Kansas Department of Revenue HealthCheck Report

Prepared by Brian Carr
Burleson Consulting

Joint Committee on Information Technology
October 21, 2013
Attachment 4

Document Information

Change History

Release	Date	Description	By
1.0	09/09/13	Initial release	Brian Carr
2.0	09/10/13	Internal review	Troy Ross
3.0	09/14/13	Clarifications based on client review	Brian Carr

Contents

Project Overview

Executive Summary

Kansas Dept of Revenue has a SQL Server database running on a Windows 2008 server and was interested in acquiring professional consulting services to assist their in-house IT department to diagnose performance problem and recommend best practices.

For the purpose of this engagement, SQL Server error logs, Event Viewer error logs, system tables and various scripts were used to diagnose overall system performance. Findings are ordered in categorized groups based upon their bottleneck from highest impact to lowest. An estimated difficulty level is also included for each Finding.

Architecture

OS version: SQL Server 2008 Enterprise Edition 64-bit (Clustered)
RAM: 147,443MB
CPU: 24

Storage: Dell/EMC SAN CX3-20c (iSCSI)
VM Datastore: 1.8TB VMFS Volume (dedicated) (7) 450GB 10K RAID 5

Summary of Findings

The table below shows the Findings in order of impact from highest to lowest. Due to the nature of bottleneck tuning, the findings are categorized in order as well to specifically tackle each bottleneck on the system from highest to lowest. As bottlenecks are reduced or removed, SQL Server is free to make more efficient and heavy use of CPU in order to process rows instead of waiting on row retrievals, parsing and other wait events.

Database time in SQL Server is a metric which is composed of time spent waiting or working. The goal of tuning is to reduce the 'waiting' aspect of the database time and improve the 'working' aspect.

Over normalization was not readily apparent. Tables with heavy access in ATTM and MOVRS were checked to ensure they:

- have an identifier
- store only data for a single type of entity
- avoid nullable columns where possible
- do not have repeating values or columns

HEALTH CHECK REPORT

Finding	Area	Difficulty	Impact
1. Degree of parallelism is not set to recommended value	Configuration	Low	Medium to High
2. Memory contention	Configuration	Low	Medium
3. Very low blocked process threshold setting detected	Configuration	Low	Medium to Low
4. I/O requests taking 15+ seconds to complete	Disk I/O	Low	Medium
5. Data and Log files on the same volume	Disk I/O	Low	Low
6. Disk sector [0] alignment of server may not be configured optimally	Disk I/O	High	Medium
7. Nightly maintenance window runs long	Statistics & Integrity	Low	Low
8. Database files and backups exist on the same volume	Misc Findings	Low	Low
9. Authentication mode not set to best practice	Misc Findings	Low	Low
10. SQL Server login password policy strength and password expiry	Misc Findings	Low	Low
11. SQL Server Admin role membership check	Misc Findings	Low	Low
12. Auto close enabled	Misc Findings	Low	Low
13. Missing and Needed Indexes	SQL Tables/Stored Procedures/Object Tuning	Medium	Medium to High

Finding Details

Configuration

Finding 1: Degree of parallelism is not set to recommended value

Max degrees of parallelism (a.k.a MAXDOP) in SQL Server is used to control the maximum number of threads per execution plan operators work. MAXDOP does not restrict the number of CPU's/Threads used by single batch or Query. Currently MAXDOP is set to 1.

Ideally MAXDOP should be equal to number of online schedulers per node. You can use the below query to get the number of online schedulers per node. All the parallel threads for the tasks of the query will be assigned from schedulers of same node so having MAXDOP beyond the number of online schedulers per node may not really improve the performance (With some exceptions).

The query below shows the number of online schedulers per node which is 12:

```
select count(*) as Maxdopcount, parent_node_id from
sys.dm_os_schedulers where status='VISIBLE ONLINE' group by
parent_node_id
```

Recommendation

1.In SQL Server Studio Manager, right-click a server and select Properties.
2.Click Advanced from select a page.
3.In the Max Degree of Parallelism box, select the maximum number of processors to use in parallel plan execution..

Difficulty: Low

Impact: Medium to High

Finding 2: Memory contention

Significant portions of SQL Server Memory have been paged out on occasion.

Recommendation

In an effort to resolve the memory contention in the system it's recommended you do the following:

Step 1: create a user account (eg. SQLServiceAccount) that will be used to start the "SQL Server (MSSQLSERVER)" Windows Service

Step 2: turn on "lock pages in memory" via gpedit.msc > Computer Configuration > Windows Settings > Security Settings > Local Policies > User Rights Assignment > double-click Lock pages in memory then click add. In the "Select Users or Groups" add the SQLServiceAccount

Difficulty: Low

Impact: Medium

Finding 3: Very low blocked process threshold setting detected

The current value of 2 specified for the "blocked process threshold" option is very low. Using low values for this threshold can result in deadlock monitor running constantly and consuming resources. The minimum time it can detect a blocked process is 5 seconds anyway.

Recommendation

Set the blocked process threshold option to a value of 5 or higher

```
sp_configure 'show advanced options', 1 ;
GO
RECONFIGURE ;
GO
sp_configure 'blocked process threshold', 5 ;
GO
RECONFIGURE ;
GO
```

Difficulty: Low

Impact: Medium to Low

Disk I/O

SQL Server's primary function is I/O both from disk and RAM. Provided enough memory is given to SQL Server, recently used blocks from the data files will be pulled into RAM to service query requirements. This type of read is known as a logical I/O. However, when there is not enough space in the buffer cache for required blocks, SQL Server must read data from disk by performing a physical I/O.

When disk I/O is the top wait event on a system, the SQL Server environment is not free to adequately utilize CPU which is the fastest component of the system. An I/O bound environment

can be tuned by reducing or speeding up disk I/O to the point where required data is cached, and therefore accessed in RAM via CPU.

Finding 4: I/O requests taking 15+ seconds to complete

SQL Server has encountered 1 occurrence(s) of I/O requests taking longer than 15 seconds to complete on file. On two occasions (2/18/2013 and 3/21/2013) SQL Server reported Event ID 833 when I/O requests take longer than 15 seconds to finish. This indicates a potential problem with the disk I/O subsystem or configuration. This affects SQL Server performance.

Recommendation

- Schedule batch jobs to run during off-hours
- Ensure no full backups are being executed during the day
- Ensure statistics are not being updated during the day via maintenance tasks (auto update of stats is fine)
- Ensure no rebuild/reorganize of indexes are happening during the day
- Tune I/O intensive queries (as discussed in a separate finding)

Difficulty: Low

Impact: Medium

Finding 5: Data and Log files on the same volume

Databases [Admin, Archive, ATMM, Common, KDOR, MOVRS] present on this server have their data and log files in the same volumes.

Placing both data and log files on the same device can cause contention for that device and result in poor performance

Recommendation

During the limited periods of observation there was no indication of the G: drive disk queue being too high. However it is a best practice for data and log files to be placed on separate volumes.

Difficulty: Low

Impact: Low

Finding 6: Disk sector [0] alignment of server may not be configured optimally

On most storage systems, hard disk partitions are configured optimally when they have a starting offset of 65536 bytes, or a multiple of 65536 bytes. Hard disk partitions with starting offsets which differ from this recommendation were detected [Volume Identifier: Disk #11, Partition #1, Starting Offset: 17408]. Disk #11 is the G: drive where all the data, log and temp files are located.

