# Work Instruction

Azure SQL Database Health Check

**Version 1.0**

Contents

[Work Instruction 1](#_Toc184138637)

[Document Control 3](#_Toc184138638)

[Distribution List 3](#_Toc184138639)

[Amendment Record 3](#_Toc184138640)

[Terminology 3](#_Toc184138641)

[Executive Summary 4](#_Toc184138642)

[Pre-requisites 4](#_Toc184138643)

[General Guidelines 5](#_Toc184138644)

[Health Check 7](#_Toc184138645)

[Database Information and Resource Limits 7](#_Toc184138646)

[Memory Utilisation 8](#_Toc184138647)

[Memory Distribution 8](#_Toc184138648)

[Buffer Pool 8](#_Toc184138649)

[Plan Cache 8](#_Toc184138650)

[Stolen Memory 9](#_Toc184138651)

[I/O 10](#_Toc184138652)

[Percentage of Service Tier Limit 10](#_Toc184138653)

[Waits 10](#_Toc184138654)

[Latency 10](#_Toc184138655)

[CPU 12](#_Toc184138656)

[Percentage of Service Tier Limit 12](#_Toc184138657)

[Waits 12](#_Toc184138658)

[Indexes 13](#_Toc184138659)

[Index Fragmentation 13](#_Toc184138660)

[Missing Indexes 13](#_Toc184138661)

## Document Control

### Distribution List

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Representing** |
| Marcelo Farinelli | Senior Technical Consultant | Blue Crystal Solutions |
| Thuc Do | Client Delivery Manager | Blue Crystal Solutions |
| Steven Macris | Delivery Manager | Blue Crystal Solutions |

### Amendment Record

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Status** | **Comment** | **Date** |
| 1.0 | Open | Initial version | 03/12/2024 |

### Terminology

|  |  |
| --- | --- |
| Acronym | Definition |
| BCS | Blue Crystal Solutions |
| DML | Data Manipulation Language |
| SSD | Solid State Drive |

## Executive Summary

Customers often engage BCS to perform a health check on nominated Azure SQL Databases.

The DBA can use the procedure described in this document to complete a health check with minimal setup or dependencies by connecting directly to the Azure SQL Database from a laptop, using exclusively T-SQL scripts, and without requiring access to the customer Azure Portal.

The Azure SQL Database health check described in this document focuses primarily on resource utilisation by the database rather than instance configuration best practices. Only database scoped configuration parameters can be changed on Azure SQL Databases.

A customer might request a more detailed analysis of the workload such as setting up Extended Events for data collection and query execution plan optimisation. This is out of the scope of this document and should follow standard performance analysis practices.

## Pre-requisites

The customer must provide:

1. Access to the Azure SQL Database – the customer must create the necessary firewall rule to enable access from the DBA endpoint. This could be:
   1. the BCS internet IP address (if working from the office), or
   2. the DBA’s individual internet IP address (if working from home).
2. Credential for database connectivity and minimum required privileges:
   1. a SQL authenticated credential or a domain account (if Azure AD integration is available), and
   2. membership in the ##MS\_ServerStateManager## database role.

The DBA will need:

1. This document.
2. SQL Server Management Studio.
3. T-SQL scripts that accompany this document. The location of these scripts is specified in each section of this document.

## General Guidelines

It is important to understand the fundamental differences between SQL Server instances and Azure SQL Databases:

* An Azure SQL Database differs from SQL Server instances in that there is no actual SQL Server instance. Instead, there is a [logical server](https://learn.microsoft.com/en-us/azure/azure-sql/database/logical-servers?view=azuresql&tabs=portal) that hosts the master database and all databases created on that server.
* SQL logins are managed at the logical server level to manage all databases, or at the database level to manage individual databases.
* It is not possible to modify configuration parameters in the master database. Only database scoped configuration can be modified.
* Compute resources are not assigned exclusively to the logical server. Instead, each database can have its own [purchasing model](https://learn.microsoft.com/en-us/azure/azure-sql/database/purchasing-models?view=azuresql): [vCore](https://learn.microsoft.com/en-us/azure/azure-sql/database/purchasing-models?view=azuresql#vcore-purchasing-model) or [DTU](https://learn.microsoft.com/en-us/azure/azure-sql/database/purchasing-models?view=azuresql#dtu-purchasing-model). Purchasing models determine the hardware configuration (CPU type and memory limits). Under each purchasing model there are service tiers to choose from. The service tier limits the storage type, size and IOPS, as well as the number of cores. It also offers high availability options. The customer chooses a suitable service tier for each database based on business continuity, storage and performance requirements.

