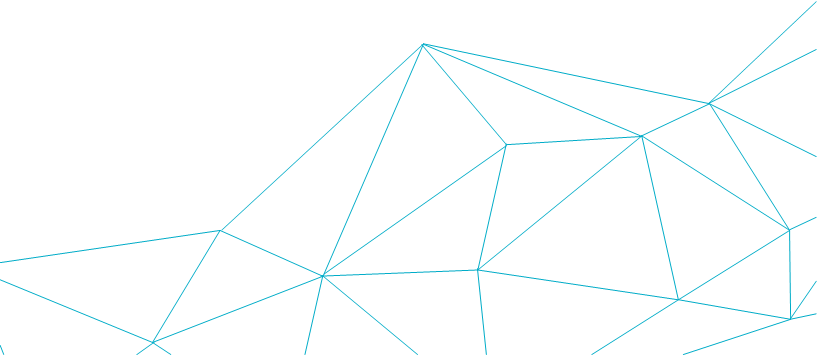


SQL Server Maintenance

March 28, 2024

This document contains unpublished confidential and proprietary information of Odessa, Inc. No disclosure or use of any of these materials may be made without the express written consent of Odessa. ©1998-2024 Odessa, Inc. All Rights Reserved.



## Document revision history.

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Description | Version | Author |
| 3 Feb 2020 | Document Created | 1.0 | DBA Team |
| 26 March 2024 | Added DB growth details | 2.0 | DBA Team |
|  |  |  |  |
|  |  |  |  |

Contents

[Document revision history. 2](#_Toc162531389)

[Introduction: 4](#_Toc162531390)

[Database Backups: 4](#_Toc162531391)

[Database Maintenance Jobs: 4](#_Toc162531392)

[Database Setup: 5](#_Toc162531393)

[SQL Configuration Setup: 5](#_Toc162531394)

[Audit Logging: 6](#_Toc162531395)

[Security: 6](#_Toc162531396)

[Performance Monitoring and troubleshooting: 6](#_Toc162531397)

[Database Forecast: 7](#_Toc162531398)

## Introduction:

The Document has steps and processes for SQL server Maintenance specific to OdessaCore & Datawarehouse databases. Document is applicable to both SQL server on Azure and on-premise. For Databases on PaaS (Azure SQL Managed Instances), it is detailed in each section if the recommendation is applicable or not. Some of the changes listed in this document may need downtime and need to be performed during the maintenance window.

## Database Backups:

Odessa recommends having the database in Full Recovery mode and having Full, Differential, Log backups for OdessaCore database. Backup schedule and recovery will vary based on the Organization requirements and their polices. For On-premise and Azure SQL VMs, below is the general recommendation from Odessa.

Full Backup – Weekly  
Differential Backup – Daily  
Log Backup – Every 10 - 30min (Customers may change this based on their requirement)

For Datawarehouse databases, Odessa recommends setting them to Simple recovery mode and having below backups.

Full Backup – Weekly  
Differential Backup – Daily

The above recommendations are not valid for Azure SQL Managed Instances. Backups will be completely managed by Microsoft in SQL Managed Instances.   
<https://learn.microsoft.com/en-us/azure/azure-sql/managed-instance/automated-backups-overview?view=azuresql>

## Database Maintenance Jobs:

Odessa recommends installing Ola Hallengren scripts (<https://ola.hallengren.com>) for scheduling Database Maintenance jobs (For Azure and Onpremise databases)

Below Jobs are to be scheduled:

1. Database Integrity Check
2. Index Rebuild
3. Update Statistics

**Database Integrity Check:**

This job checks for the integrity and health of the databases. Schedule the job as per the Organization standards, it is recommended to run it once a week. Customers may schedule this job differently based on their requirement.

DatabaseIntegrityCheck - SYSTEM\_DATABASES  
DatabaseIntegrityCheck - USER\_DATABASES

**Index Rebuild:**

Odessa recommends the Rebuild Index job to be run every other weekend. Use the below script to change in the job step. Rebuilding an index is an offline operation, job needs to be scheduled during the maintenance window or when there’s no user activity.  
  
EXECUTE dbo.IndexOptimize  
@Databases = 'OdessaMainDBName',   
@FragmentationLow = NULL,  
@FragmentationMedium = 'INDEX\_REORGANIZE,INDEX\_REBUILD\_ONLINE,INDEX\_REBUILD\_OFFLINE',  
@FragmentationHigh = 'INDEX\_REBUILD\_ONLINE,INDEX\_REBUILD\_OFFLINE',  
@FragmentationLevel1 = 20,@FragmentationLevel2 = 30,@UpdateStatistics = 'ALL'

**Update Statistics:**  
Odessa recommends Update Statistics to be run on daily basis. Use the script below to change in the job step. Schedule the job after the business window or when there’s low user activity.

