# Installation and User Manual


September 2025
Version 5.31

**DISCLAIMER**

## Document Revisions

| Date | Version Number | Document Changes |
|---|---|---|
| 14-07-2021 | 1.0 | Initial Release |
| 14-01-2022 | 1.1 | Second Release |
| 16-05-2022 | 1.2 | Third Release |
| 18-07-2022 | 1.3 | Fourth Release |
| 14-09-2022 | 1.4 | Fifth Release |
| 03-01-2023 | 1.5 | Sixth Release |
| 05-04-2024 | 1.6 | Seventh Release |
| 16-10-2024 | 1.7 | Eight Release |
| 14-01-2025 | 1.8 | Ninth Release – Section for Performance Insights™ added |
| 21-02-2025 | 1.9 | Minor changes |
| 03-04-2025 | 2.0 | Extension compatibility is updated |
| 29-05-2025 | 2.1 | Image of Blocking and Deadlock errors are added |
| 21-06-2025 | 2.2 | BC/Dynamics NAV component diagram is updated |
| 06-09-2025 | 2.3 | Fuzzy  filtering of users in  Performance Insights™ is added |

# Table of Contents

# 1 PREFACE

## 1.1 Description of the User

SQL Mantra Tools'© user manual is intended for system administrators, IT Managers, SQL DBA, Dynamics AX/Dynamics NAV/Business Central Consultants and developers. It is assumed that the user is familiar with SQL Management Studio and executing sql stored procedures as well as key setups in SQL Server, ERP database such as Dynamics AX, Dynamics NAV and Business Central database. Understanding of the interaction between Dynamics NAV CSIDE/AL code with the underlying SQL DBMS environment will be an advantage.

## 1.2 Obtaining Documentation and Information

### 1.2.1 Internet

The latest version of the documentation is available for download at the following address: https://www.sqlmantratools.com

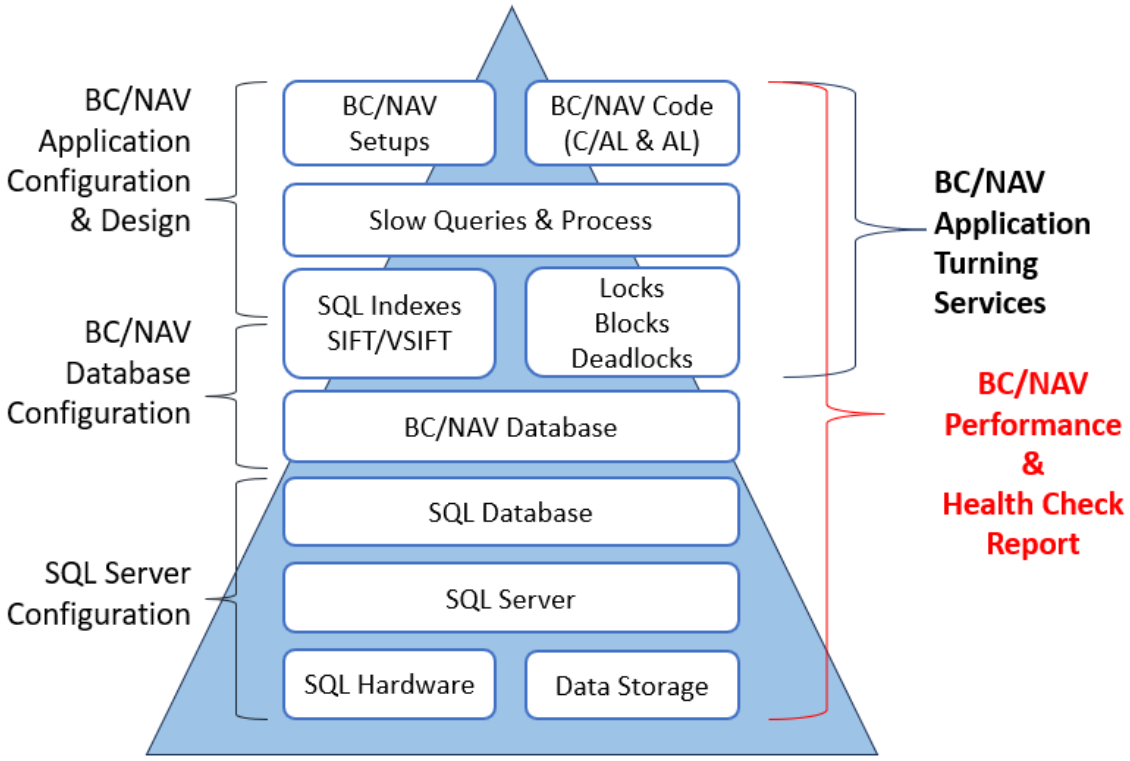
### 1.2.2 Documentation Feedback

When you are reading SQL Mantra Tools' product documentation, any comments can be submitted through "Contact us" on our website (https://www.sqlmantratools.com/contact-us). Comments can also be sent to sales@sqlmantratools.com.

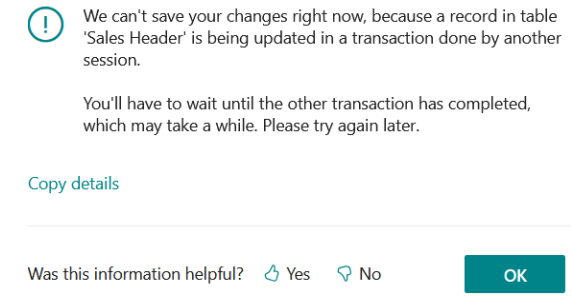
We appreciate your comments and feedbacks.

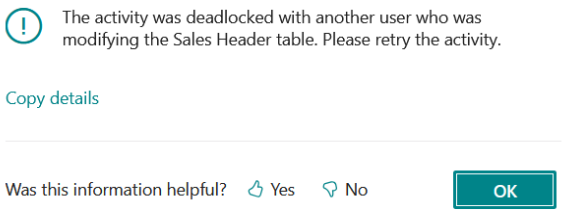
# 2 Performance Tuning Strategy

We take a top-down approach when it comes to improving systems' performance. The ERP system such as Dynamics NAV or Business Central is made out of many components; they are connected to each other in layers. Performance issues may exist in one or many of the layers and components. Unless you address the underlying issues and weakness in all of these components you will not effectively address the performance issue. Therefore, it is crucial to identify what problem exists in each of these components correctly. As part of our System Health Check Service, we analyse and look at all the following components. As experts, we have specialised knowledge about all the components and the complex interaction which takes place between them.

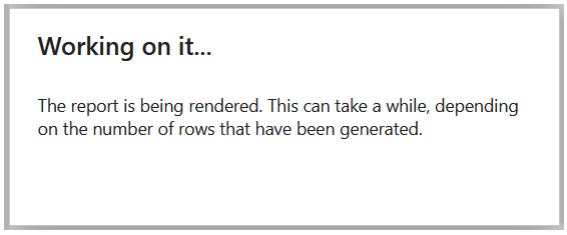



Using our unique array of SQL Mantra Tools, we can pinpoint the performance issues better compared to many other software, solving the issue once and for all for a fraction of cost. As users you may get frozen screens, slow system response, blocking and deadlock error messages such as the followings in your ERP application and quite often you will have no clue about what has caused this problem. Our tools will identify the source of the problem (to the application code level) and record such issues 24/7, making the diagnostic process simpler.

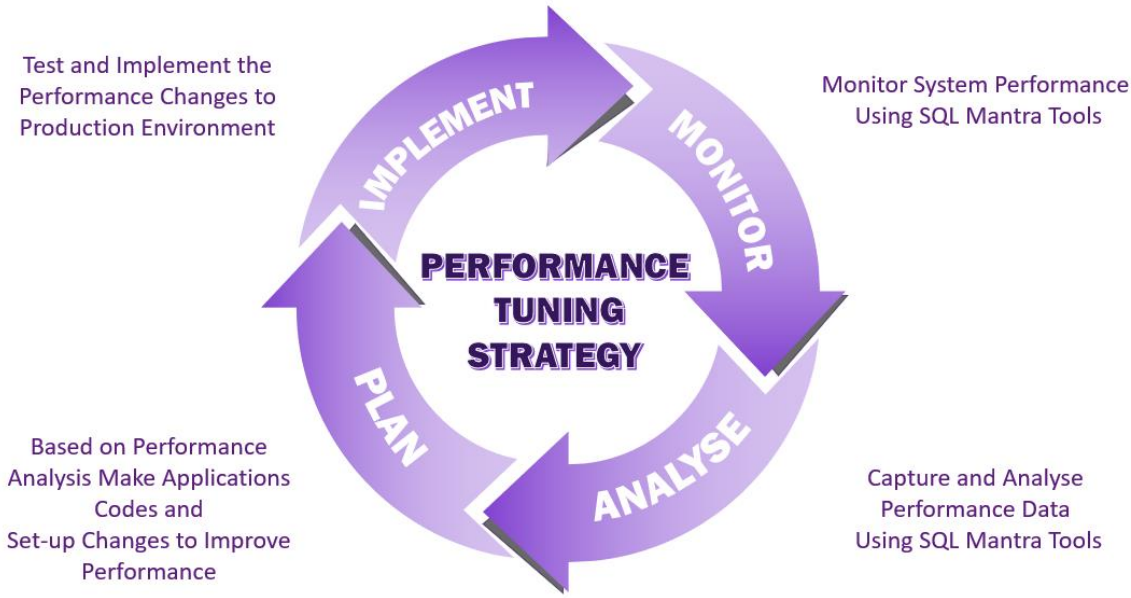







Whilst tuning the system, we follow a unique performance tuning strategy (see below) where we go through the phases Monitor, Analyse, Plan and Implement. Depending on the complexity of the issue, we may have to go round this cycle multiple times. For the process to succeed, you need a proper performance monitoring and analysing tool as well as the correct interpretation of the performance data and years of experience in performance tuning to come up with the correct solution. We have all of these under one roof, simply visit our website https://www.sqlmantratools.com/contact-us and let us know your requirements and we can take care of the rest.


Test and Implement the Performance Changes to Production Environment
IMPLEMENT
Monitor System Performance Using SQL Mantra Tools
MONITOR
PERFORMANCE TUNING STRATEGY
PLAN
Based on Performance Analysis Make Applications Codes and Set-up Changes to Improve Performance
ANALYSE
Capture and Analyse Performance Data Using SQL Mantra Tools

# 3 Description of the product

## 3.1 Purpose of the Product

SQL Mantra Tool can be used to troubleshoot performance issues in any application running on Microsoft SQL Server environment hosting a variety of systems such as ERP system, BI database to website. It also has an add-on for Dynamics NAV and Business Central application. Once activated, the SQL Mantra Tool will capture the application user name and application code involved in various performance incident logs, which makes the SQL Mantra Tool a unique tool in the market.

SQL Mantra Tools' Performance Insights Product can analyse and report performance bottlenecks and software weaknesses across all the extensions in a Business Central 365 instance hosted in Microsoft SaaS environment. It can be also used to analyse and report performance issues in latest versions of Business Central on-prem installations as well.

SQL Mantra Tool can also analyse SQL Server setups, SQL Database setups against industry standard best practice guidance and will recommend the necessary action to rectify with best performance in mind.

SQL Mantra Tool can also be used to optimise indexes for a busy big database. Thanks to its innovative adaptive algorithm to optimise fill factor, it will proactively maintain index to minimise fragmentation and will reduce database space.

SQL Mantra Tool also comes with a unique scheduler feature for SQL Express, extending SQL Agent feature for SQL Express. Using this, system administrators can automate many SQL related task in SQL Express Edition.

SQL Mantra Tool is either licensed to a specific SQL Server instance / cluster or to all the SQL Servers running in a specific domain. The following SQL code can be used to establish the SQL Server, Cluster and the domain details. Run this code and pass the details of the environment when requesting a license (both evaluation and production license).

```
DECLARE @ClusterName VARCHAR(MAX);
SET @ClusterName = '';
IF ISNULL(serverproperty ('IsHadrEnabled'),0) = 1
    SET @ClusterName =
    (
        SELECT TOP 1 c.name
        FROM sys.dm_hadr_availability_replica_cluster_states s
             JOIN sys.availability_groups_cluster c ON s.group_id = c.group_id
        WHERE s.replica_server_name = @@SERVERNAME
    );

PRINT 'Domain Name :- ' + DEFAULT_DOMAIN()
PRINT 'Server Name :- ' + @@SERVERNAME
PRINT 'ClusterName :- ' + @ClusterName
```

## 3.2 Product Overview

SQL Mantra Tool comes with the four modules for its on-prem offerings. They are:


**Maintenance**


**Performance Logging**


**System Analysis**


**Scheduler**

Each of these modules are packaged into various products. Please visit https://www.sqlmantratools.com/products for more details.

### 3.2.1 Maintenance Module

In a very busy and big OLTP database such as Dynamics NAV, Dynamics AX, and Business Central database data is quite often unevenly distributed. Typically, 90% of the data will be held in10% of the tables and quite often these tables are very active with lots of inserts, updates, deletes and reads operations performed on them. Due to these characteristic, vast majority of the index rebuilding strategies normally used in the wider SQL communities will not work to keep the index in a healthy state. SQL Mantra Tools has come up with a unique index maintenance strategy specifically geared towards such busy and big OLTP databases. Even though it is geared towards busy and big OLTP database, it will equally work well for a variety of database and workloads such as very small databases, web back-end database to BI databases. Some of its key features can be found below. Without proper index maintenance, performance will suffer in the long run. Our maintenance strategy is “prevention is better than cure”. SQL Mantra Tools’ unique index maintenance package will proactively maintain indexes and will take care of routine maintenance such as Backups, checking database for errors and many more with automated alerting feature giving organisation piece of mind.