Noncompliance with storage configuration best practices for SQL Server database software is a common root cause of performance issues. The reason is often shown to be misalignment between Windows, storage, disk controllers, and cache segment lines.

Failure to perform partition alignment may result in significant performance degradation. Disk partition alignment is a requirement for partitions from which high performance is demanded and that were created on RAID disk.

Aligned disks have been shown by Microsoft to increase disk I/O operations in excess of 30%.

To verify the offset issue, reported by SQL Server is correct check via wmic:

```
C:\>wmic partition get BlockSize, StartingOffset, Name, Index

BlockSize  Index  Name                    StartingOffset
512        0      Disk #11, Partition #0  17,408
```

Recommendation

When choosing a valid partition starting offset, refer first to your storage vendor best practices. Make certain their recommendations correlate with the stripe unit size and file allocation unit size configured for SQL Server.

Difficulty: High

Impact: Medium

Statistics & Integrity

Finding 7: Nightly maintenance window runs long

The nightly maintenance window runs for several hours and on occasion has to be canceled as its affects users in the morning.

Recommendation

The following two steps could be moved to the weekend rather than running each night. These two tasks consume 84 minutes and 225 minutes respectively:

- Check Database Integrity Task
- Update Statistics - Admin and Archive (If data in the Admin and Archive databases are relatively static then auto update stats should remain off. Otherwise auto update should be enabled.)

Note: Disabling automatic statistics recomputation can cause the SQL Server query optimizer to choose a less optimal strategy for queries that involve the specified table. It is a best practice to update statistics each night for databases with heavy DML (updates, inserts, deletes). Most SQL shops are able to only rebuild indexes on the weekend, but this is case by case. My understanding is you have had serious performance issues when not updating these nightly.

Difficulty: Low

Impact: Low

Misc Findings

Finding 8: Database files and backups exist on the same volume

SQL Server reports Database files and backups exist on the same volume.

Recommendation

Ensure the backup files are moved off to tape in a timely fashion.

Difficulty: Low

Impact: Low

Finding 9: Authentication mode not set to best practice

The authentication mode is not set to recommended value. Windows Authentication is the default authentication mode, and is much more secure than SQL Server Authentication. Windows Authentication uses Kerberos security protocol, provides password policy enforcement with regard to complexity validation for strong passwords, provides support for account lockout, and supports password expiration

Recommendation

Configure Authentication mode to Windows mode when possible

Difficulty: Low

Impact: Low

Finding 10: SQL Server login password policy strength and password expiry

Password complexity policies are designed to deter brute force attacks by increasing the number of possible passwords. Turning OFF the password policy could result in weak SQL Server login passwords for these noncompliant logins: kdorinterface , kdorreadonly , kdortaxunit , rvcnczf , rvcnhxs , rvcnmxt , rvcnrzm , RVDMCAP , RVISACW , rvisass , rvisdra , RVISGLH , RVISJZH , rvismmw , RVISNCS , RVISVAP , RVOPSAA , sa.

Recommendation

Enforce password expiration" for all the SQL Server logins

Difficulty: Low

Impact: Low

Finding 11: SQL Server Admin role membership check

The Sysadmin role membership should be granted to limited users. Check the membership and determine if users other than Local System, SQL Server built in admin (SA), SQL Server Engine and Agent service account or service SID are granted Sysadmin permission. Current number of logins with Sysadmin role is 14.

Recommendation

Grant Sysadmin role membership to users who require it. Carefully evaluate the membership of the Sysadmin role. Granting Sysadmin membership to users who do not require it represents potential security risk.

Difficulty: Low

Impact: Low

Finding 12: Auto close enabled

AUTO_CLOSE is a database option available in SQL Server. When this option is turned ON, a database will be shut down after all resources that reference this database are freed.
After a database is shutdown, next time an application attempts to use the database, the database has to be first opened and then get the status to online. This could take some time and can result in application time outs.

Recommendation

For each database, except for master, tempdb, msdb, and model

```
ALTER DATABASE 'DB_NAME' SET auto_close off
```

Difficulty: Low

Impact: Low

SQL Tables/Stored Procedures/Object Tuning

Finding 13: Missing and Needed Indexes

At the end of August Brian captured a four hour workload during the busy monthend timeframe. This workload was then run through the SQL tuning advisor. However due to the large size of the databases relative to the C: drive on the server running a full analysis via the advisor was not possible.

Recommendation

1. See Appendix A for a list of SQL statements which SQL Server states would be improved 97.23% to 99.57% by implementing the associated indexes and creating additional statistics listed in Appendix B. This is specific to the ATTM database.

2. Appendix C lists statistics which should improve the performance of the MOVRS, according to SQL Server tuning advisor. Unfortunately due to space constraints on drive C: of the server the full workload could not be analyzed by the tuning advisor tool.

3. **On-going basis:** Increase the size of the C: drive by 300% to allow for workloads to be fully analyzed by SQL tuning advisor. This is a process the DBA should take on a periodic basis (2-4 times per year). The best practice is to record the workload, like Brian did in August, then on a duplicate test server perform the analysis, implement the recommendations, and then test. If testing shows a positive improvement then implement the recommendations on the production server.

   Difficulty: Medium

Impact: Medium to High

Best Practices

TEMPDB considerations

The SQL Server is currently configured with 16 temp data files.

Recommendation

1. Monitor SQL Server Error Logs periodically for issues related to tempdb:
2. Leave Auto Create Statistics and Auto Update Statistics on tempdb.
3. Do not shrink the tempdb.
4. If you are seeing an error code 1101, 1105 then a session has to allocate more space in tempdb in order to continue. This will create performance degradation while the current operation waits for the space to be created.
5. Divide tempdb amongst multiple physical files. One for each physical core. While this system has 24 processors, 16 temp files are good. The reason to have multiple data files is to reduce possible allocation contention, as objects are created and destroyed in TempDB. So unless contention on tempdb is happening 16 temp files are good.
6. Use RAID 1 or 10 for tempdb operations.
7. Minimize Disk I/O and isolate from other disk operations if possible.
8. The TempDB files should definitely not be on the system (C) drive and ideally not on the G: drive. If TempDB grows out of control, perhaps due to a large sort query, the system can run out of hard drive space and fail to start.
9. Always create temp data files of the same size. If all of the TempDB data files are the same size, SQL Server will use all of them equally.

Process to identify high-impact SQL

1. On a regular basis archive old data from the tables involved.
2. Next run the following query to identify the next Top 5 I/O intensive queries and tune the SQL and archive data where possible to keep the objects (tables, indexes, statistics) trimmed for optimal performance.
3. Consider using Indexed Views to address performance issues caused by any table normalization.

```
select top 5 (total_logical_reads/execution_count) as
avg_logical_reads, (total_logical_writes/execution_count) as
avg_logical_writes,(total_physical_reads/execution_count) as
avg_physical_reads, Execution_count, statement_start_offset,
p.query_plan, q.text from sys.dm_exec_query_stats cross apply
```

```
sys.dm_exec_query_plan(plan_handle) p cross apply
sys.dm_exec_sql_text(plan_handle) as q order by
(total_physical_reads)/execution_count DESC,
(total_logical_reads + total_logical_writes)/execution_count
Desc
```

Deadlocks and timeouts

The process specified in step 3 of Finding 13 is the method to follow. This will go a long way. Once one or two full workload analysis has been completed and the recommendations implemented then any remaining timeouts and deadlocks will need analyzed. These remaining deadlocks will most likely require changes in application logic.