EXECUTE dbo.IndexOptimize @Databases = 'OdessaMainDBName',  
@FragmentationLow = NULL , @FragmentationMedium = NULL , @FragmentationHigh = NULL ,  
@UpdateStatistics = 'ALL' , @OnlyModifiedStatistics = 'Y' ,@LogToTable = 'Y'

Alerting the respective teams for job failures can be done using monitoring tools or with SQL DBmail.   
It is recommended to configure alerts and troubleshoot/fix the failures for the above jobs.

## Database Setup:

1. Ensure Database files and Tempdb files are placed as per the Microsoft Best Practices. Data files, Log files and Tempdb files are to be configured in different drives.  
   <https://learn.microsoft.com/en-us/sql/relational-databases/policy-based-management/place-data-and-log-files-on-separate-drives?view=sql-server-ver16>
2. Stripe the tempdb to multiple files, Number of files should be same as number of logical processors (with a maximum of 8 data files). Increase the initial size of each tempdb data file (size of tempdb can be upto 50% of total database size)

<https://learn.microsoft.com/en-us/sql/relational-databases/databases/tempdb-database?view=sql-server-ver16>

1. For Azure SQL VMs, please see below for Microsoft recommendations.

<https://learn.microsoft.com/en-us/azure/azure-sql/virtual-machines/windows/performance-guidelines-best-practices-checklist?view=azuresql>

The above recommendations are not applicable for Azure SQL Managed Instance.

## SQL Configuration Setup:

1. Execute the following command in Odessa\_monitorv2 DB (including replica database instances). This will ensure the next 5 points are set to the recommended value:

Exec ChangeSQLSettings 'OdessaMain'

1. Maximum Memory allocated: Ensure SQL server is allocated up to 85% of total memory of the server. If there are other applications on the database server, consider reducing the memory allocated to 75-80%.

(Only applicable for Onpremise and Azure SQL Server VMs)

1. Cost of threshold parallelism: Change the cost of threshold parallelism to 50
2. Maximum degree of parallelism: Change the MaxDOP values to half the number of logical processors (maximum value to 8 if there are more than 16 processors). If SQL server is on server with 2 cores with each core having 8 logical processors, change MaxDOP to 8 from 0.

<https://learn.microsoft.com/en-US/sql/database-engine/configure-windows/configure-the-max-degree-of-parallelism-server-configuration-option?view=sql-server-ver16>

1. Ensure Database compatibility mode is set to the latest.
2. Ensure OdessaCore database has “Read Committed Snapshot” enabled.

select name, is\_read\_committed\_snapshot\_on   
from sys.databases   
where name =db\_name()

1. Enable Database Instance File Initialization (Only applicable for Onpremise and Azure SQL Server VMs)

<https://learn.microsoft.com/en-us/sql/relational-databases/databases/database-instant-file-initialization?view=sql-server-ver16>

1. Change/Increase the initial size of data and log files to higher values. Change the auto incremental value of data file and log files. Data file autoincrement value to 250mb and log file value to 150mb. Adjust these based on daily growth.
2. Ensure the fillfactor for all the indexes are less than 99. Compression for large table indexes also needs to be enabled. Run the below command to change fillfactor and compression settings for the tables.

Exec Changefillfactor

## Audit Logging:

* For SQL VMs and Onpremise systems, ensure “Both Successful and Failed Logins” are enabled at SQL Instance level.  
  For SQL Managed Instance, create a Server Audit to add both “FAILED\_LOGIN\_GROUP” and “SUCCESSFUL\_LOGIN\_GROUP” and route it to Storage account or Event hub.
* Ensure CDC and Audit features are configured and enabled to monitor data level changes for any of the config tables.

## Security:

For Azure and Onpremise systems, ensure the following:

1. Disable sa login
2. Change the dbowner of the databases from individual users to sa
3. Disable guest account from user databases.
4. Grant elevated access only to Administrators
5. Perform Quarterly Audit for Logins
6. Limit the use of SQL Authentication for individual users (Use Windows/AAD)

## Performance Monitoring and troubleshooting:

Below sections are valid for both Azure and Onpremise SQL servers.