The following routines are included in our out-of-the-box “set and forget” maintenance module:

- ✓ Index Maintenance
- ✓ Database Size Maintenance
- ✓ Check Database for errors
- ✓ Clearing SQL Server logs to keep the server healthy
- ✓ Full Database Backups
- ✓ Transaction Log Backups
- ✓ Housekeeping routine keeping the Dynamics NAV/Business Central database healthy

The following are some of the key features of “set and forget” index maintenance module.

- Only rebuild index which are fragmented more than 5% on a daily basis. The rest of the index will be rebuilt proactively in a round robin fashion covering different portions of database each day.
- Routine will not reorganise index as the reorganising process is single threaded and will not use parallel operation, hence the execution is much slower.
- Options to override index selection / exclusion on each day’s index rebuild load.
- Fill factor settings are adaptively changed and optimised based on index’s volatility, data type, size and bad page splits.
- Option to terminate/limit the execution of the re-index after a set time period, so that it will not interfere/block busy/critical process.
- Option to log the details of re-index process (such as what index detail, start and end time, page split, old and new fill factor, etc…).
- Ability to re-index using multiple parallel re-index processes (only recommended for really large databases which has a very good disk system) to reduce the re-index time window. The concurrent re-index sessions will coordinate and share the work load equally among themselves and they make sure they do not block each other by working on different tables and its dependent index.
- Option to rebuild index online (subject to having SQL Enterprise Edition).
- Option to target re-index process on any database within the same SQL Server instance.
- If an index is static, it will not be rebuilt at all. The routine will prioritise and rebuild index which have the most fragmentation first.
- If time is limited in a day, option to only re-index badly fragmented index (i.e. more than 5%).
- While rebuilding index, options to change the recovery model to bulk logged or simple to minimise the transaction log space usage. Once completed the original recovery model will be restored.
- Fully configurable routine to Remove unused index.

### 3.2.2 Performance Logging Module

SQL Mantra Tools Performance Logging module will record performance related KPIs 24/7 automatically and will work as a “black box” providing vital inside information for a performance incident. This makes the performance logging module the heart of a busy multiuser system. With the analysis module the logging module would play a major role in providing invaluable inside information for performance tuning and troubleshooting and will lead to extending systems life and its scalability.

- ✓ Blocking*
- ✓ Deadlocks*
- ✓ Slow Queries
- ✓ Index Usage
- ✓ Disk Response
- ✓ CPU Usage

Note:

* With an optional feature for Dynamics NAV and Business Central (versions NAV2013 to all BC versions) SQL Mantra Tool will capture the Dynamics NAV/BC user detail as well as the Dynamics NAV/BC Application code details involved in each of the performance incident in the logs making it possible to identify and tackle performance problems and bottlenecks with pin-point accuracy which will make it a must-have tool for any Dynamics NAV/BC installation.

### 3.2.3 System Analysis Module

SQL Mantra Tools System Analysis Module will analyse how the system is configured and performing and will provide best practice recommendation to remedy any defects. It will also analyse the performance KPI recorded by the Performance Logging module and will summarise the data in an easy-to-understand user friendly interface.

The following areas are analysed:

- ✓ SQL Server Setup Analysis
- ✓ Tempdb Setup Analysis
- ✓ Server Memory Analysis
- ✓ CPU Utilisation Analysis
- ✓ SQL Server Wait Analysis
- ✓ Maintenance Job Analysis
- ✓ Database Setup Analysis
- ✓ Scheduler Job Analysis
- ✓ Deadlock Analysis
- ✓ Database Size History Analysis
- ✓ Table Size Analysis
- ✓ Index Usage Analysis
- ✓ Slow Query Analysis
- ✓ Blocking Analysis
- ✓ Dynamics NAV Setup Analysis
- ✓ Index Fragmentation Analysis
- ✓ Slow Application Code Analysis
- ✓ Slow SQL Stored Procedure Analysis
- ✓ Live Blocking Analysis

### 3.2.4 Scheduler Module

The Scheduler Module of SQL Mantra Tools will provide a much-needed scheduler facility to any SQL Express Edition. Using this module, one could schedule to run any SQL command, Stored procedure or SQL Script targeting any database within the SQL Server instance with an option to send email alerts to IT help desk when the code fails to execute. The Scheduler Module will take the strain of scheduling routine maintenance task such as taking backups, rebuilding index, running any business-critical task in a reliable way on a SQL Express Edition.

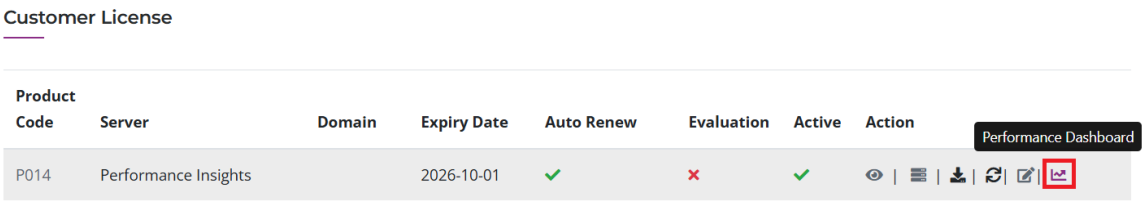
The following are the key features of Scheduler for SQL Express:

- ✓ Set and forget SQL based scheduler
- ✓ Could help to automate a range of tasks in an environment giving similar functionality to SQL Agent
- ✓ Fully configurable multi-threaded scheduling feature
- ✓ Support for one off and recurring tasks – features to run every minute to specific day(s) in a week
- ✓ User can specify any SQL command, Stored procedure or SQL Script to run
- ✓ Ability to target any database within the SQL Instance
- ✓ Ability to send e-mail notification when a task fails
- ✓ Comes with predefined / customisable templates to speed up deployment
- ✓ Must have feature for POS/Till database

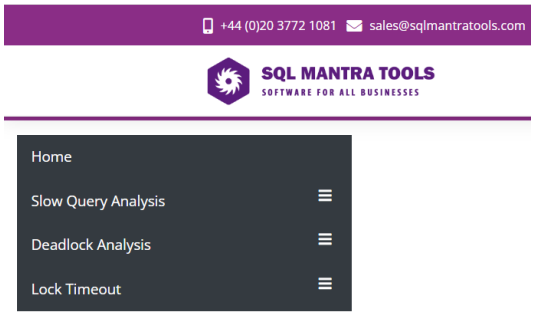
### 3.2.5 Performance Insights

Performance Insights© is designed to give unprecedented insights into Business Central 365 SaaS environment detailing performance issues such as blocking, deadlock activities and slow running processes giving crucial information such as SQL Code, AL Code and the users involved via the SQL Mantra Tools web portal with interactive graphs and tables. Performance Insights can be also used on-prem implementation of Business Central (BC16.1 or above) as well. Once configured (see Section 3.10 How Setup Performance Insights) customers can login to the SQL Mantra Tools website and access the performance data using a variety of devices from laptops to mobile devices accessing crucial performance data on the move, in a click of a button. Partners can also have access to the same performance charts and tables for all their customers via the SQL Mantra Tools website as well.

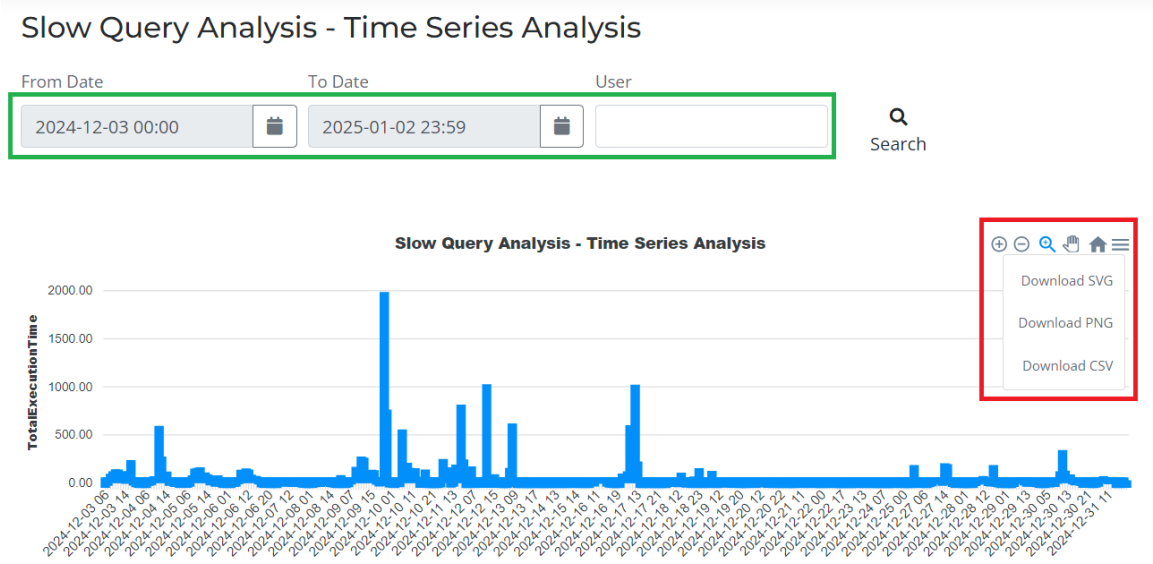
Once logged into the SQL Mantra Tools web portal, users need to click the customer in question under the "Manage Customers" section. Then they need to click "Manage Customer License". In the Customer License section, they need to click "Performance Dashboard" button (see below).



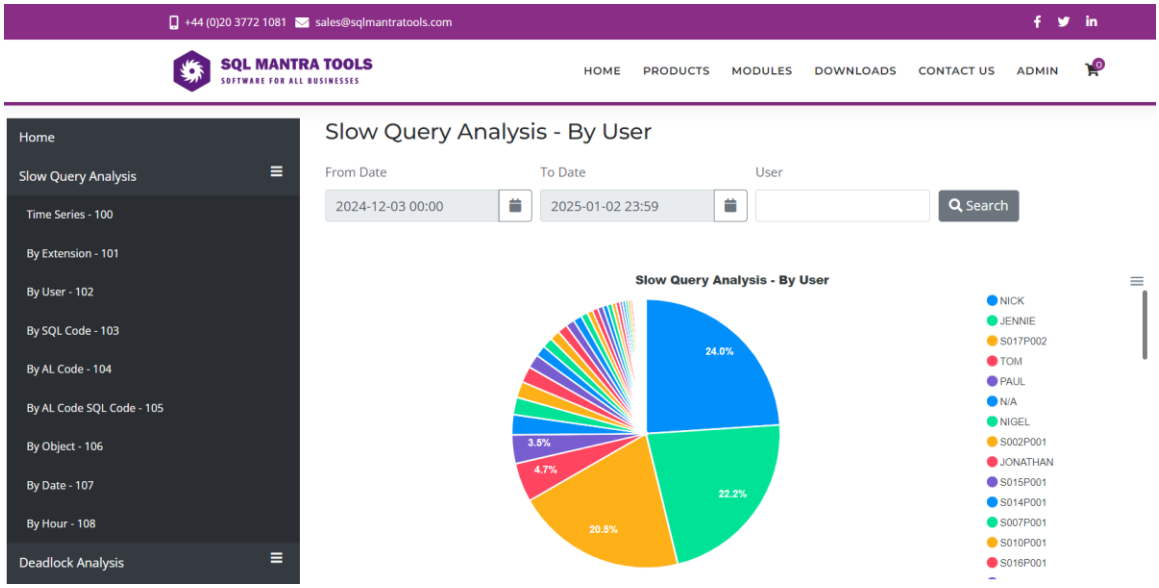

Once authenticated, users will see the performance dashboard menu, displaying all three areas where one can have performance issues. Once clicked each area will expand and will show further menus. They are typically Timeseries Analysis, Analysis by Extension, Analysis by User, Analysis by SQL Code, Analysis by AL Code, Analysis by AL Code & SQL Code, Analysis by Object, Analysis by Date and Analysis by Time.