Monitoring & Diagnosing best practices (error logs, notifications, etc.)

Reviewing the SQL Server log and Windows event viewer each day for errors and warning (filter out informational messages) should be a task of the DBA and Sys Admin respectively.
Additionally Windows offers the ability to easily email errors from the windows Event Viewer via the "Attach a task to this log". This is a great method to be notified of errors at the OS or SQL Server level.

Applying these recommendations

As discussed in finding 13 the best approach is to duplicate the SQL Server into a test environment and apply the recommendations there. One at a time, followed by a test cycle.
While a true duplicate environment would be costly it would provide amazing benefits. SQL Server provides a simple tool to gather a workload from production, then run that through the test environment, it will then generate recommendations to improve performance. These can be applied to test and the same workload can be re-executed. This keeps all variables the same between both environments and the workload is also identical allowing for confident performance testing.

Schema Management best practices

Schemas should be used for logical grouping and managing security. See the section titled Using Schemas in SQL Server in Microsoft technet note. This describes how schemas are used to manage security of a single schema for segregation of duties between DBAs and Developers.

Appendix A – High Impact SQL

```
SELECT [t0].[CashDrawerStatusID], [t0].[CashDrawerID], [t0].[CashDrawerStatusTypeID],
[t0].[TotalBalance], [t0].[CashPortion], [t0].[Active]
FROM [dbo].[CashDrawerStatus] AS [t0]
WHERE ([t0].[Active] = 1) AND ([t0].[CashDrawerID] = @p0)


SELECT              CAI.CashDrawerActivityItemID                AS [CashDrawerActivityItemID],
                    CAPM.CashDrawerAcceptedPaymentMethodID      AS
[CashDrawerAcceptedPaymentMethodID],
                    CAPM.DisplayValue                                       AS
[CashDrawerAcceptedPaymentMethodDisplayValue],
                    SP.PaymentNumber                                        AS [PaymentNumber],
                    SP.HasStatement                                                 AS
[HasStatement],
                    ST.StatementNumberID                                AS [StatementNumberID],
                    SP.CustomerNumber                                               AS [CustomerNumber],
                    SP.Adjustment                                           AS [IsAdjustment],
                    Sp.Amount                                                       AS [Amount],
                    CAI.CreatedBy                                           AS [UserName],
                    CAI.CreatedDate                                                 AS
[CreatedDate]
FROM                        CashDrawerActivityItem CAI
INNER JOIN          StatementPayment SP ON SP.StatementPaymentID = CAI.StatementPaymentID
INNER JOIN          CashDrawerAcceptedPaymentMethod CAPM ON CAPM.CashDrawerAcceptedPaymentMethodID =
SP.CashDrawerAcceptedPaymentMethodID
LEFT OUTER JOIN     Statement ST ON ST.StatementID = SP.StatementID
WHERE               CAI.CashDrawerID = @CashDrawerID
AND                 CAI.Active = @True
AND                 CAI.OriginalTransactionTimestamp BETWEEN @FromDate AND @ThroughDate

UNION

-- We want to include all non-statement payments (funding) regardless of date,
-- so the balance will make sense for the user.
SELECT              CAI.CashDrawerActivityItemID                AS [CashDrawerActivityItemID],
                    CAPM.CashDrawerAcceptedPaymentMethodID      AS
[CashDrawerAcceptedPaymentMethodID],
                    CAPM.DisplayValue                                       AS
[CashDrawerAcceptedPaymentMethodDisplayValue],
                    SP.PaymentNumber                                        AS [PaymentNumber],
                    SP.HasStatement                                                 AS
[HasStatement],
                    ST.StatementNumberID                                AS [StatementNumberID],
                    SP.CustomerNumber                                               AS [CustomerNumber],
                    SP.Adjustment                                           AS [IsAdjustment],
                    Sp.Amount                                                       AS [Amount],
                    CAI.CreatedBy                                           AS [UserName],
                    CAI.CreatedDate                                                 AS
[CreatedDate]
FROM                        CashDrawerActivityItem CAI
INNER JOIN          StatementPayment SP ON SP.StatementPaymentID = CAI.StatementPaymentID
INNER JOIN          CashDrawerAcceptedPaymentMethod CAPM ON CAPM.CashDrawerAcceptedPaymentMethodID =
SP.CashDrawerAcceptedPaymentMethodID
LEFT OUTER JOIN     Statement ST ON ST.StatementID = SP.StatementID
WHERE               CAI.CashDrawerID = @CashDrawerID
AND                 CAI.Active = @True
AND                 SP.StatementID IS NULL
AND                 SP.CustomerNumber IS NULL

--AND                       StatementPayment.Active = true

/*
SELECT              PA.CashDrawerPaymentID                              AS [CashDrawerPaymentID],
                    PA.CashDrawerStatementID                            AS [CashDrawerStatementID],
                    APM.CashDrawerAcceptedPaymentMethodID AS [CashDrawerAcceptedPaymentMethodID],
```


4-15

```
                APM.DisplayValue                                                AS
[CashDrawerAcceptedPaymentMethodValue],
                PA.PaymentNumber                                        AS [PaymentNumber],
                CASE
                WHEN PA.AdjustmentATMMProductID IS NULL
                        THEN @False
                        ELSE @True
                END                                                             AS
[Adjustment],
                PA.Amount                                                       AS [Amount],
                PA.OriginalTransactionTimestamp                 AS
[OriginalTransactionTimestamp],
                PA.AdjustmentATMMProductID                      AS [AdjustmentATMMProductID],
                PA.OriginalCashDrawerPaymentID                  AS
[OriginalCashDrawerPaymentID],
                        CASE
                WHEN PA.CashDrawerStatementID IS NULL
                        THEN 'Funded'
                        ELSE cast(PA.CashDrawerStatementID as varchar)
                END                                                             AS
[StatementText]
FROM                    CashDrawerPayment PA
INNER JOIN      CashDrawerAcceptedPaymentMethod APM ON APM.CashDrawerAcceptedPaymentMethodID =
PA.CashDrawerAcceptedPaymentMethodID
WHERE           PA.CashDrawerID = @CashDrawerID
AND             PA.OriginalTransactionTimestamp BETWEEN @FromDate AND @ThroughDate
AND             PA.Active = @True
--AND                   CDPA.ReconciledByCashDrawerActivityID IS NULL
AND             PA.CashDrawerActivitySetID IS NULL
*/


SELECT [t0].[CashDrawerID] AS [CashDrawerID], [t0].[CashDrawerTypeID] AS [CashDrawerTypeID],
[t3].[DisplayValue] AS [CashDrawerTypeDisplayValue], [t0].[CashDrawerPaymentProfileID] AS
[CashDrawerPaymentProfileID], [t4].[DisplayValue] AS [CashDrawerPaymentProfileDisplayValue],
[t1].[CashDrawerStatusTypeID] AS [CashDrawerStatusTypeID], [t2].[DisplayValue] AS
[CashDrawerStatusTypeDisplayValue], [t1].[CashDrawerStatusTypeID] AS
[OriginalCashDrawerStatusTypeID], [t0].[Description], [t1].[TotalBalance] AS [TotalBalance],
[t1].[CashPortion] AS [TotalCash], [t0].[Active]
FROM [dbo].[CashDrawer] AS [t0]
INNER JOIN [dbo].[CashDrawerStatus] AS [t1] ON [t0].[CashDrawerID] = [t1].[CashDrawerID]
INNER JOIN [dbo].[CashDrawerStatusType] AS [t2] ON [t1].[CashDrawerStatusTypeID] =
[t2].[CashDrawerStatusTypeID]
INNER JOIN [dbo].[CashDrawerType] AS [t3] ON [t0].[CashDrawerTypeID] = [t3].[CashDrawerTypeID]
INNER JOIN [dbo].[CashDrawerPaymentProfile] AS [t4] ON [t0].[CashDrawerPaymentProfileID] =
[t4].[CashDrawerPaymentProfileID]
WHERE ([t0].[CashDrawerID] = @p0) AND ([t1].[Active] = 1)</StatementString>


SELECT          CAI.CashDrawerActivityItemID            AS [CashDrawerActivityItemID],
                CAPM.CashDrawerAcceptedPaymentMethodID  AS
[CashDrawerAcceptedPaymentMethodID],
                CAPM.DisplayValue                                       AS
[CashDrawerAcceptedPaymentMethodDisplayValue],
                SP.PaymentNumber                                        AS [PaymentNumber],
                SP.HasStatement                                                 AS
[HasStatement],
                ST.StatementNumberID                            AS [StatementNumberID],
                SP.CustomerNumber                                       AS [CustomerNumber],
                SP.Adjustment                                           AS [IsAdjustment],
                Sp.Amount                                                       AS [Amount],
                CAI.CreatedBy                                           AS [UserName],
                CAI.CreatedDate                                                 AS
[CreatedDate]
FROM                    CashDrawerActivityItem CAI
INNER JOIN      StatementPayment SP ON SP.StatementPaymentID = CAI.StatementPaymentID
INNER JOIN      CashDrawerAcceptedPaymentMethod CAPM ON CAPM.CashDrawerAcceptedPaymentMethodID =
SP.CashDrawerAcceptedPaymentMethodID
LEFT OUTER JOIN Statement ST ON ST.StatementID = SP.StatementID
WHERE           CAI.CashDrawerID = @CashDrawerID
```