A basic health check for an Azure SQL Database consists of:

1. Collecting configuration information about the database and service tier limits.
2. Collecting and analysing recent history of resource utilisation (past 14 days are available via DMVs) as percentages of the service tier limit.
3. Examining Query Store (enabled by default) for expensive queries.
4. Collecting and examining buffer pool and plan cache memory distribution and effective utilisation.
5. Investigating index fragmentation and missing indexes.
6. Investigating excessive waits.

The health check should answer the following questions:

1. Based on recent history, are compute resources under or over utilised?
2. Are there optimisation opportunities that can improve resource utilisation? More specifically:
   1. Is buffer pool memory being wasted with half-empty database pages? This is caused by index fragmentation (page splits) and/or index fillfactor settings.
   2. Is buffer pool memory being stolen to perform sort and hash operations? This memory is never released back to the buffer pool.
   3. Are there CPU intensive queries with excessive logical reads (index scans)? Missing indexes could be the cause. Can query design be improved?
   4. Is the plan cache bloated with single-use execution plans? How much memory is consumed by these single-use plans versus multiple-use plans? Enabling “Optimise for Ad-hoc Workloads” for the database might help, but this is not an universal solution.
3. Is there disk latency? If so, is it because the workload is hitting the max IOPS limit determined by the service tier?
4. If resources are underutilised, is there an opportunity to reduce cost by moving to a lower service tier without negatively impacting the workload?

## Health Check

There is no template for this health check. However, the following documents are real examples of final deliverables that can be used as references:

[SR0062007 AzureSQL Database Health Check](https://bluecrystalsolutions.sharepoint.com/:f:/s/BCSDelivery/EuIDcLYwtZ9Prs4-n4heB0MBW1mUcDJ56916t6Cxx5_5Gg?e=tkZP8O)

[SR0060948 Azure SQL Database Health Check](https://bluecrystalsolutions.sharepoint.com/:f:/s/BCSDelivery/EniEP1-LfNVJviSH31aN49wBH-44XkTR5cnsDHXaUsEDdQ?e=xxzcHE)

You should use Excel to assist your analysis and produce the charts used in the deliverables. Example:



Scripts and instructions to collect the data used in these documents are supplied in the subsequent sections. Each script will indicate whether it must be run on the database target of your investigation or the master database. This is important because each database has information about its own resource utilisation and service tier limits. Note that if you run a script on the master database, you will get information about the logical server instead.

### Database Information and Resource Limits

Use the following script to populate a table in the deliverable with information about the database:

[Settings and Events.sql](https://bluecrystalsolutions.sharepoint.com/:u:/s/BCSDelivery/EXrKAVwlvg5Jqp4AUWzNrVsB7gHVczCmc_YMrPM3ruOlog?e=pVGSs1)

Example:

|  |  |
| --- | --- |
| Product | Azure SQL Database |
| Logical Server Name | mpmsql01 |
| Database Name | rapidMATION |
| Database Size | 1944GB Allocated. 1518GB Used. |
| Service Level Objective | GP\_S\_Gen5\_12 |
| Max vCores | 12 |
| Max Memory | 36GB |
| Max Data IOPS | 3840 |
| Max Data Size | 3TB |

### Memory Utilisation

Memory Distribution

Use the following script to collect memory distribution across buffer pool, plan cache and stolen memory (sorting, hashing):

[Memory Distribution.sql](https://bluecrystalsolutions.sharepoint.com/:u:/s/BCSDelivery/EVdfcMLX6XJHs45XA_PFeFoBU3zsxxq4oPvX9W7sM7HqTg?e=PLNQmv)

Use the results to produce a chart for the deliverable. A large plan cache can be a sign of many redundant (single use) plans. Stolen memory is primarily for sorting and hashing due to expensive queries requesting large memory grants and this memory is never released back to the buffer pool. Plan cache and stolen memory will be explored later in this document.

Buffer Pool

Use the following script to determine how much memory is allocated to actual data and empty space in the buffer pool:

[Buffer Pool Memory Used vs Wasted.sql](https://bluecrystalsolutions.sharepoint.com/:u:/s/BCSDelivery/Eba6lyDNH7lDsZyyew_h-GoBsDIA6N8JyYbW33kE_GFy0A?e=db52ku)

Use the results to produce a chart for the deliverable. With this script you can also discover which indexes are the worst offenders and their respective fillfactor. While fragmentation is usually the main cause of empty space left within pages after page splits, a wrongly set fillfactor will reserve empty space when the index is rebuilt or reorganised, and that empty space will never be filled if the row is never updated, resulting in wasted memory.