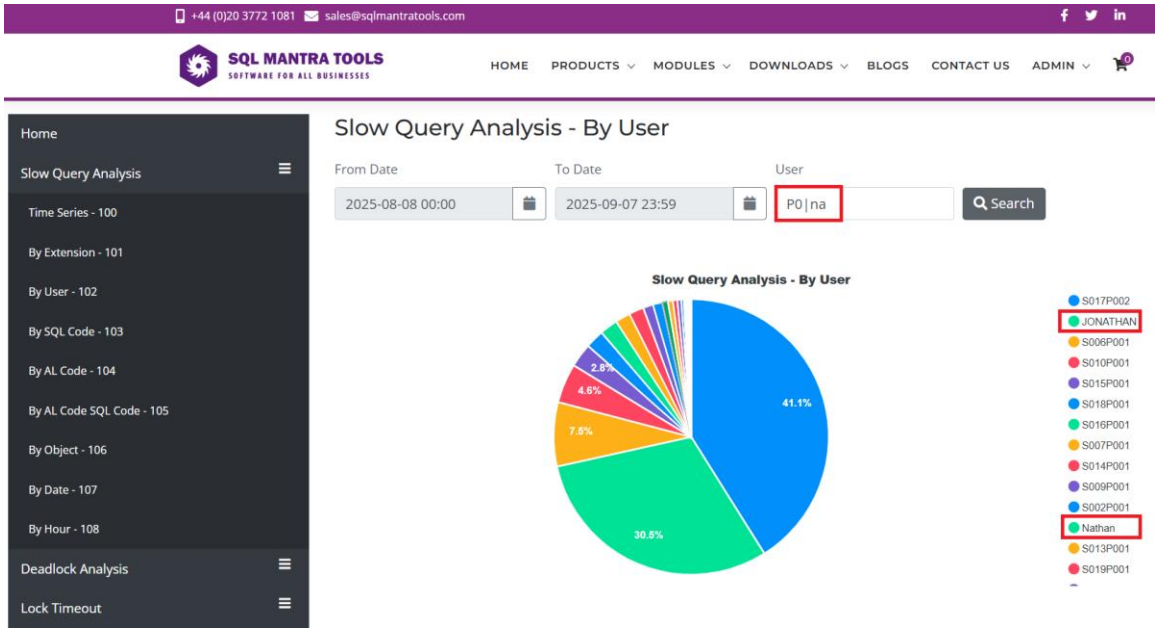

In each of the analysis page, users can set the necessary start date / time as well as end date / time along with a user filter to filter the data. Once the filters are changed, users need to click the "search" button to refresh the graph and data presented for the specific criteria. Further, the charts can be downloaded in Svg, Png and Csv formats and could be shared with a wider audience such as executive board, support desk, operational team, development team and can be included in various presentations within the organisation. See below for example.



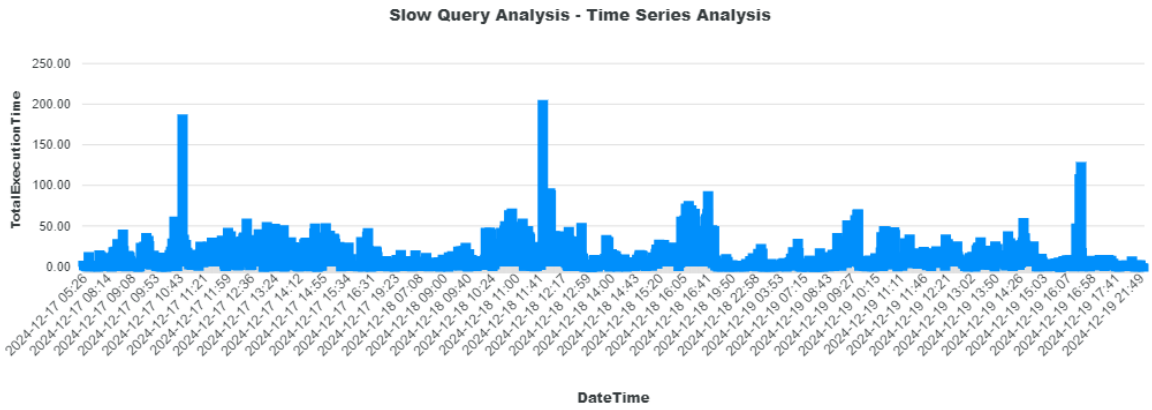

One of the unique features of Performance Insights is its ability to report the actual Business Central User in its analysis. See below for example.



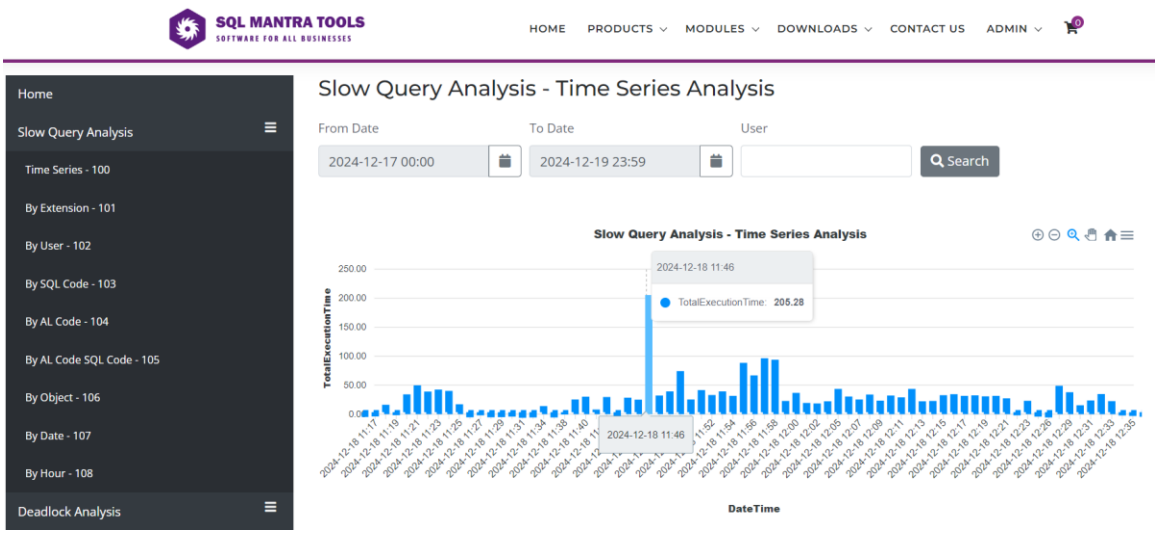

Further, users can apply a wild card filter to select one or more users, separated by "|". System will perform a case insensitive fuzzy filtering to match the users. This is a unique and powerful feature inbuilt in Performance Insights. See example below. In this example by specifying "P0|na" as the user filter, users can filter performance data for all the POS users as well as any user with "na" in their name and can focus performance issues experienced by them. Whilst having the same filter, when the user clicks the "By AL Code", "By SQL Code", "By AL Code SQL Code" they will get the AL Code, SQL Code behind the slowness of the system for all the POS. Similarly, by filtering user by "S012" one can see performance issues for store S012. The same concept can be applied to all the analysis.



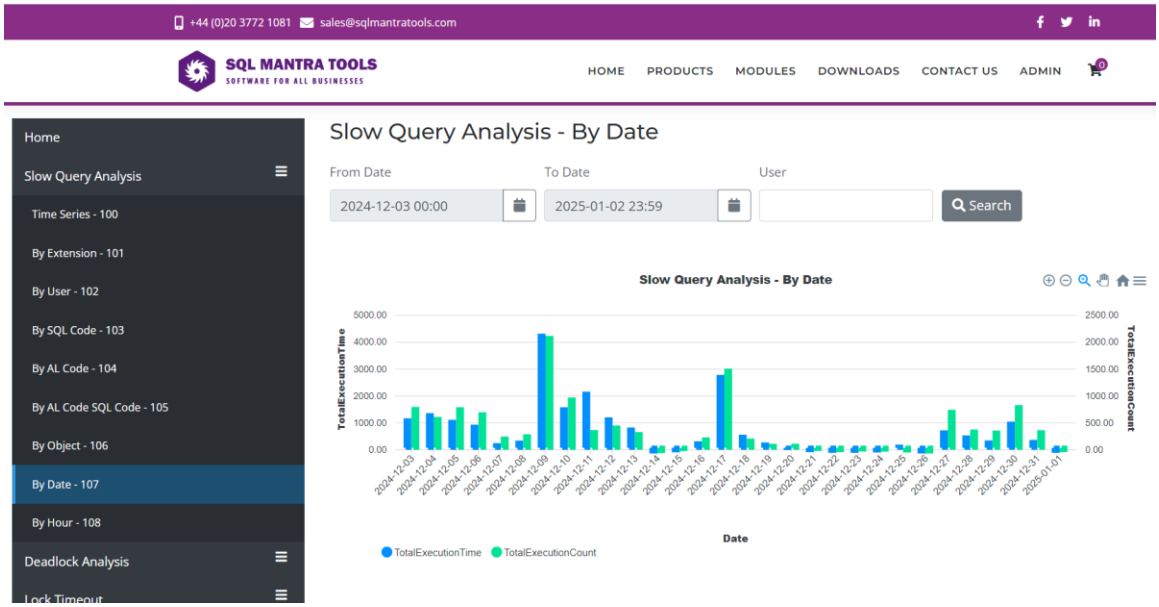

When first loaded, time series analysis will give an overview of system's performance summarised into day and into hourly buckets, listing the total execution time in seconds in the customers' own time zone, once again a unique feature of Performance Insights. See below for an example.



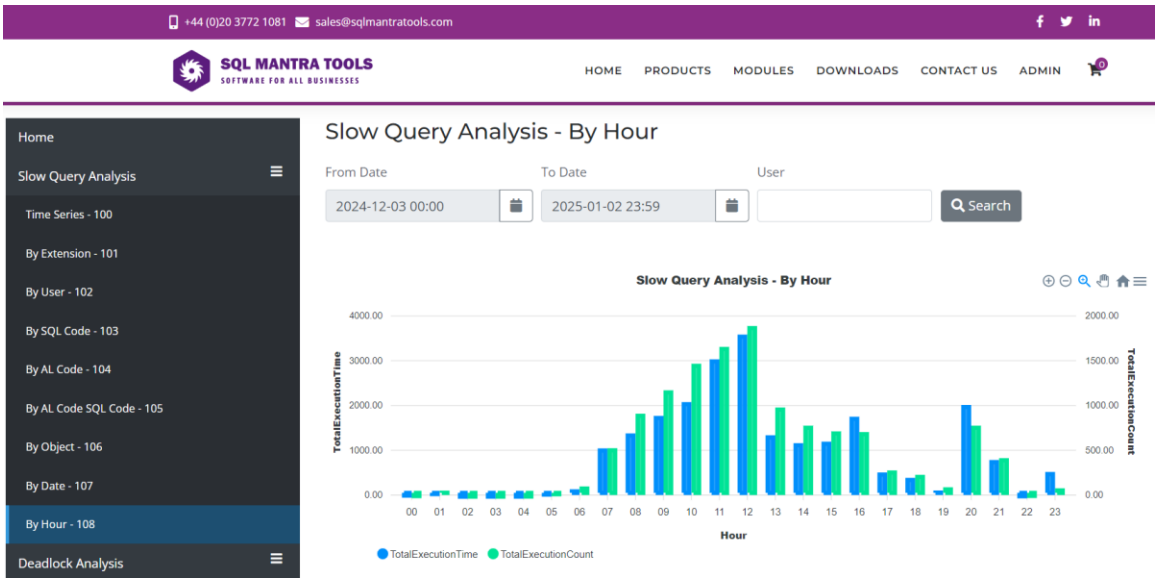

Users can zoom into the interested portion of the graph by drawing a rectangle around the area. Further, users can also focus onto a range of dates / times by specifying on the "From Date", "To Date" filter using the date/time picker. The new filter can be applied by clicking the "Search" button. If the date range (ie difference between "From Date", "To Date") is less than 3 days, the analysis will be switched to day and minute buckets. See below for example. By pointing to any bar in the graph, users can see the date / time detail as well as how slow the system is during that time.



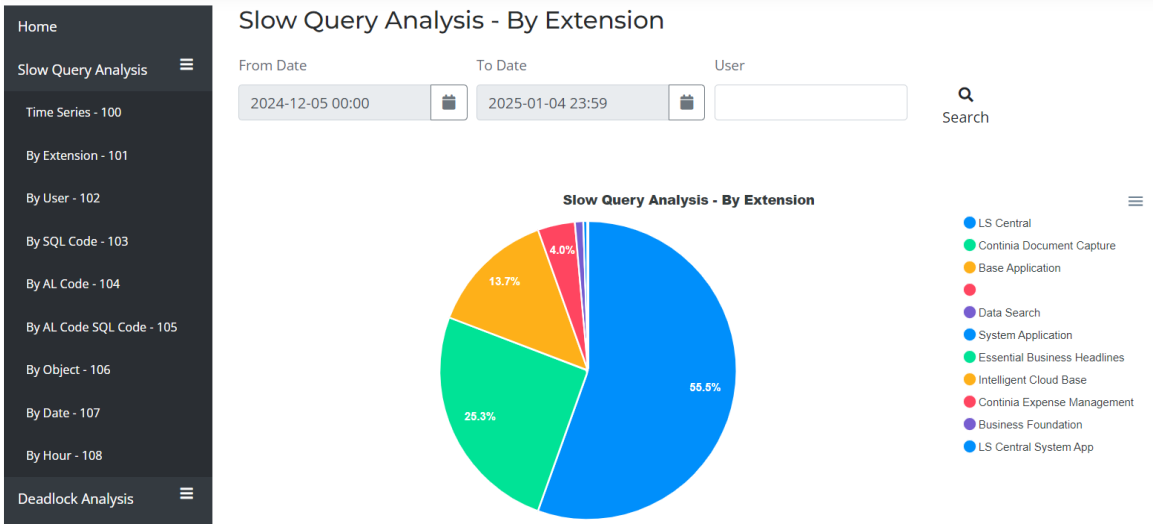

Similarly, Analysis by Date will give slow query details by date. See below for example.