4-16

```
AND                 CAI.Active = @True
AND                 CAI.OriginalTransactionTimestamp BETWEEN @FromDate AND @ThroughDate

UNION

-- We want to include all non-statement payments (funding) regardless of date,
-- so the balance will make sense for the user.
SELECT              CAI.CashDrawerActivityItemID                AS [CashDrawerActivityItemID],
                    CAPM.CashDrawerAcceptedPaymentMethodID      AS
[CashDrawerAcceptedPaymentMethodID],
                    CAPM.DisplayValue                                       AS
[CashDrawerAcceptedPaymentMethodDisplayValue],
                    SP.PaymentNumber                                        AS [PaymentNumber],
                    SP.HasStatement                                             AS
[HasStatement],
                    ST.StatementNumberID                                AS [StatementNumberID],
                    SP.CustomerNumber                                       AS [CustomerNumber],
                    SP.Adjustment                                           AS [IsAdjustment],
                    Sp.Amount                                                   AS [Amount],
                    CAI.CreatedBy                                           AS [UserName],
                    CAI.CreatedDate                                             AS
[CreatedDate]
FROM                        CashDrawerActivityItem CAI
INNER JOIN          StatementPayment SP ON SP.StatementPaymentID = CAI.StatementPaymentID
INNER JOIN          CashDrawerAcceptedPaymentMethod CAPM ON CAPM.CashDrawerAcceptedPaymentMethodID =
SP.CashDrawerAcceptedPaymentMethodID
LEFT OUTER JOIN     Statement ST ON ST.StatementID = SP.StatementID
WHERE               CAI.CashDrawerID = @CashDrawerID
AND                 CAI.Active = @True
AND                 SP.StatementID IS NULL
AND                 SP.CustomerNumber IS NULL

--AND                       StatementPayment.Active = true

/*
SELECT              PA.CashDrawerPaymentID                          AS [CashDrawerPaymentID],
                    PA.CashDrawerStatementID                        AS [CashDrawerStatementID],
                    APM.CashDrawerAcceptedPaymentMethodID AS [CashDrawerAcceptedPaymentMethodID],
                    APM.DisplayValue                                            AS
[CashDrawerAcceptedPaymentMethodValue],
                    PA.PaymentNumber                                        AS [PaymentNumber],
                    CASE
                    WHEN PA.AdjustmentATMMProductID IS NULL
                           THEN @False
                           ELSE @True
                    END                                                                 AS
[Adjustment],
                    PA.Amount                                                   AS [Amount],
                    PA.OriginalTransactionTimestamp             AS
[OriginalTransactionTimestamp],
                    PA.AdjustmentATMMProductID                  AS [AdjustmentATMMProductID],
                    PA.OriginalCashDrawerPaymentID              AS
[OriginalCashDrawerPaymentID],
                           CASE
                    WHEN PA.CashDrawerStatementID IS NULL
                           THEN 'Funded'
                           ELSE cast(PA.CashDrawerStatementID as varchar)
                    END                                                                 AS
[StatementText]
FROM                        CashDrawerPayment PA
INNER JOIN          CashDrawerAcceptedPaymentMethod APM ON APM.CashDrawerAcceptedPaymentMethodID =
PA.CashDrawerAcceptedPaymentMethodID
WHERE               PA.CashDrawerID = @CashDrawerID
AND                 PA.OriginalTransactionTimestamp BETWEEN @FromDate AND @ThroughDate
AND                 PA.Active = @True
--AND                       CDPA.ReconciledByCashDrawerActivityID IS NULL
AND                 PA.CashDrawerActivitySetID IS NULL
*/
```

```
SELECT              CAI.CashDrawerActivityItemID                AS [CashDrawerActivityItemID],
                    CAPM.CashDrawerAcceptedPaymentMethodID      AS
[CashDrawerAcceptedPaymentMethodID],
                    CAPM.DisplayValue                                   AS
[CashDrawerAcceptedPaymentMethodDisplayValue],
                    SP.PaymentNumber                                    AS [PaymentNumber],
                    SP.HasStatement                                             AS
[HasStatement],
                    ST.StatementNumberID                        AS [StatementNumberID],
                    SP.CustomerNumber                                   AS [CustomerNumber],
                    SP.Adjustment                                       AS [IsAdjustment],
                    Sp.Amount                                                   AS [Amount],
                    CAI.CreatedBy                                       AS [UserName],
                    CAI.CreatedDate                                             AS
[CreatedDate]
FROM                        CashDrawerActivityItem CAI
INNER JOIN          StatementPayment SP ON SP.StatementPaymentID = CAI.StatementPaymentID
INNER JOIN          CashDrawerAcceptedPaymentMethod CAPM ON CAPM.CashDrawerAcceptedPaymentMethodID =
SP.CashDrawerAcceptedPaymentMethodID
