represented by this evidence. The trace logs are the output of SQL Server's standard trace infrastructure, and are optionally guided by Qure-provided trace templates. The templates maximize the capture of useful information and minimize the capture of useless information so as to reduce the overhead on the production server during the trace collection.

3. Initiate a Workload Tuning Session

Qure can be installed on any workstation or server with a connection to the analysis environment. You launch an analysis by providing just a few simple parameters for the analysis: the server to be used for the analysis, the database(s) and trace log(s) to be analyzed, and a selection of preferred analysis options.

Automated Data Collection

Once the analysis process has been launched, Qure works completely unattended. The analysis duration depends on the size of the database and the workload being analyzed, the workload's characteristics, the analysis server's resources, and many other factors. The analysis may last from a few hours to a few days. To get an idea of the depth of analysis performed, look at Figure 3, the progress reporting screen.

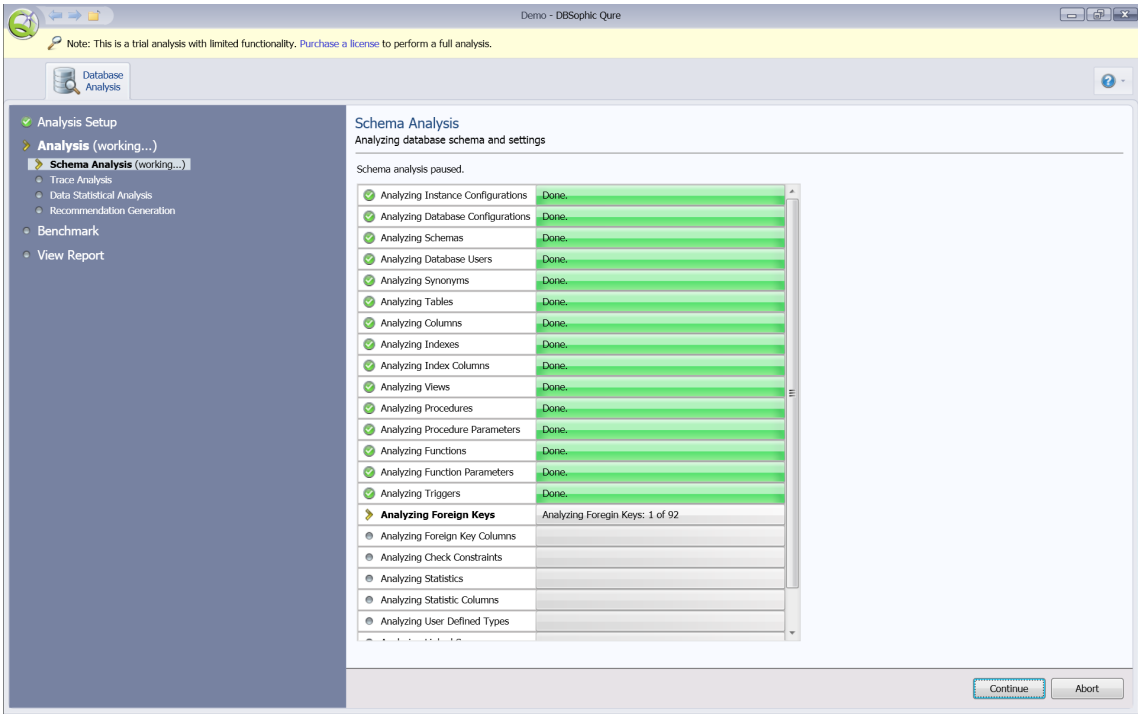


Figure 3 –Automated Analysis Process

During the data collection phase:

- Qure assesses all instances of each query (where an instance is a single execution of a query with a particular set of input parameters).
- Qure examines each query not as a single entity, but as part of a group of entities that, when stripped of its variable parameters, share the same function and syntax. Such a group is known as a “batch”. Of course, each particular instance of a batch may behave quite differently from the other instances, depending on the specific input parameters used.
- Qure’s algorithms rank every query according to its weighted consumption of resources, taking into account not just the average performance of each query, but also the number of times each query is executed within a normal workload