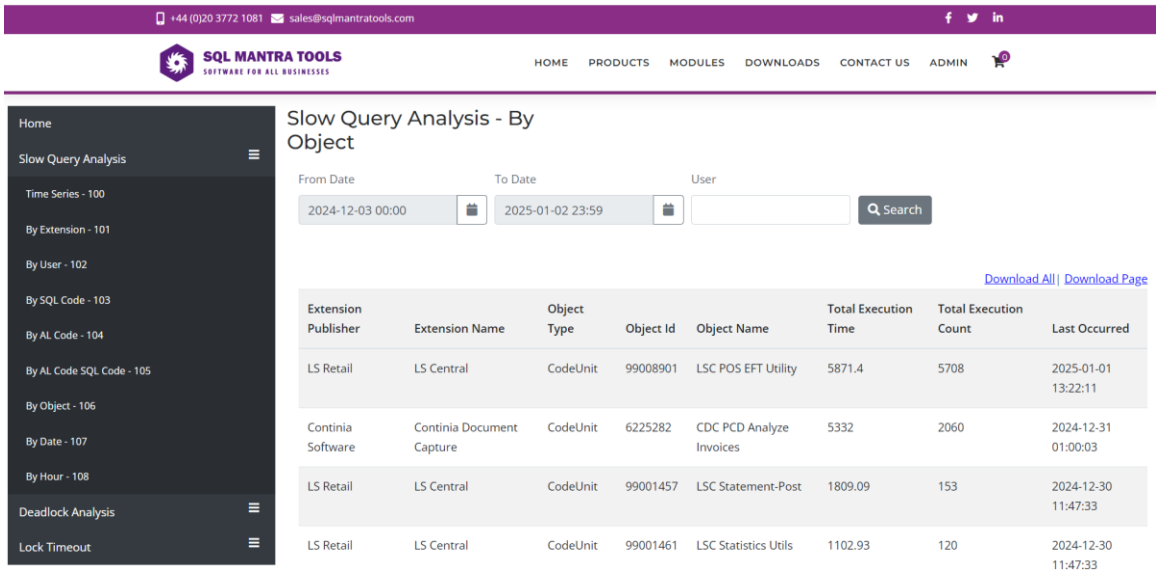

Slow query analysis by Hour will give slow query details by hour. See below for example.



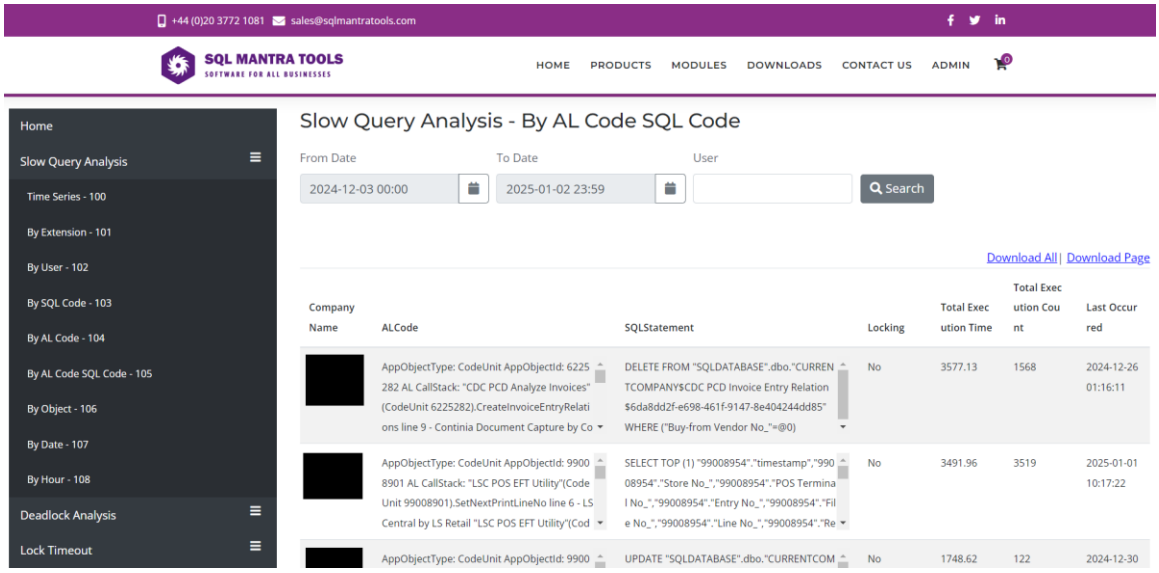

Slow query analysis by extension will give system slowness broken down by extension in Business Central. See example below.



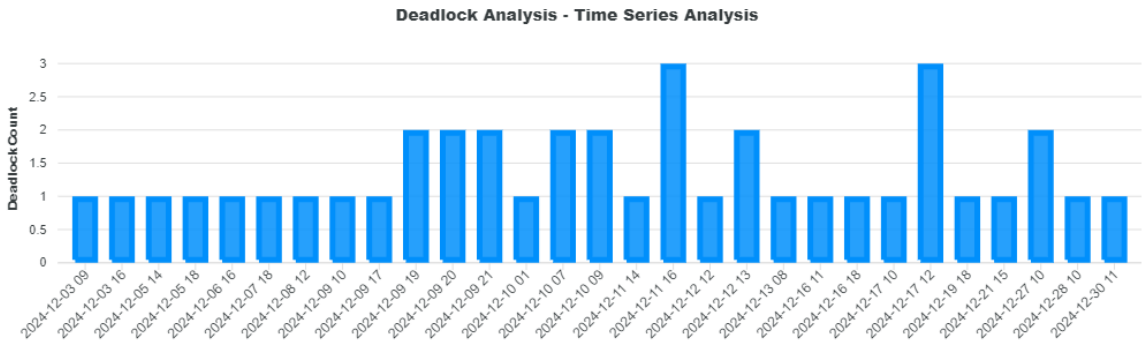

Slow query analysis by object will give system slowness broken down by application area and objects so that one can plan, focus and prioritise the performance tuning tasks better. See example below.

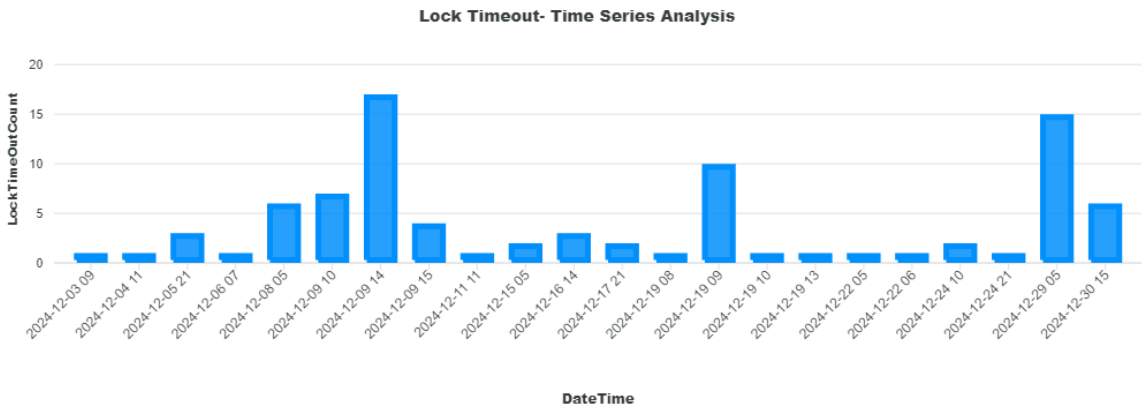



Further Slow query analysis by AL Code / SQL Code will help developers to identify the problematic code and application area with pinpoint accuracy. See below for example. Further the tables can be downloaded as an excel document and can be shared with the developers and extension authors. By clicking on to the column header, users can sort the data in table as they wish. In the example below, the table is sorted on “Total Execution Time” column.




Similar to slow query analysis, once selected Performance Insights will show performance data and analysis related to Deadlock and Lock timeout (blocking) as well. See example below.



## 3.3 General SQL Stored procedure Definition

### 3.3.1 Tool_GetLicense

Run the following SQL code against the "Tool" database to check the license. Pay attention to server edition, status, expiry date and the server. Make sure the SQL Mantra tool is licensed to the correct server or cluster. Please note each module is individually licensed. If the expiry date is blank, the particular module is not licensed.

```
EXEC [dbo].[Tool_GetLicense]
```

### 3.3.2 Tool_DeleteLog

Run the following SQL code against the "Tool" database to delete old log entries from the Tool database. Normally this SQL Stored procedure is called in SQL Agent Job called in "Tool : Delete Old Log".

The parameters are explained below.

```
EXEC [dbo].[Tool_DeleteLog]
   @RetentionDays
```

@RetentionDays :- Specify the number of days to retain data in various performance incident log tables in the tool database. Default value is 30 days.

### 3.3.3 Tool_Activate

Run the following SQL code against the "Tool" database to activate the SQL Mantra Tool. Normally this SQL Stored procedure is called as part of the installation process. Once executed you will get a summary of what modules are successfully activated along with detail tasks to follow. The parameters are explained as below.

```
EXEC @RC = [dbo].[Tool_Activate]
   @InstallMaintenance
  ,@InstallLogging
  ,@InstallScheduler
```

@InstallMaintenance :- Set this parameter to 1 to activate maintenance module.
@InstallLogging :- Set this parameter to 1 to activate maintenance module.
@InstallScheduler :- Set this parameter to 1 to activate maintenance module.
Note :- Any activation is subject to License.

### 3.3.4 Tool_DeActivate

Run the following SQL code against the “Tool” database to deactivate the SQL Mantra Tool. Normally this SQL Stored procedure is called as part of the uninstall process.

The parameters are explained as below.

```
EXEC [dbo].[Tool_DeActivate]
   @DeInstallMaintenance
  ,@DeInstallLogging
  ,@DeInstallScheduler
```

@DeInstallMaintenance :- Set this parameter to 1 to deactivate maintenance module.
@DeInstallLogging :- Set this parameter to 1 to deactivate maintenance module.
@DeInstallScheduler :- Set this parameter to 1 to deactivate maintenance module.

## 3.4 Maintenance SQL Stored procedure Definition

### 3.4.1 Tool_Reindex

This stored procedure is licensed under maintenance module. Run the following SQL code against the "Tool" database to rebuild index. Normally this SQL Stored procedure is called in SQL Agent Job called in "Maintenance : <Database Name> – Reindex"

Pay attention to @TargetDatabase parameter as this will determine the target database to rebuild index. The parameters are explained as below.

```
EXEC [dbo].[Tool_Reindex]
   @CurrentBatch
  ,@TotalNoOfBatches
  ,@OnLine
  ,@NewFillFactor
  ,@StopAfter
  ,@TargetDatabase
  ,@RecoveryModel
  ,@IncludeExcludeBatch
  ,@RebuildDisabledIndex
  ,@Debug
```

@CurrentBatch :- Current portion of the database to rebuild index. Set this parameter to 0 to rebuild badly fragmented index only.

@TotalNoOfBatches :- Total number of portions of database used in index rebuild.

@OnLine :- Use online Index rebuild (if supported by the SQL Server Edition).

@NewFillFactor :- Fill factor to use while index rebuild. It is not recommended to use this option under normal circumstances. If this parameter is omitted, fill factor will be optimised adaptively based on the type of fields in the index and its volatility.

@StopAfter :- Set this parameter to stop the execution of the re-index after the set amount of minutes. It's quite a handy parameter if the maintenance window is limited and if you want to ensure the index rebuild finish within a certain time.

@TargetDatabase :- Set the name of the database to rebuild the index using this parameter.

@RecoveryModel :- Recovery model of the database will be set to this parameter value while rebuilding the index. Valid values are 'SIMPLE', 'BULK_LOGGED', 'FULL'.

@IncludeExcludeBatch :- Define the include/exclude batch of set of tables. Tables are defined in the Tool_ReindexIncludeExcludeList table.

@RebuildDisabledIndex :- Set this parameter to 1 to rebuild/enable disabled index.

@Debug :- Set this parameter to 1 to log the detail of index been rebuilt, timing and other debugging details. Details are logged in Tool_ReindexLog table.

The default parameter values are listed below. See extended property of the SQL Procedure for more examples.

@CurrentBatch = 1,
@TotalNoOfBatches = 1,
@OnLine = 0,
@NewFillFactor = 0,
@StopAfter = 0,
@TargetDatabase = '',
@RecoveryModel = 'SIMPLE',
@IncludeExcludeBatch = 0,
@RebuildDisabledIndex = 0,
@Debug = 0

Note :-

1) If another index maintenance job is in place, it is highly recommended to disable it as these jobs are likely to conflict with each other. Further more if you are using the SQL Mantra Tools' index maintenance job you will not need another one.

2) If the database is participating in log shipping or part of always availability cluster, please make the following adjustments to the reindexing job. Please see details below.

   a) If the database is participating in log shipping, set the @RecoveryModel parameter to @RecoveryModel = 'BULK_LOGGED'. It is also necessary to disable SQL Mantra Tools' Transaction log backup job.

   b) If the database is part of always availability cluster, set the @RecoveryModel parameter to @RecoveryModel = 'FULL'.

### 3.4.2 Tool_CheckDatabase

This stored procedure is licensed under maintenance module. Run the following SQL code against the "Tool" database to check the database for any errors. Normally this SQL Stored procedure is called in SQL Agent Job called "Maintenance : <Database Name> - Check Database"

Pay attention to @TargetDatabase parameter as this will determine the target database to check for errors.

```
EXEC [dbo].[Tool_CheckDatabase]
   @TargetDatabase
```

### 3.4.3 Tool_DeleteBackupHistory

This stored procedure is licensed under maintenance module. Run the following SQL code against the "Tool" database to delete old backup history to keep the msdb database healthy. Normally this SQL Stored procedure is called in a SQL Agent Job called "Maintenance : <Database Name> - Check Database"

Pay attention to @TargetDatabase parameter as this will determine the target database to remove the backup history. The parameters are explained as below.

```
EXEC [dbo].[Tool_DeleteBackupHistory]
   @oldest_date
  ,@TargetDatabase
  ,@Type
```

@oldest_date :- This parameter will define the oldest date to delete the backup history.