Plan Cache

Use the following script to determine how much memory is allocated to single use versus multiple use plans in the Plan Cache:

[Plan Cache Memory for Single vs Multiple use plans.sql](https://bluecrystalsolutions.sharepoint.com/:u:/s/BCSDelivery/EWiN44siPiVEuVCer1Rpn94BHQA7ele-xpvcytQYn4O1CA?e=F6czDZ)

Use the results to produce a chart for the deliverable. You can also investigate which queries have the most redundant plans using the following script:

[Redundant Single-Use Plans from Plan Cache.sql](https://bluecrystalsolutions.sharepoint.com/:u:/s/BCSDelivery/EZhydZ4fDSNNnngd4c8NQLMBK2TNKtlz3SGHEr07Ex6Zfw?e=1qua0V)

Use the results to supply examples of nearly identical queries that have many single-use plans in an Excel spreadsheet.

If the application executes queries that produce many single use plans bloating the Plan Cache, enabling OPTIMIZE\_FOR\_AD\_HOC\_WORKLOADS for the database is not an universal solution to minimise memory consumed by single use plans, but it is generally recommended and can help in these cases.

Stolen Memory

Memory can be stolen from the Buffer Pool mainly to perform sort or hash operations. The amount of memory stolen is highly dependent on memory grants requested by the workload.

Query Store is enabled by default in Azure SQL Database, and you can find worst offenders by going to “Top Resource Consuming Queries” and looking for queries with high memory consumption:

A screenshot of a computer

Description automatically generated

Query Store in Azure SQL Database is configured by default to retain query information for only 30 days. Identifying which queries are requesting large memory grants requires data collection for an extensive period; ideally long enough for all possible workloads to run by the application against the database. This is out of the scope of this document. The work can be undertaken as part of an in-depth performance analysis and will require the following to be set up:

* Azure storage account and container to store data collection.
* Database master key to encrypt the database scoped credential.
* Database scoped credential to access the storage account.
* Extended Event Session to capture events for the analysis.

### I/O

I/O bottleneck can be identified by examining I/O waits, disk latency, and utilisation as a percentage of service tier limit.

Percentage of Service Tier Limit

Use the following script to capture I/O utilisation as a percentage of service tier limit over the past 14 days:

[Resource Utilisation.sql](https://bluecrystalsolutions.sharepoint.com/:u:/s/BCSDelivery/Eel85DZGkAVKpEpt7cTE9mcBaoUiNGdrtvRHAv5BBGwvAA?e=CWb69J)

Use columns UTC\_time, avg\_data\_io\_percent and avg\_log\_write\_percent to produce a chart for the deliverable. The date column is in UTC. Analyse the chart for peaks and sustained high utilisation. Low utilisation might generate a cost saving opportunity if other compute resources are also under-utilised.

The DMV holds only 14 days of data, which may not be sufficient to get a picture of utilisation for all types of workloads (transactional, data warehouse etc). Historical information would be available only through the Azure Portal.

You can also look for queries with excessive physical reads and large row counts in Query Store under “Top Resource Consuming Queries”.

Waits

Use the following script to identify I/O related waits:

[Waits.sql](https://bluecrystalsolutions.sharepoint.com/:u:/s/BCSDelivery/EZ50zMvXLWFJgKdq9X1aYi0BCR8rURJEoO0BS1QKroEqeg?e=heJbJN)

Consult the documentation for details on each wait to determine if they are problematic and why. Write a recommendation or observations based on your analysis.

You can also look for the top wait types and their respective queries in Query Store under “Query Wait Statistics”.

Latency

Use the following script to identify I/O latency:

[Disk Latency.sql](https://bluecrystalsolutions.sharepoint.com/:u:/s/BCSDelivery/EdtuhqQ7tPdPmu59j0sFqm0Bho7ttjzYeNpkUk9CSodWBQ?e=2if6ub)

The script returns latency information since the logical server last started. The important columns are:

* ReadLatencyByGovernance and WriteLatencyByGovernance: latency introduced by resource governance. It means resource governance capped IOPS at the service tier limit and latency was introduced as a result.
* ReadLatency and WriteLatency: latency observed by the workload. These values are compounded by latency introduced by resource governance.