Automated Generation of Recommended Changes

Based on the collected data, Qure automatically generates recommendations to optimize each of the processes within the database that can be tuned. These recommendations can take several forms:

- Create, Modify & Drop Indexes
- Refine Object Code
- Modify Schema
- Adjust Configurations
- Refine Query Syntax
- Miscellaneous

The magic of these recommendations comes from a unique combination of factors:

1. A built-in expert knowledgebase, based on the experience of world renowned Microsoft SQL Server gurus, that can prescribe the one or more solutions for each query or process that best fits the specific workload demands.
2. The best index optimizer on the market. What makes it distinctive is that it does not optimize each index on a query-by-query basis (individually), but rather, holistically. It will recommend fewer indexes, each serving multiple processes using heuristics that determine how to achieve the best overall gain. It is also aware of the competing needs of the queries that access the same data, so it will balance the goal of increasing retrieval speed with the goal of minimizing index update penalties. The final outcome is a smaller number of newly introduced indexes, and a holistic, overall improvement in performance.
3. The ability to take into account information from multiple aspects of the database when coming up with recommendations for every process. For example:
    - Qure knows which data is being modified more frequently, so it can avoid over-indexing it so as to avoid the associated update penalty.
    - Qure uses the data’s statistical profile to tweak the actions of the indexing

optimizer. If, for example, a column has very low cardinality, that means it will not be very efficient for indexing anyway, and so it may be moved to be in the Included columns instead of in the Key columns.

- Object and query rewrites are informed by an awareness of the schema. Take for example the recommendation shown in Figure 6 (regarding a predicate data type mismatch). Such mismatches can only be found with an awareness of both the schema and the query structure.

4. The ability to automatically revise objects, batches and indexes to create new, executable versions.

More detail about the recommendations is provided in step 4, below.

Automated Baseline Creation

After the recommendation generation phase is complete, Qure automatically produces a performance baseline for the given workload by replaying that workload against the copy of the production database.

Measurements taken during each benchmark run include physical reads, logical reads, writes, CPU and duration of execution.

Automated Optimization of the Analysis Environment

Qure automatically applies its recommended changes (that is, its revisions to query syntax, objects, indexes and configurations) to the copy-of-production database. Schema changes may also, at your option, be automatically applied and benchmarked.

With potentially hundreds or even thousands of recommendations, it's quite handy that these revisions are tested and validated to be functionally identical to the originals, syntactically correct, error-free, and ready to execute. This is obviously an enormous time saver for the DBA who would otherwise have to write or apply or adjust the recommendations manually.

Automated Benchmarking of the Performance Impact

With the recommended modifications to the database and its objects in place, Qure replays the entire workload once again to accurately measure the performance gains achieved. It uses these results to fine-tune the recommendations of specific batches that need further work. Qure may revise and re-benchmark such batches multiple times until an optimal improvement is achieved.

4. Review Qure Recommendations, Rationales and Results

The empirical "before" and "after" measurements are presented to you in an easy-to-understand set of interactive reports detailing the magnitude and nature of the performance improvements (see Figure 4).

A common curiosity about Workload Tuning is how the recommendations are generated and what they look like.