LEFT OUTER JOIN     Statement ST ON ST.StatementID = SP.StatementID
WHERE               CAI.CashDrawerID = @CashDrawerID
AND                 CAI.Active = @True
AND                 CAI.OriginalTransactionTimestamp BETWEEN @FromDate AND @ThroughDate

UNION

-- We want to include all non-statement payments (funding) regardless of date,
-- so the balance will make sense for the user.
SELECT              CAI.CashDrawerActivityItemID                AS [CashDrawerActivityItemID],
                    CAPM.CashDrawerAcceptedPaymentMethodID      AS
[CashDrawerAcceptedPaymentMethodID],
                    CAPM.DisplayValue                                   AS
[CashDrawerAcceptedPaymentMethodDisplayValue],
                    SP.PaymentNumber                                    AS [PaymentNumber],
                    SP.HasStatement                                             AS
[HasStatement],
                    ST.StatementNumberID                        AS [StatementNumberID],
                    SP.CustomerNumber                                   AS [CustomerNumber],
                    SP.Adjustment                                       AS [IsAdjustment],
                    Sp.Amount                                                   AS [Amount],
                    CAI.CreatedBy                                       AS [UserName],
                    CAI.CreatedDate                                             AS
[CreatedDate]
FROM                        CashDrawerActivityItem CAI
INNER JOIN          StatementPayment SP ON SP.StatementPaymentID = CAI.StatementPaymentID
INNER JOIN          CashDrawerAcceptedPaymentMethod CAPM ON CAPM.CashDrawerAcceptedPaymentMethodID =
SP.CashDrawerAcceptedPaymentMethodID
LEFT OUTER JOIN     Statement ST ON ST.StatementID = SP.StatementID
WHERE               CAI.CashDrawerID = @CashDrawerID
AND                 CAI.Active = @True
AND                 SP.StatementID IS NULL
AND                 SP.CustomerNumber IS NULL

--AND                       StatementPayment.Active = true

/*
SELECT              PA.CashDrawerPaymentID                      AS [CashDrawerPaymentID],
                    PA.CashDrawerStatementID                    AS [CashDrawerStatementID],
                    APM.CashDrawerAcceptedPaymentMethodID AS [CashDrawerAcceptedPaymentMethodID],
                    APM.DisplayValue                                            AS
[CashDrawerAcceptedPaymentMethodValue],
                    PA.PaymentNumber                                    AS [PaymentNumber],
                    CASE
                    WHEN PA.AdjustmentATMMProductID IS NULL
                            THEN @False
                            ELSE @True
                    END                                                                 AS
[Adjustment],
                    PA.Amount                                                   AS [Amount],
                    PA.OriginalTransactionTimestamp             AS
[OriginalTransactionTimestamp],
```

```
                    PA.AdjustmentATMMProductID                          AS [AdjustmentATMMProductID],
                    PA.OriginalCashDrawerPaymentID                      AS
[OriginalCashDrawerPaymentID],
                          CASE
                    WHEN PA.CashDrawerStatementID IS NULL
                          THEN 'Funded'
                          ELSE cast(PA.CashDrawerStatementID as varchar)
                    END                                                                 AS
[StatementText]
FROM                      CashDrawerPayment PA
INNER JOIN          CashDrawerAcceptedPaymentMethod APM ON APM.CashDrawerAcceptedPaymentMethodID =
PA.CashDrawerAcceptedPaymentMethodID
WHERE               PA.CashDrawerID = @CashDrawerID
AND                 PA.OriginalTransactionTimestamp BETWEEN @FromDate AND @ThroughDate
AND                 PA.Active = @True
--AND                     CDPA.ReconciledByCashDrawerActivityID IS NULL
AND                 PA.CashDrawerActivitySetID IS NULL
*/


SELECT [t0].[CashDrawerID] AS [CashDrawerID], [t0].[CashDrawerTypeID] AS [CashDrawerTypeID],
[t3].[DisplayValue] AS [CashDrawerTypeDisplayValue], [t0].[CashDrawerPaymentProfileID] AS
[CashDrawerPaymentProfileID], [t4].[DisplayValue] AS [CashDrawerPaymentProfileDisplayValue],
[t1].[CashDrawerStatusTypeID] AS [CashDrawerStatusTypeID], [t2].[DisplayValue] AS
[CashDrawerStatusTypeDisplayValue], [t1].[CashDrawerStatusTypeID] AS
[OriginalCashDrawerStatusTypeID], [t0].[Description], [t1].[TotalBalance] AS [TotalBalance],
[t1].[CashPortion] AS [TotalCash], [t0].[Active]
FROM [dbo].[CashDrawer] AS [t0]
INNER JOIN [dbo].[CashDrawerStatus] AS [t1] ON [t0].[CashDrawerID] = [t1].[CashDrawerID]
INNER JOIN [dbo].[CashDrawerStatusType] AS [t2] ON [t1].[CashDrawerStatusTypeID] =
[t2].[CashDrawerStatusTypeID]
INNER JOIN [dbo].[CashDrawerType] AS [t3] ON [t0].[CashDrawerTypeID] = [t3].[CashDrawerTypeID]
INNER JOIN [dbo].[CashDrawerPaymentProfile] AS [t4] ON [t0].[CashDrawerPaymentProfileID] =
[t4].[CashDrawerPaymentProfileID]
INNER JOIN [dbo].[CashDrawerLocationProfile] AS [t5] ON [t0].[CashDrawerID] = [t5].[CashDrawerID]
WHERE ([t5].[LocationProfileID] = @p0) AND ([t0].[Active] = 1) AND ([t1].[Active] = 1) AND
([t5].[Active] = 1)
```