@TargetDatabase :- Set the name of the database to delete the backup history using this parameter.

@Type :- This parameter will define what type of log backup to delete (e.g., 'L' for transaction log backup, 'D' for database backup). By default, this parameter is set to 'L'. It is not recommended to use 'D' as it will lead to deleting data needed for database growth analysis.

### 3.4.4 Tool_DeleteAutoStats

This stored procedure is a licensed under maintenance module. Run the following SQL code against the "Tool" database to delete auto starts created by SQL server and turn off auto creation and update of auto stats. Normally this SQL Stored procedure is called in SQL Agent Job called "Maintenance : <Database Name> - Check Database"

Pay attention to the @TargetDatabase parameter as this will determine the target database to remove auto stats.

```
EXEC [dbo].[Tool_DeleteAutoStats]
   @TargetDatabase
   ,@DeleteUserCreatedStats
```

@TargetDatabase :- Set the name of the database to delete the backup history using this parameter.

@DeleteUserCreatedStats:- This optional parameter will define whether to delete user created stats or not. Set this parameter to 1 to delete user created stats. Default is 0.

### 3.4.5 Tool_AutoExpandDatabase

This stored procedure is licensed under the maintenance module. Run the following SQL code against the "Tool" database to automatically expand the database when the usages exceed a set percentage, keeping the percentage usage size of the database between the @MinUsedPercent and @MaxUsedPercent. Normally this SQL Stored procedure is called in SQL Agent Job called "Maintenance : <Database Name> - Auto Grow Database". This job works as an early warning system when it comes to disk space shortage.

Pay attention to @TargetDatabase parameter as this will determine the target database to grow. The parameters are explained as below.

```
EXEC [dbo].[Tool_AutoExpandDatabase]
   @TargetDatabase
  ,@MinUsedPercent
  ,@MaxUsedPercent
```

@TargetDatabase :- Set the name of the database to expand.

@MinUsedPercent :- Set the minimum usage percentage of the database. Default value is 70.

@MaxUsedPercent :- Set the maximum usage percentage of the database. Default value is 90.

### 3.4.6 Tool_RemoveAndCreateIndex

This stored procedure is licensed under the maintenance module. Run the following SQL code against the "Tool" database to remove unused index and to create new index in SQL server based on configuration in "Tool_IndexRemovalAdditionList" table. See the extended property of this table for details on how to configure for index removal and index creation. Normally this SQL Stored procedure is called in SQL Agent Job called "Maintenance : <Database Name> - Housekeeping".

Pay attention to @TargetDatabase parameter as this will determine the target database to remove and add index. The parameters are explained as below.

```
EXEC [dbo].[Tool_RemoveAndCreateIndex]
   @TargetDatabase
```

@TargetDatabase :- Set the name of the target database to add/remove index.

### 3.4.7 Tool_ClearSessionEvent

This stored procedure is licensed under the maintenance module. Run the following SQL code against the "Tool" database to remove old and orphaned records from Session Event table in Dynamics NAV/BC database. Even though the Dynamics NAV/BC Service tier should clear old data, in reality, it does not, leading to significant data load in Session Event system table leading to performance issues. Normally this SQL Stored procedure is called in SQL Agent Job called "Maintenance : <Database Name> - Housekeeping".

Pay attention to @TargetDatabase parameter as this will determine the target database to clear the session event data. The parameters are explained as below.

```
EXEC [dbo].[Tool_ClearSessionEvent]
   @TargetDatabase
   ,@RetentionDays
```

@TargetDatabase :- Set the name of the database to clear Session Event.

@RetentionDays :- Number of days to retain the Session Event log. Default value is 7.

### 3.4.8 Tool_ClearChangeLogEntry

This stored procedure is licensed under the maintenance module. Run the following SQL code against the "Tool" database to remove old change log entries in Dynamics NAV/BC database in all companies.

Pay attention to @TargetDatabase parameter as this will determine the target database to clear the change log entries. The parameters are explained as below.

```
EXEC [dbo].[Tool_ClearSessionEvent]
   @TargetDatabase
   ,@RetentionDays
```

@TargetDatabase :- Set the name of the database to clear Change Log Entries.

@RetentionDays :- Number of days to retain the Change Log Entries. Default value is 0. This parameter needs to be set before calling the SQL Procedure.

If the following sample code is executed against the production database, it will clear the Change Log Entries in all companies, keeping the last 60 days of entries in the database.

```
Declare @DBName nvarchar(200)

Set @DBName = DB_NAME()

EXEC [Tool].[dbo].Tool_ClearChangeLogEntry @TargetDatabase =
@DBName,@RetentionDays = 60
```

### 3.4.9 Tool_ClearJobQueueLogEntry

This stored procedure is licensed under the maintenance module. Run the following SQL code against the "Tool" database to remove old Job Queue Log Entries in Dynamics NAV/BC database in all companies.

Pay attention to @TargetDatabase parameter as this will determine the target database to clear the Job Queue Log entries. The parameters are explained as below.

```
EXEC [dbo].[Tool_ClearJobQueueLogEntry]
   @TargetDatabase
   ,@RetentionDays
```

@TargetDatabase :- Set the name of the database to clear Job Queue log entries.

@RetentionDays :- Number of days to retain the Job Queue Log Entries. Default value is 0. This parameter neds to be set before calling the SQL Procedure.

If the following sample code is executed against the production database, it will clear the Job Queue Log Entries in all companies, keeping the last 60 days of entries in the database.

```
Declare @DBName nvarchar(200)

Set @DBName = DB_NAME()

EXEC [Tool].[dbo].Tool_ClearJobQueueLogEntry @TargetDatabase =
@DBName,@RetentionDays = 60
```

## 3.5 Performance Logging SQL Stored procedure Definition

### 3.5.1 Tool_UpdateNavInfo

This stored procedure is licensed under the performance logging module. Run the following SQL code against the "Tool" database to collect and update Dynamics NAV/BC user and CAL and AL code involved to performance logs. Normally this SQL Stored procedure is called in SQL Agent Job called "Tool : Update Logs With Nav Info". The parameters are explained as below.

```
EXEC [dbo].[Tool_UpdateNavInfo]
   @DebugUserActivity
  ,@DebugApplicationCode
```

@DebugUserActivity :- Set this parameter to 1 to debug Dynamics NAV/BC user activity. Default value is 0. Setting this parameter to 1 will for a long time will lead to significant database size increase and hence setting the value to 1 is not normally recommended.

@DebugApplicationCode :- Set this parameter to 1 to debug Dynamics NAV/BC Application Code. Default value is 0. Setting this parameter to 1 will lead to a significant database size increase for a long time and hence setting the value to 1 is not normally recommended.

### 3.5.2 Tool_LogSlowQuery

This stored procedure is licensed under the performance logging module. Run the following SQL code against the "Tool" database to collect the top 100 slow query details into the Tool_SlowQueryLog table. Normally this SQL Stored procedure is called in SQL Agent Job called "Tool : Log Slow Query". The parameters are explained as below.

```
EXEC [dbo].[Tool_LogSlowQuery]
   @LogMode
```

@LogMode :- Set 0 to order the slow query by logical reads, 1 to order the slow query by execution count, 2 to clear all procedure cache, 4 to capture execution plan. For example, to capture execution plan and to order queries by execution count set this parameter to 4+1 = 5. Default value of this parameter is zero.

### 3.5.3 Tool_DiskResponseHistory

This stored procedure is licensed under the performance logging module. Run the following SQL code against the "Tool" database to collect disk response details into Tool_DiskResponseLog table. Normally this SQL Stored procedure is called in SQL Agent Job called "Tool : Log Disk Response".

```
EXEC [dbo].[Tool_DiskResponseHistory]
```

### 3.5.4 Tool_LogIndexUsageHistory

This stored procedure is licensed under the performance logging module. Run the following SQL code against the "Tool" database to collect disk response details into Tool_IndexUsageLog table. Normally this SQL Stored procedure is called in SQL Agent Job called "Tool : Log Index Usage". The parameters are explained as below.

```
EXEC [dbo].[Tool_LogIndexUsageHistory]
   @TargetDatabase
```

@TargetDatabase :- Set the name of the database to capture the index usage detail.

### 3.5.5 Tool_CPUUtilisationHistory

This stored procedure is licensed under the performance logging module. Run the following SQL code against the "Tool" database to collect CPU usage details into the Tool_CPUUtilizationLog table. Normally this SQL Stored procedure is called in SQL Agent Job called "Tool : Log CPU Utilisation".

```
EXEC [dbo].[Tool_CPUUtilisationHistory]
```

## 3.6 Analysis SQL Stored procedure Definition

### 3.6.1 Analyse_SQL_Server_Setup

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse SQL Server Setup and to validate it against best practice guideline. The parameters are explained below.

```
EXEC [dbo].[Analyse_SQL_Server_Setup]
   @TargetDatabase
```

@TargetDatabase :- Set the name of the database to the main database in your SQL server such as Dynamics NAV/BC/AX database.

When executed the Storedprocedure will return multiple record sets and they are as follows:

- General detail.
- Security Audit (list of users who have sysadmin rights).
- Cluster details (if applicable)

### 3.6.2 Analyse_Database_Setup

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse Database Setup and to validate it against best practice guideline. The parameters are explained below.

```
EXEC [dbo].[Analyse_Database_Setup]
   @TargetDatabase
```

@TargetDatabase :- Set the name of the database to analyse.

When executed the Storedprocedure will return multiple record sets and they are as followed:

- General detail.
- Security Audit (list of users who have direct access to the database).

### 3.6.3 Analyse_SQL_Server_Memory

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse SQL Server's memory usage and memory related key setups and to validate it against best practice guideline. Memory and its configuration are the most important hardware related factor which could influence system's performance.

```
EXEC [dbo].[Analyse_SQL_Server_Memory]
```

When executed the Storedprocedure will return multiple record sets and they are as followed:

- Overall memory usage detail.
- Detail memory usage information.
- Memory usage by database.
- Memory buffer analysis.

### 3.6.4 Analyse_CPU_Utilisation

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse SQL Server's CPU usage and to validate it against best practice guideline.

```
EXEC [dbo].[Analyse_CPU_Utilisation]
```

When executed the Stordedprocedure will return multiple record sets and they are as followed:

- Recent CPU usage details
- Signal and Resource wait details
- CPU usage by database.

### 3.6.5 Analyse_SQL_Waits

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse SQL Server's waits. This will give a very broad view of what type of task and resource SQL server is waiting the most. When there is a performance issue, quite often this analysis will give a bird's eye view of the area to concentrate.

```
EXEC [dbo].[Analyse_SQL_Waits]
```

When executed the Storedprocedure will return multiple record sets and they are as followed:

- Signal wait details
- Wait summary by Wait type.

### 3.6.6 Analyse_SQL_Waits

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse SQL Server's waits. This will give a very broad view of what type of task and resource SQL server is waiting the most. When there is a performance issue, quite often this analysis will give a bird's eye view of the area to concentrate.

```
EXEC [dbo].[Analyse_SQL_Waits]
```

When executed the Storedprocedure will return multiple record sets and they are as followed:

- Signal wait details
- Wait summary by Wait type.

### 3.6.7 Analyse_Maintenance_Job

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse maintenance jobs setup in the SQL Server. Having the correct maintenance jobs, which is fit for purpose is vital for the system's best performance.

```
EXEC [dbo].[Analyse_Maintenance_Job]
```

When executed the Storedprocedure will return multiple record sets and they are as followed:

- List of SQL Agent Jobs and when they are scheduled to run.
- SQL Agent Job execution statistics.
- Maintenance job error details.

### 3.6.8 Analyse_DynamicsNAV_Setup

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse key performance related configuration in a Dynamics NAV/BC database and to validate it against the best practice guideline. The parameters are explained below.

```
EXEC [dbo].[Analyse_DynamicsNAV_Setup]
   @TargetDatabase
```

@TargetDatabase :- Set the name of the database to analyse.

When executed the Storedprocedure will return multiple record sets and they are as followed:

- Dynamics NAV Application Settings.
- Service Tier details (if applicable)
- Costing related setups.
- Analysis view details.
- Workflow Settings.
- Change log details.
- Security audit (details of users who has SUPER access).

### 3.6.9 Analyse_SchedulerJob

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse various scheduler job details and logs in Dynamics NAV/BC database. The parameters are explained below.

```
EXEC [dbo].[Analyse_SchedulerJob]
   @TargetDatabase
  ,@Debug
```

@TargetDatabase :- Set the name of the database to analyse.

@Debug :- Set this parameter to 1 to get the SQL Code to analyse the Scheduler details. This is handy when this SQL stored procedure errors with missing field names and we can manually correct it and run it to get the details. Default value is 0.

When executed, the Stored procedure will return multiple record sets and they are as followed:

- LS Retail Replication logs such as PreAction, Actions
- LS Retail Scheduler.
- Job Queue.
- BAS Scheduler.
- Connect IT Scheduler.

### 3.6.10 Analyse_File_Layout

This stored procedure is licensed under system analysis module. Run the following SQL code against the "Tool" database to analyse file layout details of the database. Disk and its configuration along with how the database files are laid out are the second most important hardware related factor which could influence system's performance. The parameters are explained below.

```
EXEC [dbo].[Analyse_File_Layout]
   @TargetDatabase
```

@TargetDatabase :- Set the name of the database to analyse.

