

Query timeouts

Last updated by | Vitor Tomaz | Jun 8, 2022 at 5:37 AM PDT

Contents


- See [Query timeouts](#) under SQL DB as well for more TSGs.
- Self-help content presented in Azure Portal
 - [Command Timeout](#)
 - [Connection Timeout](#)
 - [Performance degradation - monitoring and performance t...](#)
 - [Performance recommendations when migrating to Manag...](#)
 - [Compare instance settings on SQL Server and Managed In...](#)
 - [Resources](#)


See [Query timeouts](#) under SQL DB as well for more TSGs.

Self-help content presented in Azure Portal

(This content was shown to the customer during case submission. It's also visible on 'Diagnose and solve problems' blade.)

Command Timeout


In general, command timeouts can be investigated using [query data store](#) , where `exec_type = 3` captures queries with a timeout.

That said, by investigating the wait stats on a specific query, we can further identify the bottlenecks with the problematic/slow performing query. Follow the document [Investigate Waitstats](#)  to further analyze waits.

If the query bottlenecks on IO:

- Increase the query timeout
- Further optimize query if possible
- If query must perform lots of IOs and is latency sensitive than latencies per IO are better in BC offers

If the query bottleneck is on CPU:

- Check other CPU intensive activities
- Identify an other user load
- Optimize or [batch](#)  the query or scale the database

Connection Timeout

By default, there is a 30 second timeout in .net connection. Increasing this will probably not help, as the issue could be that you are running same query many times and they starts blocking each other.

Go to your database then Query Store, and see your top queries run time (Duration). Start optimisation from there. Potential places could be EntityFramework (if you are using it); this could generate queries returning large amounts of data, which slows down the query and uses a lot of IO.

If you still want to increase timeout, do so in the [.config](#) file for your connection string by adding `;Connection Timeout=<value>`.

Performance degradation - monitoring and performance tuning

SQL Server has its own monitoring and diagnostic capabilities that SQL Database and SQL Managed Instance leverage, such as [query store](#) and [dynamic management views \(DMVs\)](#). See [Monitoring using DMVs](#) for scripts to monitor for a variety of performance issues.

Query Performance Insight doesn't support SQL Database Managed Instance, and we don't have an ETA for when it will. Instead, use [Azure SQL Analytics](#) for monitoring Managed Instance performance, with automated troubleshooting based on AI.

Here is a [short demo video](#) that walks through Performance Monitoring for Azure SQL Managed Instance with Azure SQL Analytics.

In addition, [this library](#) can be used to track performance of workloads in Managed Instance using TSQL.

Performance recommendations when migrating to Managed Instance

When you migrate your databases from SQL Server (on-premises or Azure VM) to Azure SQL Managed Instance, the first thing to do is to compare the performance of your database in the new environment with the performance of the original database on the source SQL Server. In some cases you might see that performance of the databases on Managed Instance are worse than performance of source SQL Server. This is sometimes expected because Managed Instance has some management overhead (automatic backups, resource limits required to guarantee 99.99% availability), while in other cases these are some settings that can be configured to improve performance of Managed Instance.



If you are experiencing performance differences between Managed Instance and SQL server, find hints and diagnostic checks that you can perform to identify the root causes of the performance issues [in this Medium.com article](#).

Compare instance settings on SQL Server and Managed Instance for optimal performance

One reason why there is performance difference between SQL Server and Managed Instance is that their databases are not configured same way (such as different recovery model, compatibility level, legacy cardinality estimator, trace flags, encryption, tempdb settings). There are a lot of settings that might impact performance and it is hard to identify them.

When you migrate your databases from one SQL Server instance to another or from SQL Server to cloud (for example Azure SQL Managed Instance), first compare the workload performance between source and target environment. Sometimes, you will get different performance, although you believe that the source and target environment are the same. There are several factors that can cause different performance on source and target instances, such as:

- Different server/database properties on source and target instance (compatibility levels, cardinality estimator, encryption, etc.)
- Different trace flag settings
- Different tempdb settings (number of files, encryption)

If you experience performance difference, compare the settings between the SQL Server and Managed Instance where the performance tests are being conducted. SQL scripts are located [in GitHub](#) , and the instructions to use the scripts can be found [in this Medium.com article](#) .

Resources

- [Monitoring and performance tuning](#) .

How good have you found this content?