Appendix B – Recommended Indexes and Statistics (ATTM)

```
use [ATMM]
go

CREATE NONCLUSTERED INDEX [_dta_index_CashDrawerActivityItem_7_841418417__K3_K7_K5_K4_K1_K8_K9] ON
[dbo].[CashDrawerActivityItem]
(
    [CashDrawerID] ASC,
    [Active] ASC,
    [OriginalTransactionTimestamp] ASC,
    [StatementPaymentID] ASC,
    [CashDrawerActivityItemID] ASC,
    [CreatedBy] ASC,
    [CreatedDate] ASC
)WITH (SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = ON) ON [PRIMARY]
go

CREATE STATISTICS [_dta_stat_841418417_5_3] ON
[dbo].[CashDrawerActivityItem]([OriginalTransactionTimestamp], [CashDrawerID])
go

CREATE STATISTICS [_dta_stat_841418417_4_5_3] ON [dbo].[CashDrawerActivityItem]([StatementPaymentID],
[OriginalTransactionTimestamp], [CashDrawerID])
go

CREATE STATISTICS [_dta_stat_841418417_1_5_3] ON
[dbo].[CashDrawerActivityItem]([CashDrawerActivityItemID], [OriginalTransactionTimestamp],
[CashDrawerID])
go

CREATE STATISTICS [_dta_stat_841418417_3_7_1_5] ON [dbo].[CashDrawerActivityItem]([CashDrawerID],
[Active], [CashDrawerActivityItemID], [OriginalTransactionTimestamp])
go

CREATE STATISTICS [_dta_stat_841418417_1_4_3_7] ON
[dbo].[CashDrawerActivityItem]([CashDrawerActivityItemID], [StatementPaymentID], [CashDrawerID],
[Active])
go

CREATE STATISTICS [_dta_stat_841418417_1_8_9_3] ON
[dbo].[CashDrawerActivityItem]([CashDrawerActivityItemID], [CreatedBy], [CreatedDate],
[CashDrawerID])
go

CREATE STATISTICS [_dta_stat_841418417_1_3_7_8_9] ON
[dbo].[CashDrawerActivityItem]([CashDrawerActivityItemID], [CashDrawerID], [Active], [CreatedBy],
[CreatedDate])
go

CREATE STATISTICS [_dta_stat_841418417_1_8_9_5_3_7] ON
[dbo].[CashDrawerActivityItem]([CashDrawerActivityItemID], [CreatedBy], [CreatedDate],
[OriginalTransactionTimestamp], [CashDrawerID], [Active])
go

CREATE NONCLUSTERED INDEX [_dta_index_StatementPayment_7_120035859__K1_K4_K2_K3_K7_K12_K11_K6] ON
[dbo].[StatementPayment]
(
    [StatementPaymentID] ASC,
    [CashDrawerAcceptedPaymentMethodID] ASC,
    [StatementID] ASC,
    [CustomerNumber] ASC,
    [PaymentNumber] ASC,
    [HasStatement] ASC,
    [Adjustment] ASC,
```

```
    [Amount] ASC
)WITH (SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = ON) ON [PRIMARY]
go

CREATE NONCLUSTERED INDEX [_dta_index_StatementPayment_7_120035859__K2_K3_K1_K4_K7_K12_K11_K6] ON
[dbo].[StatementPayment]
(
    [StatementID] ASC,
    [CustomerNumber] ASC,
    [StatementPaymentID] ASC,
    [CashDrawerAcceptedPaymentMethodID] ASC,
    [PaymentNumber] ASC,
    [HasStatement] ASC,
    [Adjustment] ASC,
    [Amount] ASC
)WITH (SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = ON) ON [PRIMARY]
go

CREATE STATISTICS [_dta_stat_120035859_1_2] ON [dbo].[StatementPayment]([StatementPaymentID],
[StatementID])
go

CREATE STATISTICS [_dta_stat_120035859_4_2_3] ON
[dbo].[StatementPayment]([CashDrawerAcceptedPaymentMethodID], [StatementID], [CustomerNumber])
go

CREATE STATISTICS [_dta_stat_120035859_7_12_11_6] ON [dbo].[StatementPayment]([PaymentNumber],
[HasStatement], [Adjustment], [Amount])
go

CREATE STATISTICS [_dta_stat_120035859_7_12_3_11_6_2] ON [dbo].[StatementPayment]([PaymentNumber],
[HasStatement], [CustomerNumber], [Adjustment], [Amount], [StatementID])
go

CREATE STATISTICS [_dta_stat_120035859_4_1_7_12_11_6] ON
[dbo].[StatementPayment]([CashDrawerAcceptedPaymentMethodID], [StatementPaymentID], [PaymentNumber],
[HasStatement], [Adjustment], [Amount])
go

CREATE STATISTICS [_dta_stat_120035859_1_3_4_2_7_12_11_6] ON
[dbo].[StatementPayment]([StatementPaymentID], [CustomerNumber], [CashDrawerAcceptedPaymentMethodID],
[StatementID], [PaymentNumber], [HasStatement], [Adjustment], [Amount])
go

CREATE NONCLUSTERED INDEX [_dta_index_Statement_7_523513294__K1_K2] ON [dbo].[Statement]
(
    [StatementID] ASC,
    [StatementNumberID] ASC
)WITH (SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = ON) ON [PRIMARY]
go

CREATE STATISTICS [_dta_stat_523513294_2_1] ON [dbo].[Statement]([StatementNumberID], [StatementID])
go

CREATE NONCLUSTERED INDEX [_dta_index_CashDrawerStatus_7_726605977__K6_K3_K2_K1_4_5] ON
[dbo].[CashDrawerStatus]
(
    [Active] ASC,
    [CashDrawerStatusTypeID] ASC,
    [CashDrawerID] ASC,
    [CashDrawerStatusID] ASC
)
INCLUDE ( [TotalBalance],
[CashPortion]) WITH (SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = ON) ON
[PRIMARY]
go

CREATE NONCLUSTERED INDEX [_dta_index_CashDrawerStatus_7_726605977__K2_K6_K1_3_4_5] ON
[dbo].[CashDrawerStatus]
(
    [CashDrawerID] ASC,
```

```
    [Active] ASC,
    [CashDrawerStatusID] ASC
)
INCLUDE ( [CashDrawerStatusTypeID],
[TotalBalance],
[CashPortion]) WITH (SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = ON) ON
[PRIMARY]
go

CREATE STATISTICS [_dta_stat_726605977_6_1] ON [dbo].[CashDrawerStatus]([Active],
[CashDrawerStatusID])
go

CREATE STATISTICS [_dta_stat_726605977_1_2] ON [dbo].[CashDrawerStatus]([CashDrawerStatusID],
[CashDrawerID])
go

CREATE STATISTICS [_dta_stat_726605977_3_1_2] ON [dbo].[CashDrawerStatus]([CashDrawerStatusTypeID],
[CashDrawerStatusID], [CashDrawerID])
go

CREATE STATISTICS [_dta_stat_726605977_2_6_1_3] ON [dbo].[CashDrawerStatus]([CashDrawerID], [Active],
[CashDrawerStatusID], [CashDrawerStatusTypeID])
go

CREATE STATISTICS [_dta_stat_322464573_2_3] ON [dbo].[CashDrawerLocationProfile]([CashDrawerID],
[LocationProfileID])
go

CREATE STATISTICS [_dta_stat_322464573_5_3_2] ON [dbo].[CashDrawerLocationProfile]([Active],
[LocationProfileID], [CashDrawerID])
go

CREATE STATISTICS [_dta_stat_2053946739_2_3] ON [dbo].[CashDrawer]([CashDrawerTypeID],
[CashDrawerPaymentProfileID])
go

CREATE STATISTICS [_dta_stat_2053946739_5_1] ON [dbo].[CashDrawer]([Active], [CashDrawerID])
go

CREATE STATISTICS [_dta_stat_2053946739_5_2_1] ON [dbo].[CashDrawer]([Active], [CashDrawerTypeID],
[CashDrawerID])
go

CREATE STATISTICS [_dta_stat_2053946739_1_2_3] ON [dbo].[CashDrawer]([CashDrawerID],
[CashDrawerTypeID], [CashDrawerPaymentProfileID])
go

CREATE STATISTICS [_dta_stat_2053946739_5_3_1_2] ON [dbo].[CashDrawer]([Active],
[CashDrawerPaymentProfileID], [CashDrawerID], [CashDrawerTypeID])
Go
```