### 3.6.11 Analyse_Disk_Latency

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse how fast the disk system is responding to database's I/O calls. For a busy OLTP database such as Dynamics NAV/BC/AX disk, latency needs to be less than 20ms. Less than 10ms would be ideal. More than 50ms means we have serious I/O bottle neck. Disk and its configuration along with how the database files are laid out are the second most important hardware related factor which could influence system's performance. The parameters are explained below.

```
EXEC [dbo].[Analyse_Disk_Latency]
   @TargetDatabase
```

@TargetDatabase :- Set the name of the database to analyse.

When executed the Stored procedure will return multiple record sets and they are as followed:

- Disk Latency by File.
- Disk Latency by Database.

### 3.6.12 Analyse_Database_Growth

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse historic database growth details. From this detail one could estimate the database growth trend and hence use this data for capacity planning. The parameters are explained below.

```
EXEC [dbo].[Analyse_Database_Growth]
   @TargetDatabase
```

@TargetDatabase :- Set the name of the database to analyse.

When executed the Stored procedure will return multiple record sets and they are as followed:

- Full Database Backup History.
- Database Growth Rate.
- Transaction Log Backup History.
- Transaction Log Size Analysis by Hour.

### 3.6.13 Analyse_DynamicsNAV_Table_Size

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse the table size, record count for a Dynamics NAV/BC database. Stored procedure will give additional details such as the size of SIFT, extension name (if applicable), parent table etc.. The parameters are explained below.

```
EXEC [dbo].[Analyse_DynamicsNAV_Table_Size]
   @TargetDatabase
```

@TargetDatabase :- Set the name of the database to analyse.

When executed the Stored procedure will return multiple record sets and they are as followed:

- Database size by company.
- Database size by extension (if applicable).
- Database size, record count and other details.

### 3.6.14 Analyse_Table_Size

This stored procedure is licensed under system analysis module. Run the following SQL code against the "Tool" database to analyse the table size, record count for a non Dynamics NAV/BC database. The parameters are explained below.

```
EXEC [dbo].[Analyse_Table_Size]
   @TargetDatabase
```

@TargetDatabase :- Set the name of the database to analyse.

### 3.6.15 Analyse_Index_Usage

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse the index and SIFT usage. By analysing and removing unused index, database size can be significantly reduced. For a typical Dynamics NAV/BC database size can be reduced by around 25% to 15%. The parameters are explained below.

```
EXEC [dbo].[Analyse_Index_Usage]
   @TargetDatabase
```

@TargetDatabase :- Set the name of the database to analyse.

When executed the Stored procedure will return multiple record sets and they are as followed:

- SQL Server last restart detail
- Index usage analysis.

Note :- Index usage detail is only accurate since the last server restart. It is advised to ensure the last server restart is at least a month ago to rely on the output of this SQL Stored procedure to remove unused index.

### 3.6.16 Analyse_Slow_Query

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse slow query details from SQL Server's cache. If any process is running slow, the first line of troubleshooting could be done by running this stored procedure and to see the detail of the slow query. The parameters are explained below.

```
EXEC [dbo].[Analyse_Slow_Query]
   @ShowExecutionPlan
```

@ShowExecutionPlan :- Set this parameter to 1 to show execution plan. Default value is 0.

### 3.6.17 Analyse_Live_Blocking

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to transient blocking details from SQL Server's cache. If any process is running slow, the first line of troubleshooting could be done by running this stored procedure to see if there is any blocking taking place at the very moment and to see who is blocking who and who is leading the block chain. The parameters are explained below.

```
EXEC [dbo].[Analyse_Live_Blocking]
   @TargetDatabase
```

@TargetDatabase :- Set this parameter to the database in question to get extra information on blocked resource (index & table).

### 3.6.18 Analyse_Blocking

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse historic blocking events by analysing data in Tool_BlockingLog table. If any process is running slow, the first line of troubleshooting could be done by running this stored procedure and to see if the process is affected by blocking. The parameters are explained below.

```
EXEC [dbo].[Analyse_Blocking]
   @StartDate
  ,@EndDate
  ,@ServerName
```

@StartDate :- Set this optional parameter to analyse blocking on or after a specific date and time. When omitted, no filter will be applied on the start date. Specify values in YYYY-MM-DD HH:MM:SS format. For example, `'2020-07-23 13:15:54'`. (Hours, minutes and seconds are optional).

@EndDate :- Set this optional parameter to analyse blocking on or before a specific date and time. When omitted, no filter will be applied on the end date. Specify values in YYYY-MM-DD HH:MM:SS format. For example, `'2022-07-23 03:55:14'`. (Hours, minutes and seconds are optional).

@ServerName :- Set this optional parameter to filter on the SQL Server name. When omitted, no filter will be applied on the server name.

When executed the stored procedure will return multiple record sets and they are as followed:

- Leading SQL Blocking Query detail.
- Leading SQL Blocked Query detail.
- Leading Blocking Application Code (C/AL code for Dynamics NAV and AL code for Business Central).
- Leading Blocked Application Code (C/AL code for Dynamics NAV and AL code for Business Central).
- Leading Blocking User.
- Leading Blocked Resource.
- Leading Blocked Resource (for all sessions).
- Leading Blocked User.
- Blocking Statistic by Date.
- Blocking Statistic by Hour.
- Blocking Log.

### 3.6.19 Analyse_Deadlocks

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse historic deadlock events by analysing data in Tool_DeadlockLog table. The parameters are explained below.

```
EXEC [dbo].[Analyse_Deadlocks]
   @StartDate
  ,@EndDate
  ,@ServerName
```

@StartDate :- Set this optional parameter to analyse deadlocks on or after a specific date and time. When omitted, no filter will be applied on the start date. Specify values in YYYY-MM-DD HH:MM:SS format. For example, '2020-07-23 13:15:54'. (Hours, minutes and seconds are optional).

@EndDate :- Set this optional parameter to analyse deadlocks on or before a specific date and time. When omitted, no filter will be applied on the end date. Specify values in YYYY-MM-DD HH:MM:SS format. For example, '2022-07-23 03:55:14'. (Hours, minutes and seconds are optional).

@ServerName :- Set this optional parameter to filter on the SQL Server name. When omitted, no filter will be applied on the server name.

When executed the stored procedure will return multiple record sets and they are as followed:

- Deadlock SQL Query Pair.
- Deadlock Application Code Pair (C/AL code for Dynamics NAV and AL code for Business Central).
- Deadlock Resource Pair.
- Deadlock Statistic by Date.
- Deadlock Statistic by Hour.
- Deadlocked User.
- Deadlocking User.
- Deadlocked Resource.
- Deadlocking Resource.
- Deadlock Log.

### 3.6.20 Analyse_Historic_Slow_Query

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse historic slow query by analysing data in Tool_SlowQueryLog table. The parameters are explained below.

```
EXEC [dbo].[Analyse_Historic_Slow_Query]
   @ShowExecitionPlan
  ,@StartDate
  ,@EndDate
  ,@ServerName
```

@ShowExecitionPlan :- Set this optional parameter to get the execution plan. Set this parameter to 1 to get the execution plan. Default value of this parameter is 0.

@StartDate :- Set this optional parameter to analyse slow query details on or after a specific date and time. When omitted, no filter will be applied on the start date. Specify values in YYYY-MM-DD HH:MM:SS format. For example, '2020-07-23 13:15:54'. (Hours, minutes and seconds are optional).

@EndDate :- Set this optional parameter to analyse slow query details on or before a specific date and time. When omitted. no filter will be applied on the end date. Specify values in YYYY-MM-DD HH:MM:SS format. For example, '2022-07-23 03:55:14'. (Hours, minutes and seconds are optional).

@ServerName :- Set this optional parameter to filter on the SQL Server name. When omitted, no filter will be applied on the server name.

When executed the stored procedure will return multiple record sets and they are as followed:

- Historic Slow Query Statistics.
- Historic Slow Query Statistics by Day.
- Historic Slow Query Statistics by Batch.
- Historic Slow Query Statistics by Hour.

### 3.6.21 Analyse_Historic_Disk_Latency

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse historic disk latency by analysing data in Tool_DiskResponseLog table. The parameters are explained below.

```
EXEC [dbo].[Analyse_Historic_Disk_Latency]
   @StartDate
  ,@EndDate
  ,@ServerName
  ,@TargetDatabase
```

@StartDate :- Set this optional parameter to analyse disk latency on or after a specific date and time. When omitted, no filter will be applied on the start date. Specify values in YYYY-MM-DD HH:MM:SS format. For example, `'2020-07-23 13:15:54'`. (Hours, minutes and seconds are optional).

@EndDate :- Set this optional parameter to analyse disk latency on or before a specific date and time. When omitted, no filter will be applied on the end date. Specify values in YYYY-MM-DD HH:MM:SS format. For example, `'2022-07-23 03:55:14'`. (Hours, minutes and seconds are optional).

@ServerName :- Set this optional parameter to filter on the SQL Server name. When omitted, no filter will be applied on the server name.

@TargetDatabase :- Set this optional parameter to the name of the database to analyse. If omitted, data for all the database in the SQL server will be listed. You could select multiple databases by specifying the database name as a comma separated list.

When executed the stored procedure will return multiple record sets and they are as followed:

- Historic Disk Latency by Database,File Name.
- Historic Disk Latency by Database.

### 3.6.22 Analyse_Historic_CPU_Utilisation

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse historic CPU utilisation based on data collected in Tool_CPUUtilizationLog table. The parameters are explained below.

```
EXEC [dbo].[Analyse_Historic_CPU_Utilisation]
   @StartDate
  ,@EndDate
  ,@ServerName
```

@StartDate :- Set this optional parameter to analyse CPU utilisation on or after a specific date and time. When omitted, no filter will be applied on the start date. Specify values in YYYY-MM-DD HH:MM:SS format. For example, `'2020-07-23 13:15:54'`. (Hours, minutes and seconds are optional).

@EndDate :- Set this optional parameter to analyse CPU utilisation on or before a specific date and time. When omitted, no filter will be applied on the end date. Specify values in YYYY-MM-DD HH:MM:SS format. For example, `'2022-07-23 03:55:14'`. (Hours, minutes and seconds are optional).

@ServerName :- Set this optional parameter to filter on the SQL Server name. When omitted, no filter will be applied on the server name.

### 3.6.23 Analyse_SlowApplicationCode

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse slow Dynamics NAV/Business Central application code based on data collected through sql traces. The parameters are explained below. Please send an email to sales@sqlmantratools.com for the sql trace template with your SQL Server version.

```
EXEC [dbo].[Analyse_SlowApplicationCode]
   @SlowSQLTraceFileName
  ,@SessionActivityTraceFileName
  ,@UserNameFilter
```

@SlowSQLTraceFileName :- Set this parameter to the Slow SQL Statement trace file path and name.

@SessionActivityTraceFileName :- Set this parameter to the application code stack trace file path and name. This is an optional parameter.

@UserNameFilter :- Set this optional parameter to filter on the Dynamics NAV/Business Central user. If omitted, slow application code for all users will be analysed.

When executed the stored procedure will return multiple record sets and they are as followed:

- Slow C/AL or AL Code Summary.
- Slow SQL Code Summary.
- Slow SQL and C/AL or AL Code Detail.

### 3.6.24 Analyse_IndexFragmentation

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse index fragmentation. The parameters are explained below.

```
EXEC [dbo].[Analyse_IndexFragmentation]
   @TargetDatabase
  ,@PageCountFilter
  ,@FragmentationPercentageFilter
```

@TargetDatabase :- Set this parameter to the database in question to analyse index fragmentation.

@PageCountFilter :- Set this optional parameter to set the minimum number of pages per table to include in the analysis. Default value for this parameter is 25.

@FragmentationPercentageFilter :- Set this optional parameter to set the minimum for index fragmentation percent to include in the analysis. Default value for this parameter is 5 (ie 5%).

### 3.6.25 Analyse_Slow_SQLProcedure

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse slow SQL Stored procedures. The parameters are explained below.

```
EXEC [dbo].[Analyse_Slow_SQLProcedure]
   @ShowExecutionPlan
```

@ShowExecutionPlan :- Set this optional parameter to 1 to display the execution plan for the SQL Stored procedure. Default value is 0 (ie will not show execution plan).

### 3.6.26 Analyse_Statistic_Usage

This stored procedure is licensed under the system analysis module. Run the following SQL code against the "Tool" database to analyse statistic details. The parameters are explained below.

```
EXEC [dbo].[ Analyse_Statistic_Usage]
   @TargetDatabase
```

@TargetDatabase :- Set this parameter to the database in question to analyse statistic details.

## 3.7 Scheduler SQL Stored procedure Definition

### 3.7.1 Tool_AddScheduledJob

This stored procedure is licensed under the scheduler module. Run the following SQL code against the "Tool" database to create a new scheduler job in SQL Express Edition. The parameters are explained below.

```
EXEC [dbo].[Tool_AddScheduledJob]
   @JobName
  ,@Frequency
  ,@FrequencyUnit
  ,@DailyFrequency
  ,@DailyFrequencyUnit
  ,@NextRunTime
  ,@StartTime
  ,@EndTime
  ,@EndingDate
  ,@SQLToRun
  ,@DatabaseName
  ,@NotifyErrorsTo
```

@JobName :- Name of the job to create.

@Frequency :- Set this parameter to 0 for Once, 1 for Daily, 2 for Weekly.