Based on your analysis, provide a recommendation or observations. I/O latency can vary depending on the workload type. For instance, in data warehouse (DW) environments, queries that return millions of rows of historical data often require data to be fetched from disk, as it is infrequently found in memory. Additionally, resource governance may limit IOPS based on the service tier, contributing to latency. The key consideration is whether the current I/O performance meets the business's needs, rather than automatically requiring a higher service tier.

In transactional environments, high I/O latency can be exacerbated by inefficient use of the Buffer Pool, where memory is wasted on half-empty pages due to fragmentation or incorrect fillfactor settings. It may also stem from poorly designed queries that lack appropriate JOINs, WHERE clauses, or necessary indexes.

### CPU

CPU bottleneck can be identified by examining CPU waits and utilisation as a percentage of service tier limit.

Percentage of Service Tier Limit

Use the following script to capture CPU utilisation as a percentage of service tier limit over the past 14 days:

[Resource Utilisation.sql](https://bluecrystalsolutions.sharepoint.com/:u:/s/BCSDelivery/Eel85DZGkAVKpEpt7cTE9mcBaoUiNGdrtvRHAv5BBGwvAA?e=CWb69J)

Use columns UTC\_time and avg\_cpu\_percent to produce a chart for the deliverable. The date column is in UTC. Analyse the chart for peaks and sustained high utilisation. Low utilisation might generate a cost saving opportunity if other compute resources are also under-utilised.

The DMV holds only 14 days of data, which may not be sufficient to get a picture of utilisation for all types of workloads (transactional, data warehouse etc). Historical information would be available only through the Azure Portal.

You can also look for queries with excessive CPU time in Query Store under “Top Resource Consuming Queries”.

Waits

Use the following script to identify CPU related waits:

[Waits.sql](https://bluecrystalsolutions.sharepoint.com/:u:/s/BCSDelivery/EZ50zMvXLWFJgKdq9X1aYi0BCR8rURJEoO0BS1QKroEqeg?e=heJbJN)

Consult the documentation for details on each wait to determine if they are problematic and why. Write a recommendation or observations based on your analysis.

You can also look for the top wait types and their respective queries in Query Store under “Query Wait Statistics”.

### Indexes

Index Fragmentation

Use the following script to identify index fragmentation:

[Index Fragmentation.sql](https://bluecrystalsolutions.sharepoint.com/:u:/s/BCSDelivery/EabOqtvW3eRKhHe_cUTftYwBqI5UEpq9zc3np8dh8diP4w?e=KITjxI)

Identify indexes that are heavily fragmented or have low page density. Eliminating fragmentation to increase page density will improve Buffer Pool effectiveness by filling it with less empty space and more actual data.

Incorrect fillfactor will contribute to low page density and create empty space within pages after a rebuilt or reorganise. If indexes identified have fillfactor set to anything other than 0 or 100, the customer should be notified to determine the reason and possible remediation.

Write your recommendations and observations regarding fragmented indexes and the fillfactor setting if applicable. Your recommendations should include:

* Index maintenance as part of an ongoing strategy rather than a one-time event.
* Reorganising indexes rather than rebuilding to minimise the impact to the workload and prevent long-term locks.

Supply a list of the identified indexes in an Excel spreadsheet.

Missing Indexes

Use the following script to identify missing indexes from the Plan Cache:

[Missing Indexes from Plan Cache.sql](https://bluecrystalsolutions.sharepoint.com/:u:/s/BCSDelivery/EeD4QS5rXFpIlSsp9XtElcoB2bBUqfuPPsrFQSvRQGEP_g?e=mGu4aT)

The script may take a couple of minutes to run as it fetches information from the Plan Cache.

Identify execution plans lacking proper indexing. Recommending targeted indexing for expensive queries will improve query response times and eliminate unnecessary logical and physical I/O.

Use the following main indicators in your analysis:

* **Impact:** The **impact %** column indicates the impact of the missing index on the execution plan.
* **Execution Count:** The **execution\_count** column shows the number of times the plan has been executed.
* **Logical Reads:** The **total\_logical\_reads** column shows the number of database page requests from the Buffer Pool.
* **CPU Time:** The **total\_worker\_time (ms)** column shows the total amount of CPU time that was consumed by executions of the plan.

Execution plans with high-impact missing indexes, high execution count, high number of logical reads and high CPU time are good candidates for indexing.

However, index recommendations should be kept to a minimum and should be evaluated by the application vendor for approval. New indexes consume additional disk space and there is additional overhead to maintain these indexes on every DML operation.

Supply a list of the identified queries in an Excel spreadsheet for customer reference and list the index creation commands in the deliverable. This information can be found in the script output, columns **sql\_text** and **missing\_index\_creation**.