**Enable Querystore:**

Odessa recommends enabling Querystore, to help identify the root cause for performance issues (if any).   
Below command will enable Querystore feature. Storage size is being changed to 1GB here, Customers with large database size can increase the size limit. (up to 10% of the data file size)  
USE [master]  
GO  
ALTER DATABASE [OdessaMainDBName] SET QUERY\_STORE = ON;  
ALTER DATABASE [OdessaMainDBName] SET QUERY\_STORE (OPERATION\_MODE = READ\_WRITE, MAX\_STORAGE\_SIZE\_MB = 1024,INTERVAL\_LENGTH\_MINUTES = 60,   
QUERY\_CAPTURE\_MODE = AUTO);  
GO

**Windows Performance Counters:**

For On-premise and Azure SQL VMs, windows performance counters can be created and scheduled to get the server level metrics captured. The attached document has steps to create a new data collector set with counters, which can give Server level and Database level metrics. (This section is not applicable for Azure SQL Managed Instances).



**Health Monitoring:**

Setup a SQL agent job to load Odessacore database query metrics into Odessa\_monitorv2 database. Add the below SQL statement to the job and schedule it to run it Quarterly. It loads Index usage statistics, Query plans of slow running queries and other database related metrics into Odessa\_monitorv2 database.

Exec Getdbstats

Take a backup of this database and share it with Odessa for analysis.

## Database Forecast:

Database growth forecast can be seen through any external monitoring tool. If the Customer don’t have any monitoring tool installed, getting the backup size from msdb can give us approximate size. Full backup size would roughly correspond to database size and its growth. Below command can be used to check the size of the full backups in last 6 months. By comparing the current Data file size with Backup size, growth can be estimated.  
  
Declare @DBName sysname = ' OdessaMainDBName '  
Select msdb..backupset.database\_name [DBName],   
msdb..backupset.type, ceiling (msdb..backupset.backup\_size/(1024\*1024\*1024)) as [BackupSize\_GB], msdb..backupset.backup\_start\_date, msdb..backupset.backup\_finish\_date,  
msdb..backupset.user\_name From msdb..backupset   
Where msdb..backupset.backup\_start\_date > getdate () - 180 and type='D'   
and db\_id(msdb..backupset.database\_name) = db\_id(@DBName)   
Order by msdb..backupset.backup\_start\_date desc

**Get Long Running Queries History from Querystore:**

Below Query will list out top slow running queries sorted by duration along with Query id and plan id. Any regressive query can be pinned or forced with a better execution plan id.  
  
SELECT TOP 50 rs.avg\_duration/1000000 as [AvgDurationSec],   
rs.max\_duration/1000000 as [MaxDurationSec],  
Substring(qt.query\_sql\_text,1,10000) [QueryTxt],q.query\_id,p.plan\_id,  
count\_executions [ExecutionCount],object\_name(q.object\_id) as [objectname], rs.avg\_logical\_io\_reads,rs.avg\_physical\_io\_reads,  
rs.avg\_tempdb\_space\_used/128 as [AvgTempdbSpaceUsedMB],rs.avg\_dop,  
rs.avg\_query\_max\_used\_memory/128 as [AvgQueryMemoryGrantMB],avg\_cpu\_time,  
rs.last\_execution\_time [QryCompletionTime],TRY\_CONVERT(XML, [p].[query\_plan]) AS [QueryPlan]  
FROM sys.query\_store\_query\_text AS qt  
JOIN sys.query\_store\_query AS q ON qt.query\_text\_id = q.query\_text\_id  
JOIN sys.query\_store\_plan AS p ON q.query\_id = p.query\_id  
JOIN sys.query\_store\_runtime\_stats AS rs ON p.plan\_id = rs.plan\_id  
WHERE rs.last\_execution\_time > DATEADD(MINUTE, -60, GETUTCDATE())  
ORDER BY rs.max\_duration DESC;

<https://learn.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/query-store-stored-procedures-transact-sql?view=sql-server-ver15>

**Get Current Long Running and Blocked Queries:**

Odessa\_monitorv2 db as well as Odessa Core DB has “sp\_whoisactive” SP. Running the SP can give the list of long running sessions and blocking details.

EXEC sp\_WhoIsActive

@find\_block\_leaders = 1

,@get\_plans=2

,@sort\_order = '[blocked\_session\_count] DESC'

,@output\_column\_list='[dd hh:mm:ss.mss] [session\_id] [sql\_text] [sql\_command] [wait\_info] [blocking\_session\_id] [blocked\_session\_count] [open\_tran\_count][status] [database\_name] [CPU] [tempdb\_allocations] [tempdb\_current] [login\_name] [reads] [writes] [physical\_reads] [used\_memory] [host\_name] [program\_name] [start\_time] [login\_time] [request\_id] [collection\_time][tran\_log\_writes] [tran\_start\_time] [additional\_info] [query\_plan] '

**Get Query Memory Grant and Waits:**

Run the below query in Odessa\_monitorv2 db to get the query memory grants and wait statistics.

Exec Getdbperformance 'memoryqueries'

Exec Getdbperformance 'waits'