Appendix C – Recommended Statistics (MOVRS)

```
use [MOVRS]
go

CREATE STATISTICS [_dta_stat_88165347_2_8] ON [dbo].[Vehicle]([VehicleNumber], [VehicleTypeID])
go

CREATE STATISTICS [_dta_stat_88165347_22_2] ON [dbo].[Vehicle]([EffectiveDate], [VehicleNumber])
go

CREATE STATISTICS [_dta_stat_88165347_1_8] ON [dbo].[Vehicle]([VehicleID], [VehicleTypeID])
go
```

```
CREATE STATISTICS [_dta_stat_88165347_2_22_23] ON [dbo].[Vehicle]([VehicleNumber], [EffectiveDate],
[EndDate])
go

CREATE STATISTICS [_dta_stat_88165347_22_23_2_7_6_5_8] ON [dbo].[Vehicle]([EffectiveDate], [EndDate],
[VehicleNumber], [StyleID], [MakeID], [ModelID], [VehicleTypeID])
go

CREATE STATISTICS [_dta_stat_88165347_8_2_7_19_18_6_5_10_9] ON [dbo].[Vehicle]([VehicleTypeID],
[VehicleNumber], [StyleID], [TonCodeID], [NumberOfWheelsID], [MakeID], [ModelID], [FuelTypeID],
[NumberOfCylindersID])
go

CREATE STATISTICS [_dta_stat_88165347_1_2_8_7_19_18_6_5_10_9] ON [dbo].[Vehicle]([VehicleID],
[VehicleNumber], [VehicleTypeID], [StyleID], [TonCodeID], [NumberOfWheelsID], [MakeID], [ModelID],
[FuelTypeID], [NumberOfCylindersID])
go

CREATE STATISTICS [_dta_stat_463886757_12_1] ON [dbo].[Transaction]([PriorityMailing],
[TransactionID])
go

CREATE STATISTICS [_dta_stat_463886757_1_4] ON [dbo].[Transaction]([TransactionID],
[SourceSystemTransactionTypeID])
go

CREATE STATISTICS [_dta_stat_463886757_4_5_7] ON [dbo].[Transaction]([SourceSystemTransactionTypeID],
[LocationProfileID], [WIPSetItemID])
go

CREATE STATISTICS [_dta_stat_463886757_3_1_4] ON [dbo].[Transaction]([PermitID], [TransactionID],
[SourceSystemTransactionTypeID])
go

CREATE STATISTICS [_dta_stat_463886757_4_5_1] ON [dbo].[Transaction]([SourceSystemTransactionTypeID],
[LocationProfileID], [TransactionID])
go

CREATE STATISTICS [_dta_stat_463886757_1_7_4] ON [dbo].[Transaction]([TransactionID], [WIPSetItemID],
[SourceSystemTransactionTypeID])
go

CREATE STATISTICS [_dta_stat_463886757_1_5_7_4] ON [dbo].[Transaction]([TransactionID],
[LocationProfileID], [WIPSetItemID], [SourceSystemTransactionTypeID])
go

CREATE STATISTICS [_dta_stat_52741278_3_10] ON [dbo].[VehicleOwnershipVestedParty]([CustomerNumber],
[SecurityInterestMailingAddressID])
go

CREATE STATISTICS [_dta_stat_52741278_2_8_7] ON
[dbo].[VehicleOwnershipVestedParty]([VehicleOwnershipID], [ReleaseDate],
[VehicleOwnershipControlTypeID])
go

CREATE STATISTICS [_dta_stat_52741278_8_3_7] ON [dbo].[VehicleOwnershipVestedParty]([ReleaseDate],
[CustomerNumber], [VehicleOwnershipControlTypeID])
go

CREATE STATISTICS [_dta_stat_52741278_3_1_10] ON
[dbo].[VehicleOwnershipVestedParty]([CustomerNumber], [VehicleOwnershipVestedPartyID],
[SecurityInterestMailingAddressID])
go

CREATE STATISTICS [_dta_stat_52741278_3_5_12_9] ON
[dbo].[VehicleOwnershipVestedParty]([CustomerNumber], [VestmentTypeID], [Priority], [AddDate])
go

CREATE STATISTICS [_dta_stat_52741278_7_3_5_2] ON
[dbo].[VehicleOwnershipVestedParty]([VehicleOwnershipControlTypeID], [CustomerNumber],
[VestmentTypeID], [VehicleOwnershipID])
```

```
go

CREATE STATISTICS [_dta_stat_52741278_7_2_3_10] ON
[dbo].[VehicleOwnershipVestedParty]([VehicleOwnershipControlTypeID], [VehicleOwnershipID],
[CustomerNumber], [SecurityInterestMailingAddressID])
go

CREATE STATISTICS [_dta_stat_52741278_5_12_9_2] ON
[dbo].[VehicleOwnershipVestedParty]([VestmentTypeID], [Priority], [AddDate], [VehicleOwnershipID])
go

CREATE STATISTICS [_dta_stat_52741278_5_12_10_2_8] ON
[dbo].[VehicleOwnershipVestedParty]([VestmentTypeID], [Priority], [SecurityInterestMailingAddressID],
[VehicleOwnershipID], [ReleaseDate])
go

CREATE STATISTICS [_dta_stat_52741278_8_7_3_5_12] ON
[dbo].[VehicleOwnershipVestedParty]([ReleaseDate], [VehicleOwnershipControlTypeID], [CustomerNumber],
[VestmentTypeID], [Priority])
go

CREATE STATISTICS [_dta_stat_52741278_3_2_5_12_9] ON
[dbo].[VehicleOwnershipVestedParty]([CustomerNumber], [VehicleOwnershipID], [VestmentTypeID],
[Priority], [AddDate])
go

CREATE STATISTICS [_dta_stat_52741278_2_3_7_8_5_12] ON
[dbo].[VehicleOwnershipVestedParty]([VehicleOwnershipID], [CustomerNumber],
[VehicleOwnershipControlTypeID], [ReleaseDate], [VestmentTypeID], [Priority])
go

CREATE STATISTICS [_dta_stat_52741278_7_3_10_8_5_12] ON
[dbo].[VehicleOwnershipVestedParty]([VehicleOwnershipControlTypeID], [CustomerNumber],
[SecurityInterestMailingAddressID], [ReleaseDate], [VestmentTypeID], [Priority])
go

CREATE STATISTICS [_dta_stat_52741278_8_10_7_2_3_5_12] ON
[dbo].[VehicleOwnershipVestedParty]([ReleaseDate], [SecurityInterestMailingAddressID],
[VehicleOwnershipControlTypeID], [VehicleOwnershipID], [CustomerNumber], [VestmentTypeID],
[Priority])
go

CREATE STATISTICS [_dta_stat_215501834_6_1] ON [dbo].[Title]([TitleNumber], [TitleID])
go

CREATE STATISTICS [_dta_stat_215501834_5_3_4] ON [dbo].[Title]([TitleStatusTypeID],
[LocationProfileID], [TitleTypeID])
go

CREATE STATISTICS [_dta_stat_215501834_2_1_5] ON [dbo].[Title]([VehicleOwnershipID], [TitleID],
[TitleStatusTypeID])
go

CREATE STATISTICS [_dta_stat_215501834_1_5_3_4] ON [dbo].[Title]([TitleID], [TitleStatusTypeID],
[LocationProfileID], [TitleTypeID])
go

CREATE STATISTICS [_dta_stat_1920555913_1_10] ON
[dbo].[VehicleRegistrationDetail]([VehicleRegistrationDetailID], [Penalty])
go

CREATE STATISTICS [_dta_stat_1920555913_9_13] ON [dbo].[VehicleRegistrationDetail]([CurrentOwner],
[RegistrationEndDate])
go

CREATE STATISTICS [_dta_stat_1920555913_9_10] ON [dbo].[VehicleRegistrationDetail]([CurrentOwner],
[Penalty])
go

CREATE STATISTICS [_dta_stat_1920555913_1_9_10] ON
[dbo].[VehicleRegistrationDetail]([VehicleRegistrationDetailID], [CurrentOwner], [Penalty])
```

```
go

CREATE STATISTICS [_dta_stat_1920555913_10_4_2] ON [dbo].[VehicleRegistrationDetail]([Penalty],
[VehicleRegistrationDetailUsageTypeID], [VehicleRegistrationID])
go