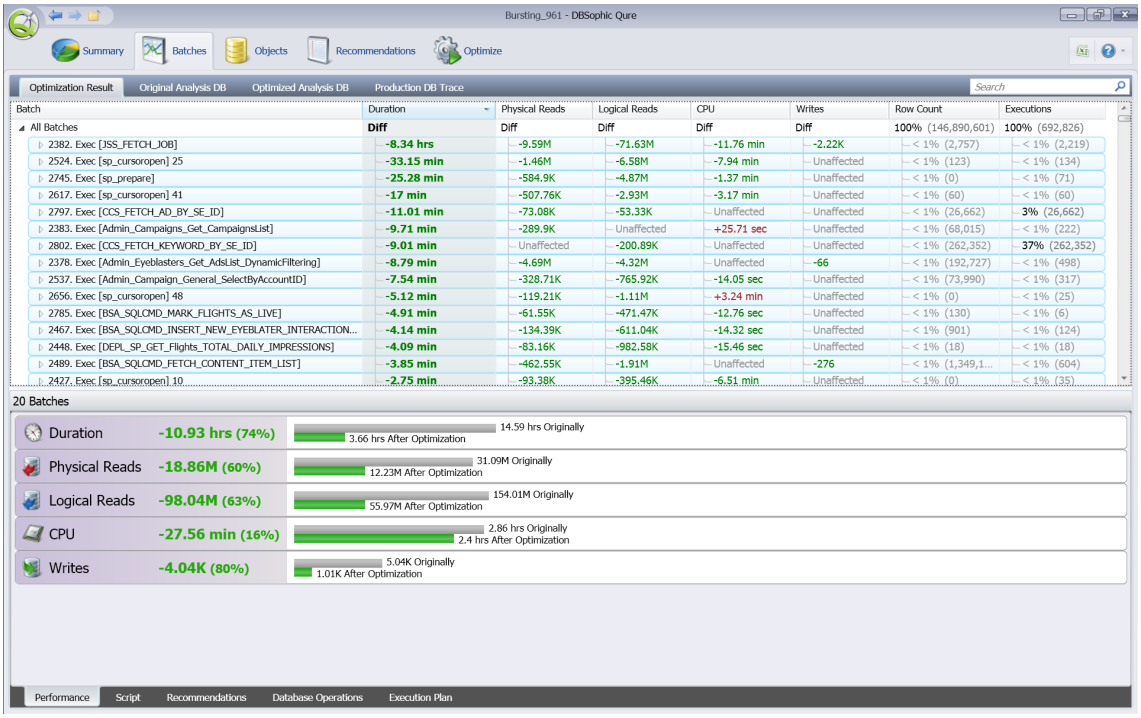


Figure 4 - Workload Performance Improvements

The Recommendations

Recommendations are based on information about many aspects of the database and the workload, and the cross-references between them: query structures, table structures, index structures, access patterns, statistics and metadata.

Recommendations generated by Qure contain three parts:

1. A rationale, in plain English, for the suggested change.
2. A generated correctional script.
3. A list of objects and batches affected by the recommendation.

A Plain English Rationale for the Suggested Change

Qure provides a clear and comprehensive explanation for the rationale underlying each recommended change. See Figure 6 for a sample. These explanations are intended to improve DBA team competence, and with time, the competence of the Development team.

A Generated Correctional Script

For (nearly) all recommendations, Qure will generate a fully executable correctional T-SQL script which implements the recommended change. Such corrections can be generated to revise a stored procedure, to ALTER TABLE, to CREATE INDEX, etc. See Figure 5 for one example. For more examples, watch the free webinar “Working with Qure’s Recommendations” at www.dbsophic.com/resources. Such scripts, of course, are mandatory for Qure to be able to