@FrequencyUnit :- When @Frequency parameter is set to Weekly define the week day. Set 1 for Monday, 2 for Tuesday, 4 for Wednesday, 8 for Thursday, 16 for Friday, 32 for Saturday, 64 for Sunday. You can combine the weekdays by adding its number. For example, job to run on a Saturday, Sunday and Wednesday set the parameter to 32+64+4 = 100. For all other @Frequency parameter set this parameter to 0.

@DailyFrequency :- For @Frequency parameter set to Once or Daily, set this parameter to 0 for Once, 1 for Minute, 2 for Hour.

@DailyFrequencyUnit :- For @DailyFrequency set to Minute or Hour set the number of minutes or number of hours.

@NextRunTime :- Define the date and time to run the job next.

@StartTime :- Optionally define the start time each day to run the job. If omitted it will run continually.

@EndTime :- Optionally define the end time each day to run the job. If omitted it will run continually.

@EndingDate :- Optionally define the end date for the job and beyond this date the job will not run.

@SQLToRun :- SQL Code to run. This can be any TSQL code or a call to SQL Stored procedure.

@DatabaseName :- Name of the database to run the SQL.

@NotifyErrorsTo :- Optionally, specify the Email address to notify when the job fails. Multiple emails can be combined with ';'. For this to work you need to setup a database mail.

### 3.7.2 Tool_RunScheduledJobNow

This stored procedure is licensed under the scheduler module. Run the following SQL code against the "Tool" database to run a scheduler job in an ad hoc basis in SQL Express Edition. The parameters are explained below.

```
EXEC [dbo].[Tool_RunScheduledJobNow]
   @ScheduledJobId
```

@ScheduledJobId :- ID of the scheduler job to run

### 3.7.3 Tool_EnableScheduledJob

This stored procedure is licensed under the scheduler module. Run the following SQL code against the "Tool" database to enable a scheduler job in SQL Express Edition. The parameters are explained below.

```
EXEC [dbo].[Tool_EnableScheduledJob]
   @ScheduledJobId
  ,@NextRunTime
```

@ScheduledJobId :- ID of the scheduler job to activate

@NextRunTime :- Optionally set the next Run date and time. If omitted, system will calculate the next Run date and time based on the Scheduled job parameters.

### 3.7.4 Tool_RemoveScheduledJob

Run the following SQL code against the "Tool" database to remove scheduler job in SQL Express Edition. The parameters are explained below.

```
EXEC [dbo].[Tool_RemoveScheduledJob]
   @JobScheduleId
```

@JobScheduleId :- ID of the scheduler job to delete

## 3.8 How to Install SQL Mantra Tool

After downloading the SQL Mantra Tools from its website follow these steps to install the tool. Please ensure these steps are executed in the following order.

1) Restore the Tool database from the backup. You can locate the file in “Install\Tool Database” folder.

2) Install SQL Shield and select the correct SQL Server instance when prompted. You can locate the file in “Install\SQL Shield” folder.

3) Make sure the license for the SQL Mantra Tool is loaded onto the ‘Tool’ database. After purchasing you should be able to download the license from the SQL Mantra Tools web portal by logging into it. Please send an email to sales@sqlmantratools.com if you have any issues.

4) Run “A - Install.sql” script against the ‘Tool’ database. Make sure the script runs without any error. Follow Instructions from the output. You can locate the file in “Install\Scripts” folder.

5) When installing maintenance job and if the database is participating in log shipping or part of always availability cluster, further adjustment is needed to the reindexing job. Please refer to the Tool_Reindex section for setup details.

6) If SQL Server is running on a cluster, install the SQL Shield in each of the secondary cluster nodes and add the ‘Tool’ database to the cluster node.

7) For NAV2013 to BC14 load support changes to centrally control SQL Trace for NAV2013 and any versions after that. Ensure your NAV/BC license includes permission for Table 78000, Page 78000 and Codeunit 78000. If not renumber the objects to your licensed range. Sample CAL code can be found in “Install\SQL Tracing Object for Dynamics NAV & BC” folder.

8) For BC15 to BC22 load the extension provided. You can locate the extension app file under “Install\SQL Tracing Extensions” folder. See “How to Load SQL Mantra Tools extension” section for more details. Ensure your BC license includes permission for Table 78000, Page 78000 and Codeunit 78000.

9) Add read permission to table 78000 (or the equivalent renumbered table) to all NAV/BC users.

10) For NAV2013 to BC14, open Dynamics NAV Development client and run "SQL Trace Setup" page and tick "SQL Tracing On" to activate global tracing.

11) For BC15 to BC22, open BC client and type "SQL Trace Setup" on the search box and tick "SQL Tracing On" to activate global tracing.

## 3.9 How to Load SQL Mantra Tools extension

For Business Central V15 onwards load the extension provided in the install directory to centrally turn on and off SQL Tracing which the tool relies to capture the BC user details and the AL code involved in performance incidents such as Blocking and Deadlocks. Load the extension as normal.

For BC15 to BC22 load the extension provided in the "SQL Tracing Extensions" folder by running the following powershells from Business Central Administration Shell. See example below. Adjust the code as required. Ensure you install the correct version of app for the Business central version.

For BC15 to BC20 load "SQL Trace Management_15.0.0.0.app".
For BC21 to BC22 load "SQL Trace Management_21.0.0.0.app".

Publish-NAVApp -ServerInstance <BC Instance Name> -Path ".\SQL Trace Management_21.0.0.0.app" -SkipVerification

Sync-NavApp -ServerInstance <BC Instance Name> -Name "SQL Trace Management"

Install-NAVApp -ServerInstance <BC Instance Name> -Name "SQL Trace Management"

Do not forget to search "SQL Trace Setup" on the search box of Business Central Client and tick "SQL Tracing On" to activate global tracing.

In BC23 onwards you may get "Sorry, we can't proceed with the operation for app 'xxxxx' by 'yyyyy' for tenant 'nav-systemapplication' because the app doesn't meet our Universal Code requirement. You must either upgrade the app to meet the requirement or license the non-Universal Code module that grants an exception to the requirement" error while loading the extension. This is to the change in license requirement by Microsoft especially when it comes to apps targeting on-prem environment. To overcome this error, you may:

1) Need to add "Implemented code is not cloud-optimized" module to the BC license.
   Or
2) Subscribe to the on-prem version of Performance Insights Product of SQL Mantra Tools instead. Please note we provide free upgrade and product switch.

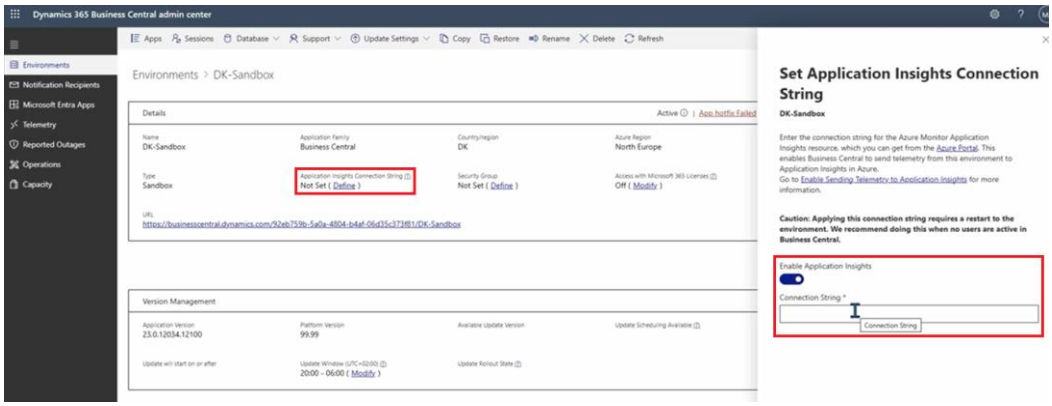
Please email sales@sqlmantratools.com for advice.

## 3.10 How Setup Performance Insights

SQL Mantra Tools' Performance Insights product works in a different way compared to its on-prem products. It listens to the raw telemetry signals emitted by the Microsoft SaaS environment and will analyse, process the raw data received and will present it in interactive graphs and tables in a user-friendly format, highlighting the issues with priority via our web portal. Please follow the steps below to get this setup.

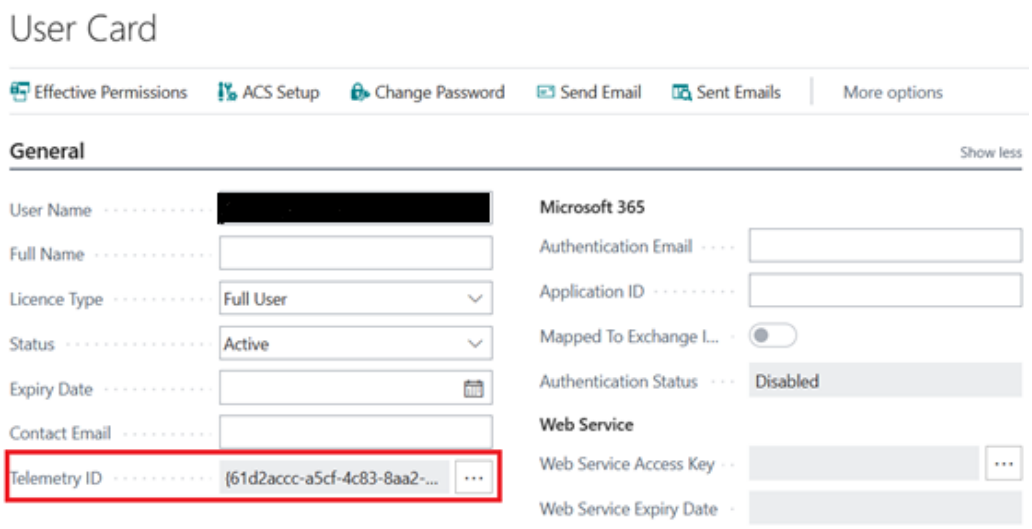
1) Send an email to sales@sqlmantratools.com expressing your desire to use Performance Insights product along with your company details, contact details clearly indicating whether the Business Central System is hosted on SaaS or On-prem.

2) Once approved, for SaaS implementation, specify the connection string provided in the customers' "Business Central Admin Center" and click "Enable Application Insights and Click the "Save" button. See below for example. This will require a service tier reboot and hence it is advised to perform this setting when users are not using the system, i.e. during the Maintenance Window.

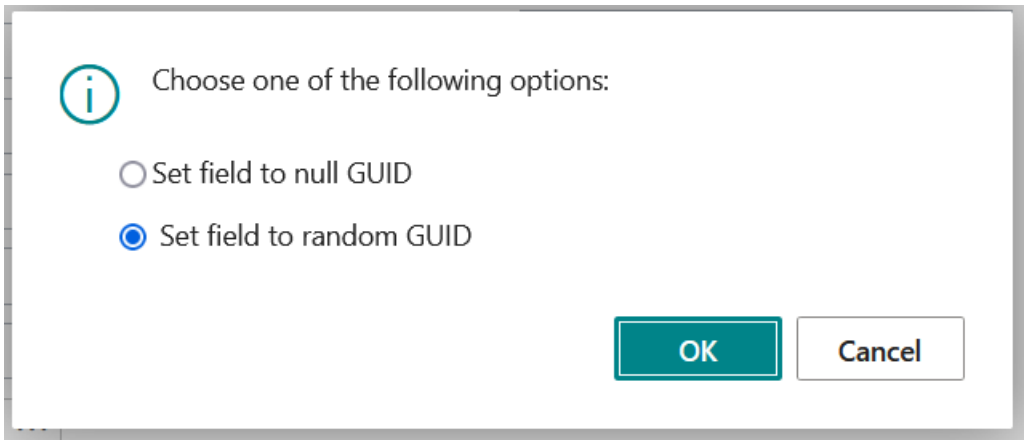



3) Next log into Business Central and check if the "Telemetry ID" has been setup on User Card for **all users**, if not, set it following the steps below to set "Telemetry ID", so that the user identity can be reported in the Performance Insights.

4) If not set, click the build button for "Telemetry ID" and assigning a GUID for the Telemetry ID field. See example below.




5) Select “Set field to random GUID” option and click the “OK” button.

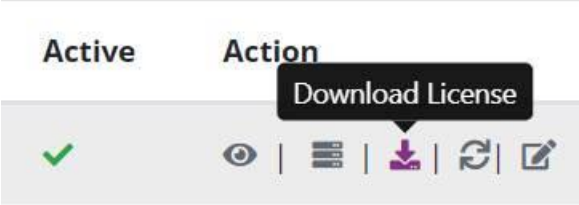



6) Once completed the configuration, send an email to sales@sqlmantratools.com confirming the setup details along with the Time Zone of the Business Central System for SQL Mantra Tools to complete the setup on the portal so that SQL Mantra Tools website will complete the handshake and will start to report the data in the customers’ own time zone and will display the customer name in the portal as well.

7) Once the handshake is complete at SQL Mantra Tools’ end, users can log onto the SQL Mantra Tools’ web portal to analyse and see the performance data such as Slow Query Details, Blocking Details as well as Deadlock Details.

## 3.11 How to Load SQL Mantra Tool License

The License file will have an extension of ".sql". If the license file is not provided, one can download it from the SQL Mantra Tools web portal. Once logged in click "Manage Customers" from the "Dashboard" option, click "Manage Customer License" and then click the "Download License" icon, see below for example. Any issue, contact us through the web portal or send an email to sales@sqlmantratools.com.