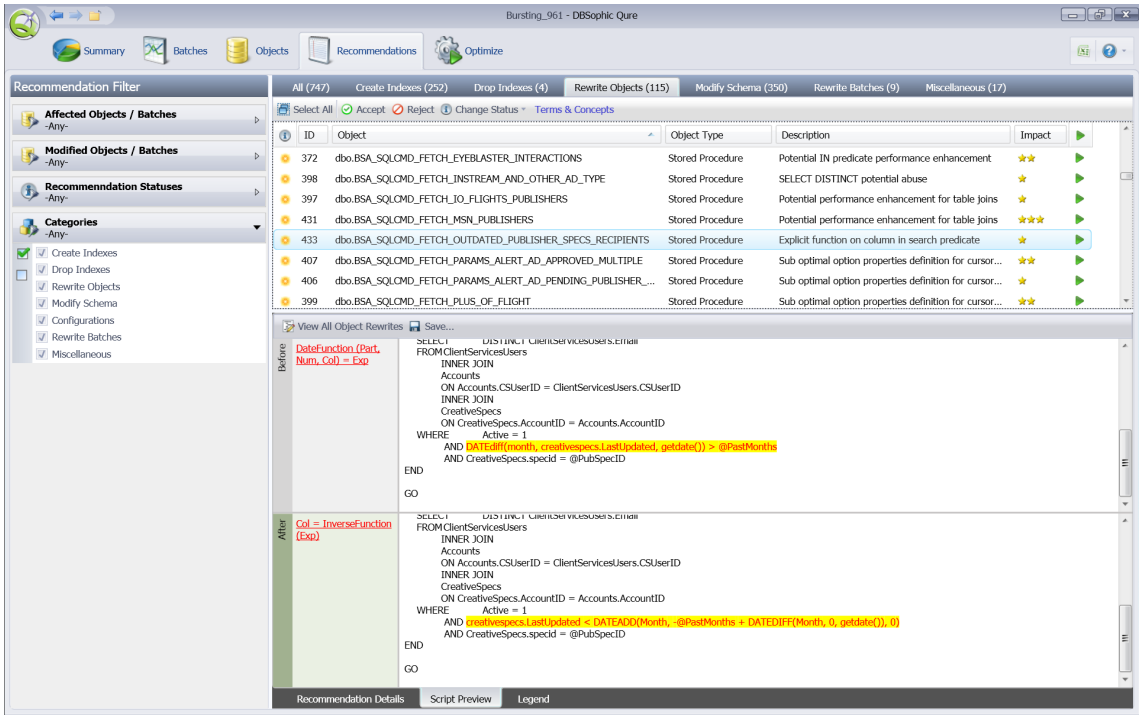


Figure 5 - Executable Scripts for Deploying Recommendations

automatically apply the recommendations during the benchmark phase. This also allows you to deploy the recommendations very quickly onto the QA or production environments.

A List of Objects and Batches Affected by the Suggested Recommendation

Because there may be multiple dependencies between objects in the database, and because DBAs are understandably careful about the downstream impacts of changes made to any one object, Qure reports all the objects and batches impacted by every recommended change. See Figure 6. This is an interactive report, so you can easily move between the co-affected objects to examine them.

Accept or Reject Specific Recommendations

You are given an interface that allows you to sort and filter the recommendations by object type, batch name, recommendation type, level of improvement, and other criteria. After reviewing one or two recommendations of a certain type, you can accept all like recommendations if you wish. This will help your DBA team to work quickly through the list, which commonly includes hundreds of individual recommendations.

5. Perform QA and Release Changes into Production

The saved collection of recommendations, which as you recall contains correctional scripts for objects and batches, can then be copied and applied onto the QA environment and tested via your organization's QA procedures. After deploying recommendations to the production environment, you will immediately experience the benefits of much improved performance.

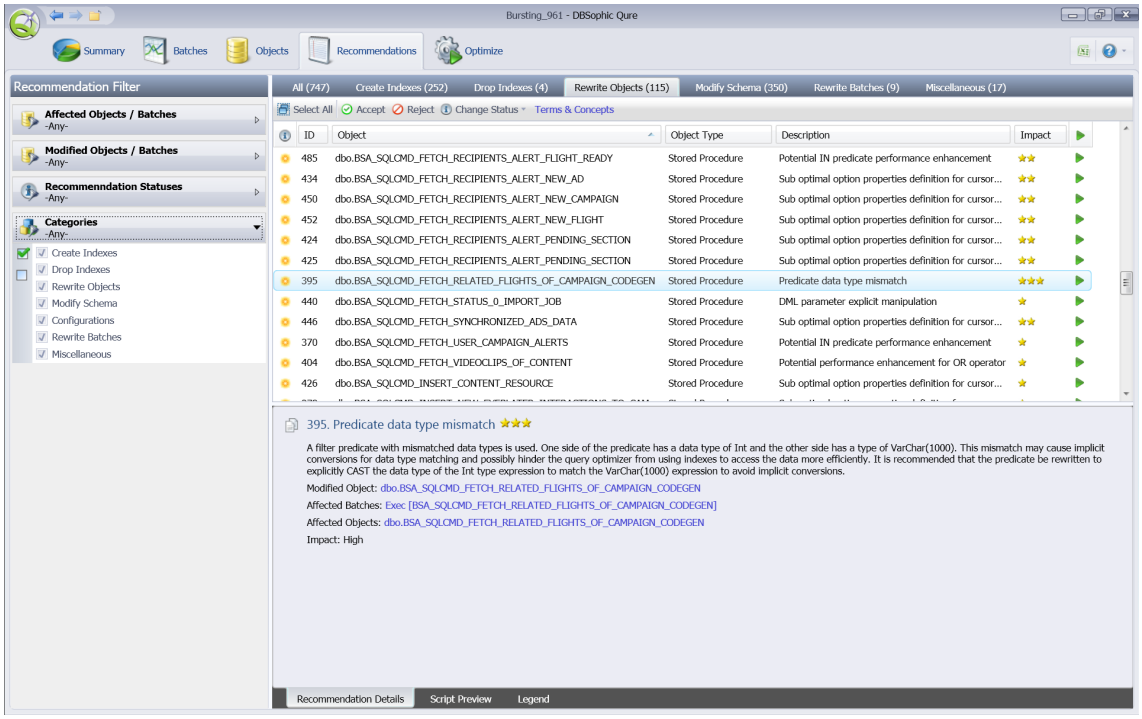


Figure 6 - Recommendation Detais

The Pros and Cons of Workload Tuning

After understanding the mechanics of Workload Tuning, and after reviewing the pros and cons of Query Tuning, there are a few pros and cons of Workload Tuning that jump out immediately:

Workload Tuning is proactive. It finds 'problems' before they become problems.

The number of problems it can find is virtually unlimited. Since Qure is an automated mechanism that analyzes your entire database including all queries and processes included in your actual production workload, Qure can optimize queries that humans would never have the time to work on. Stored Procedures that occasionally require several seconds to respond may not be considered dire enough to address using the Query-by-Query Tuning method. But Qure will always notice these, and if there is a recommendation for improving data access paths, you will hear about it.

Qure analyzes holistically, not individually. Qure uses heuristic algorithms to tune the database workload as a whole, balancing each potential adjustment against each other in a way that only a software utility can do, because there are far too many comparisons for a human.

For example:

- When Qure finds two (or more) efficiently designed structures which are sometimes underperforming, Qure assesses the available trade-offs by giving the highest performance priority to the top resource consumers and the lowest priority to the lowest resource consumers.
- Modification recommendations are not determined in isolation. If an index or stored procedure is being optimized, Workload Tuning takes into account the projected impacts

on all real-world queries that rely on that object before settling on its recommendation.

- After each recommendation is applied in the form of a modified object, batch or configuration, that recommendation is empirically benchmarked against the workload that relies on the object that was changed to test whether the recommendation was indeed optimal.
- Qure's benchmark tests many different parameter sets found in the production workload for each batch, not just a handful. This helps test those hard-to-find parameter combinations that may elude us in QA but cause us problems in production.

Qure performs intensive analyses. To support such intensive processing, a dedicated analysis server is required, capable of hosting the entire production database, and offering still more storage for the purposes of applying recommendations. A set of trace logs representing a full production workload is also needed. As a result, Workload Tuning may not be the approach that provides the quickest turnaround in a crisis.

Because Workload Tuning is holistic, and uses heuristics that seek the low-water-mark of resource usage, an intentional re-balancing takes place. The re-balancing purposefully does the most favors for the highest resource consumers. What this means is that there is no guarantee that any one process will be improved. Rather, you'll normally find that the overall workload has been significantly improved.

Workload Tuning's impact is almost always positive, and more often than not, downright impressive. Because of these types of improvements, the useful life of an organization's existing hardware and storage can be extended significantly. These savings translate to organizations experiencing a solid ROI when using Qure. Smiles are allowed.