Open the license file using SQL Management Studio and execute the content, making sure you have selected the 'Tool' database in the top ribbon. Make sure it executes without any error. Once the license is loaded, it is advisable to run the following SQL Procedure against the 'Tool' database to check the license, paying attention to the server.

```
EXEC [dbo].[Tool_GetLicense]
```

## 3.12 How to Upgrade SQL Mantra Tool

After downloading the SQL Mantra Tools from its website follow these steps to upgrade it to the latest version. Please ensure these steps are executed in the following order.

1) Restore the new Tool database as “Tool1” database.

2) Load the SQL Mantra license onto the “Tool1” database. After purchasing you should be able to download the license from the SQL Mantra Tools web portal by logging into it. Please send an email to sales@sqlmantratools.com if you have any issues.

3) Run the upgrade script “Upgrade Script.sql” to copy data from the “Tool” database to the “Tool1” database. Make sure the code is executed against the “Tool1” database. The script file can be found under “Utlity\Scripts” folder.

4) Delete the “Tool” database.

5) Rename “Tool1” database to “Tool” database.

6) Run “A - Install.sql” Script against the “Tool” database to activate - ignore all the steps and messages as maintenance jobs and others should have been done already. The install script can be found under “Install\Scripts” folder.

## 3.13 How to Uninstall SQL Mantra Tool

In the unlikely event of if you would like to uninstall the SQL Mantra Tool, follow these steps to uninstall the Tool.

1) Run “Z - UnInstall.sql” script to deactivate the tool. The script can be found under “Utlity\Scripts” folder.

2) Delete SQL Agent Jobs with the prefix of “Maintenance : <Database Name>”.

3) Delete SQL Agent Jobs with the prefix of “Tool :”.

4) Delete Extended events with the prefix of “Tool :”.

5) Delete the table and SQLStored procedure with a prefix of “Tool _” from msdb database.

6) Uninstall SQL Shield from control panel.

7) Delete the “Tool” database.

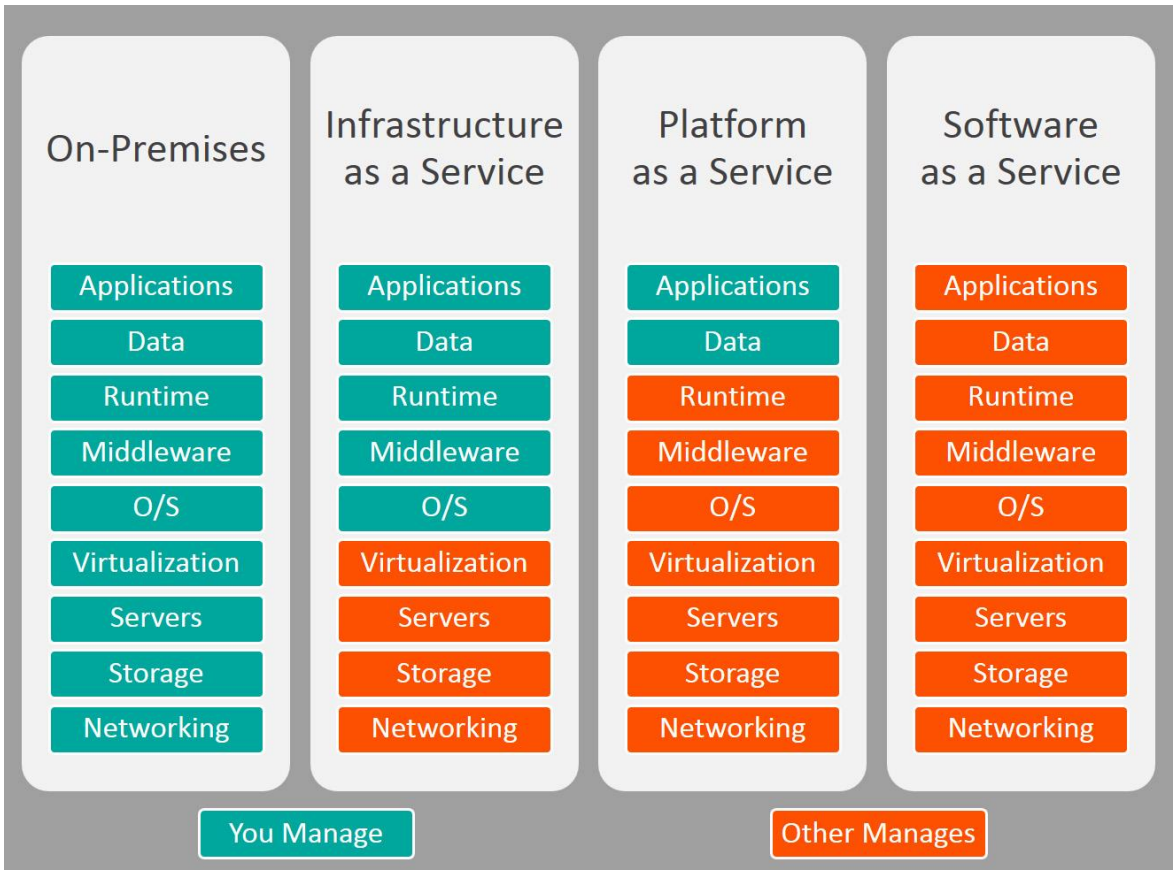
8) If applicable, run Dynamics NAV/BC client and untick run "SQL Trace Setup" page in Dynamics NAV/BC and tick "SQL Tracing On" to remove global tracing. Remove the helper objects / uninstall extension.

9) If SQL Server is running on a cluster, repeat above steps in each of the secondary cluster node.

SQL Mantra Tool believes in clean install and uninstall process.

# 4 Environment options for Business Central and ERP systems

The following diagram gives details about the options available with environmental choice for an ERP implementation such as Dynamics NAV, AX or Business Central. This diagram will also summarise what the customer will manage and what the hosting provider will manage in each of the model, giving a clear picture to the customer before they commit to an environment. There are many factors to consider before choosing the environment as there are many pros and cons in each of these options. There are many confusing, biased and conflicting opinions in the market place. Your organisation will need unbiased advice when it comes to choosing the best environment which best fits your business need, strategic direction, cost, etc… For unbiased advice contact sales@sqlmantratools.com for Infrastructure Planning & Sizing service to focus on your organisation's need.



# 5 Useful Performance Troubleshooting Scripts

Please find some important performance troubleshooting and data gathering SQL Scripts for performance tuning exercise below.

## SQL Server Setup Analysis

Run the following code against the database under investigation.

```
Declare @DBName Varchar(max)
Set @DBName = DB_NAME()
EXEC [Tool]..[Analyse_SQL_Server_Setup] @DBName
```

## Tempdb Setup Analysis

```
EXEC [Tool]..[Analyse_Database_Setup] 'tempdb'
```

## Server Memory Analysis

```
EXEC [Tool]..[Analyse_SQL_Server_Memory]
```

## CPU Utilisation Analysis

```
EXEC [Tool]..[Analyse_CPU_Utilisation]
```

For historic CPU Utilisation detail, run the following code. Optionally set @StartDate,@EndDate parameters could be used to filter the output. Dates are in YYYY-MM-DD HH:MM:SS format e.g. (2017-12-30 20:54:16)

```
EXEC [Tool]..[Analyse_Historic_CPU_Utilisation]
```

## SQL Server Wait Analysis

```
EXEC [Tool]..[Analyse_SQL_Waits]
```

## Maintenance Job Analysis

```
EXEC [Tool]..[Analyse_Maintenance_Job]
```

## Database Setup Analysis

Run the following code against the database under investigation.

```
Declare @DBName Varchar(max)
Set @DBName = DB_NAME()
EXEC [Tool]..[Analyse_Database_Setup] @DBName
```

## Dynamics NAV Setup Analysis

Run the following code against the database under investigation.

```
Declare @DBName Varchar(max)
Set @DBName = DB_NAME()
EXEC [Tool]..[Analyse_DynamicsNAV_Setup] @DBName
```

## Scheduler Job Analysis

Run the following code against the database under investigation.

```
Declare @DBName Varchar(max)
Set @DBName = DB_NAME()
EXEC [Tool]..[Analyse_SchedulerJob] @DBName
```

## Database File Layout Analysis

Run the following code against the database under investigation.

```
Declare @DBName Varchar(max)
Set @DBName = DB_NAME()
EXEC [Tool]..[Analyse_File_Layout] @DBName
EXEC [Tool]..[Analyse_File_Layout] 'tempdb'
```

## Disk Latency Analysis

Run the following code against the database under investigation.

```
Declare @DBName Varchar(max)
Set @DBName = DB_NAME()
EXEC [Tool]..[Analyse_Disk_Latency] @DBName
EXEC [Tool]..[Analyse_Disk_Latency] 'tempdb'
```

For historic disk latency details, run the following code.

```
EXEC [Tool]..[Analyse_Historic_Disk_Latency]
```

## Database Size History Analysis

Run the following code against the database under investigation.

```
Declare @DBName Varchar(max)
Set @DBName = DB_NAME()
EXEC [Tool]..[Analyse_Database_Growth] @DBName
```

## Table Size Analysis

Run the following code against the database under investigation.

```
Declare @DBName Varchar(max)
Set @DBName = DB_NAME()
EXEC [Tool]..[Analyse_DynamicsNAV_Table_Size] @DBName
```

## Index Usage Analysis

Run the following code against the database under investigation.

```
Declare @DBName Varchar(max)
Set @DBName = DB_NAME()
EXEC [Tool]..[Analyse_Index_Usage] @DBName
```

For historic index usage detail, the following code against the database under investigation.

```
Declare @DBName Varchar(max)
Set @DBName = DB_NAME()
EXEC [Tool]..[Analyse_Historic_Index_Usage] @DBName
```

## Index Fragmentation Analysis

Run the following code against the database under investigation.

```
Declare @DBName Varchar(max)
Set @DBName = DB_NAME()
EXEC [Tool]..[Analyse_IndexFragmentation] @DBName
```

## Slow Query Analysis

```
EXEC [Tool]..[Analyse_Slow_Query]
```

For historic records Run the following code. Optional @StartDate,@EndDate parameters could be used to filter the output. Dates are in YYYY-MM-DD HH:MM:SS format eg (2017-12-30 20:54:16)

```
EXEC [Tool]..[Analyse_Historic_Slow_Query]
```

For slow SQL Procedure details run the following code

```
EXEC [Tool]..[Analyse_Slow_SQLProcedure]
```

## Blocking Analysis

Run the following code to get the historic blocking data . Optional @StartDate,@EndDate parameters could be used to filter the output. Dates are in YYYY-MM-DD HH:MM:SS format eg (2017-12-30 20:54:16)

```
EXEC [Tool]..[Analyse_Blocking]
```

To Analyse Live blocking, run the following code against the database under investigation.

```
Declare @DBName Varchar(max)
Set @DBName = DB_NAME()
EXEC [Tool]..[Analyse_Live_Blocking] @DBName
```

## Deadlock Analysis

Run the following code to get the historic deadlock data. Optional @StartDate,@EndDate parameters could be used to filter the output. Dates are in YYYY-MM-DD HH:MM:SS format eg (2017-12-30 20:54:16)

```
EXEC [Tool]..[Analyse_Deadlocks]
```

# 6 Troubleshooting

a) If the SQL Mantra Tool is not working or not logging performance data, check SQL Server's Error log for further details. You may find some useful information to troubleshoot.

   It's also worth checking if the license is not expired and the SQL Mantra Tool is licensed to the correct SQL Server or cluster and if the Performance Logging Module is licensed by running the following SQL code.

```
EXEC [dbo].[Tool_GetLicense]
```

b) If Logging Module is licensed and SQL Mantra Tool is not logging performance data such as blocking, there could be some issue with the SQL Server's Block event notification or processing it. Run the following SQL code to reset.

```
EXEC [dbo].[Tool_Activate]
```

c) If the Application code involved in performance incident is blank, check if the supporting NAV helper objects are there in the database and "SQL Tracing On" field is ticked in "SQL Trace Setup" table. In Business Central, check if SQL Mantra Tools' extension is loaded and "SQL Tracing On" field is ticked in "SQL Trace Setup" table.

   Check "Tool_ApplicationCodeStack" Extended Event is active in SQL Server.

   Ensure "Tool : Update Logs With Nav Info" SQL Agent Job is running without any error.

   Note :- This feature is only supported in NAV2013 onwards and all the Business central versions.

d) If you get errors like "Incorrect syntax near '?'." You should install SQL Shield provided in the Tools.zip file and try again.

e) If an antivirus is active in SQL Server. Please ensure there is a file exclusion for **.xem** and **.xel** file extensions.

If you are unable to troubleshoot visit our website and contact us on https://www.sqlmantratools.com/contact-us with the error detail and our technical department will be glad to help you.

# 7 GLOSSARY

| Term | Meaning |
|---|---|
| AL | Language used to develop extensions for Business Central. |
| BC | Business Central 365. |
| BI | Business intelligent. |
| C/AL | Client/server Application Language. The language that is used to customise solution is Dynamics NAV Software. |
| CSIDE | Client/Server Integrated Development Environment. |
| ERP | Enterprise resource planning. |
| Fill factor | Fill factor is the value that determines the percentage of space on each leaf-level page to be filled with data. |
| IaaS | Infrastructure as a Service. |
| OLTP | Online Transaction Processing. |
| On-prem | Software is installed and runs on computers in your company's own IT environment. |
| PaaS | Platform as a Service. |
| SaaS | Software as a Service. |
| SIFT | Sum Index Field Technology. |