Workload Tuning is a natural evolutionary step in DBA tools – allowing the grunt work to be accomplished automatically, and accomplishing a portion of the expert work. If Qure's work can be considered the rough carpentry, then the DBA's remaining work can be considered the finished carpentry; the work requiring the higher level of expertise and polish. Thus Qure provides DBAs the freedom to spend more of their time at the higher end of the skill curve.

Yes, Qure produces new correctional scripts that measurably improve the performance of the database's objects and batches, and can be used as is (after being tested by your organization's QA process).

No, Qure, is not able to take into account all inputs known to expert DBAs:

- Qure uses a fixed number of optimization methods. DBAs are likely to know additional ones, and may achieve further improvements if they have the time. (Want results? Dare a DBA.)
- Qure is not aware of the relative importance each process has to the business. A DBA is. The DBA may want to rebalance the recommendations based on this additional information.
- Qure cannot determine the maximum number of recommendations (changes) that can be contained within an organization's change management process and project plan for any particular software release. Organizations may wish to put a constraint on the volume of changes considered acceptable. A DBA is needed to make this determination, and to select the desired recommendations. Qure provides a robust set of tools to help the DBA quickly sift through recommendations based on the objects being affected, the types of recommendations, and so on.

- Qure's only knowledge of your database is drawn from a copy of the production database and from the trace logs of the production workload. A DBA knows well that there are other variables besides these which can impact recommendations for change.

Nonetheless, because of the advent of Workload Tuning, DBAs are less weighed down with emergency incidents, and can place their focus on the types of complex tasks only humans can do, such as considering the impact of the business process on the database workload's performance, or making architectural improvements, or having a pleasant conversation over tea.

Conclusion

As we have seen, Query Tuning and Workload Tuning complement each other in the following ways:

	Traditional Query Tuning	Workload Tuning
Timing of Tuning Action	Reactive	Proactive
Familiar, Intuitive	Yes	Will soon be
Scope of optimization	Narrow	Broad
Test environment required	Small	Large
Awareness of impact to other processes	No	Yes
Improvements across multiple processes	No	Yes
Elevates professional skill level	Somewhat	Significantly
Useful for performance crises	Yes	Somewhat
Helps prevent emergency capital investments	No	Yes

Practitioners of Database Performance Management now have two means of tuning a database. With the introduction of Workload Tuning, Query Tuning has a capable partner. DBAs now have a broader range of tools at their disposal to get ahead of the ever-more-challenging requirements of increasing volumes at higher speeds for lower costs. Powered by a knowledgebase of SQL Server optimization techniques, and informed by a holistic analysis of all aspects of production workload, Workload Tuning makes it possible for you to automatically balance the conflicting demands found within extremely complex workloads. You exercise control by determining what constitutes your workload, reviewing the recommendation rationales and benchmark results, examining the ready-to-implement correctional scripts, and accepting or rejecting said scripts based on your business priorities and your expert knowledge of database-specific characteristics unknown to Qure. Because Qure is so fully automated, it can optimize millions of queries and objects at a crack, making it the only cost-feasible method available to optimize your entire production workload.

More information

About Ami Levin

Ami Levin, CTO & co-founder of DBSophic, is a Microsoft SQL Server MVP, with over 20 years of experience in the IT industry. For the past 12 years he has been consulting, teaching and speaking on SQL Server worldwide. He manages the Israeli SQL Server user group, leads the local SQL Server support forum, and is a regular speaker at SQL Server conferences worldwide. Ami's latest technical articles can be found on SQL-Server-Performance.com.

About DBSophic

DBSophic is an innovative provider of performance management products for database-centric applications. While traditional tools focus on optimizing individual queries, DBSophic delivers automatic workload tuning solutions. By optimizing the entire application-to-database workload, DBSophic's solutions help enterprises achieve unparalleled gains in application performance. The company is headquartered in Jerusalem.

› For more information, please visit http://www.dbsophic.com

"If I had thought about it, I wouldn't have done the experiment. The literature was full of examples that said you can't do this."

-- Spencer Silver, on the work that led to the unique adhesives on 3M Post–It notepads

DBSophic
Unleashing Database Performance

› www.dbsophic.com