CREATE STATISTICS [_dta_stat_1920555913_10_13_2] ON [dbo].[VehicleRegistrationDetail]([Penalty],
[RegistrationEndDate], [VehicleRegistrationID])
go

CREATE STATISTICS [_dta_stat_1920555913_10_1_2] ON [dbo].[VehicleRegistrationDetail]([Penalty],
[VehicleRegistrationDetailID], [VehicleRegistrationID])
go

CREATE STATISTICS [_dta_stat_1920555913_2_1_3] ON
[dbo].[VehicleRegistrationDetail]([VehicleRegistrationID], [VehicleRegistrationDetailID],
[VehicleRegistrationPlateID])
go

CREATE STATISTICS [_dta_stat_1920555913_2_10_9] ON
[dbo].[VehicleRegistrationDetail]([VehicleRegistrationID], [Penalty], [CurrentOwner])
go

CREATE STATISTICS [_dta_stat_1920555913_2_4_3_1] ON
[dbo].[VehicleRegistrationDetail]([VehicleRegistrationID], [VehicleRegistrationDetailUsageTypeID],
[VehicleRegistrationPlateID], [VehicleRegistrationDetailID])
go

CREATE STATISTICS [_dta_stat_1920555913_2_9_10_3] ON
[dbo].[VehicleRegistrationDetail]([VehicleRegistrationID], [CurrentOwner], [Penalty],
[VehicleRegistrationPlateID])
go

CREATE STATISTICS [_dta_stat_1920555913_7_2_10_9] ON
[dbo].[VehicleRegistrationDetail]([DeclaredWeight], [VehicleRegistrationID], [Penalty],
[CurrentOwner])
go

CREATE STATISTICS [_dta_stat_1920555913_9_4_2_10] ON
[dbo].[VehicleRegistrationDetail]([CurrentOwner], [VehicleRegistrationDetailUsageTypeID],
[VehicleRegistrationID], [Penalty])
go

CREATE STATISTICS [_dta_stat_1920555913_13_2_9_10_7] ON
[dbo].[VehicleRegistrationDetail]([RegistrationEndDate], [VehicleRegistrationID], [CurrentOwner],
[Penalty], [DeclaredWeight])
go

CREATE STATISTICS [_dta_stat_1920555913_3_13_2_9_10] ON
[dbo].[VehicleRegistrationDetail]([VehicleRegistrationPlateID], [RegistrationEndDate],
[VehicleRegistrationID], [CurrentOwner], [Penalty])
go

CREATE STATISTICS [_dta_stat_1920555913_1_3_13_2_9] ON
[dbo].[VehicleRegistrationDetail]([VehicleRegistrationDetailID], [VehicleRegistrationPlateID],
[RegistrationEndDate], [VehicleRegistrationID], [CurrentOwner])
go

CREATE STATISTICS [_dta_stat_1920555913_2_9_10_1_3_13] ON
[dbo].[VehicleRegistrationDetail]([VehicleRegistrationID], [CurrentOwner], [Penalty],
[VehicleRegistrationDetailID], [VehicleRegistrationPlateID], [RegistrationEndDate])
go

CREATE STATISTICS [_dta_stat_815888011_21_1] ON [dbo].[VehicleRegistration]([IssueWPPermit],
[VehicleRegistrationID])
go

CREATE STATISTICS [_dta_stat_815888011_21_16] ON [dbo].[VehicleRegistration]([IssueWPPermit],
[RegistrationEndDate])
go
```

```
CREATE STATISTICS [_dta_stat_815888011_1_12] ON [dbo].[VehicleRegistration]([VehicleRegistrationID],
[VehicleID])
go

CREATE STATISTICS [_dta_stat_815888011_1_10] ON [dbo].[VehicleRegistration]([VehicleRegistrationID],
[VehicleRegistrationCountyID])
go

CREATE STATISTICS [_dta_stat_815888011_11_1] ON [dbo].[VehicleRegistration]([CreditRefundStatusID],
[VehicleRegistrationID])
go

CREATE STATISTICS [_dta_stat_815888011_16_10_1] ON [dbo].[VehicleRegistration]([RegistrationEndDate],
[VehicleRegistrationCountyID], [VehicleRegistrationID])
go

CREATE STATISTICS [_dta_stat_815888011_8_11_9] ON
[dbo].[VehicleRegistration]([VehicleRegistrationDispositionTypeID], [CreditRefundStatusID],
[VehicleRegistrationDispositionDetailTypeID])
go

CREATE STATISTICS [_dta_stat_815888011_7_10_3] ON
[dbo].[VehicleRegistration]([VehicleRegistrationUsageTypeID], [VehicleRegistrationCountyID],
[VehicleOwnershipID])
go

CREATE STATISTICS [_dta_stat_815888011_1_7_10] ON
[dbo].[VehicleRegistration]([VehicleRegistrationID], [VehicleRegistrationUsageTypeID],
[VehicleRegistrationCountyID])
go

CREATE STATISTICS [_dta_stat_815888011_1_3_7_10] ON
[dbo].[VehicleRegistration]([VehicleRegistrationID], [VehicleOwnershipID],
[VehicleRegistrationUsageTypeID], [VehicleRegistrationCountyID])
go

CREATE STATISTICS [_dta_stat_815888011_1_8_11_9] ON
[dbo].[VehicleRegistration]([VehicleRegistrationID], [VehicleRegistrationDispositionTypeID],
[CreditRefundStatusID], [VehicleRegistrationDispositionDetailTypeID])
go

CREATE STATISTICS [_dta_stat_2070844477_1_3_21] ON [dbo].[VehicleOwnership]([VehicleOwnershipID],
[VehicleNumber], [CreatedDate])
go

CREATE STATISTICS [_dta_stat_2070844477_1_5_21] ON [dbo].[VehicleOwnership]([VehicleOwnershipID],
[CurrentTitleID], [CreatedDate])
go

CREATE STATISTICS [_dta_stat_2070844477_21_14_1_5] ON [dbo].[VehicleOwnership]([CreatedDate],
[ResidentOwnership], [VehicleOwnershipID], [CurrentTitleID])
go

CREATE STATISTICS [_dta_stat_2070844477_3_1_5_21] ON [dbo].[VehicleOwnership]([VehicleNumber],
[VehicleOwnershipID], [CurrentTitleID], [CreatedDate])
go

CREATE STATISTICS [_dta_stat_2070844477_21_14_3_1_5] ON [dbo].[VehicleOwnership]([CreatedDate],
[ResidentOwnership], [VehicleNumber], [VehicleOwnershipID], [CurrentTitleID])
go

CREATE STATISTICS [_dta_stat_257266009_2_25_26] ON [dbo].[VehicleAttributeExtension]([VehicleNumber],
[EffectiveDate], [EndDate])
go

CREATE STATISTICS [_dta_stat_257266009_2_5_6_7_8_1] ON
[dbo].[VehicleAttributeExtension]([VehicleNumber], [HullMaterialTypeID], [BoatTypeID],
[PropulsionTypeID], [MotorHomeClassID], [VehicleAttributeExtensionID])
go
```

```
CREATE STATISTICS [_dta_stat_257266009_1_25_2_5_6_7_8] ON
[dbo].[VehicleAttributeExtension]([VehicleAttributeExtensionID], [EffectiveDate], [VehicleNumber],
[HullMaterialTypeID], [BoatTypeID], [PropulsionTypeID], [MotorHomeClassID])
go


CREATE STATISTICS [_dta_stat_1563438381_7_2_3_4] ON [dbo].[VehicleColor]([EffectiveDate],
[VehicleNumber], [Color1ID], [Color2ID])
go

CREATE STATISTICS [_dta_stat_1563438381_2_3_4_5_7] ON [dbo].[VehicleColor]([VehicleNumber],
[Color1ID], [Color2ID], [Color3ID], [EffectiveDate])
go

CREATE STATISTICS [_dta_stat_1915439635_1_6] ON
[dbo].[VehicleOdometerReading]([VehicleOdometerReadingID], [OdometerUnitID])
go

CREATE STATISTICS [_dta_stat_1915439635_1_8_9] ON
[dbo].[VehicleOdometerReading]([VehicleOdometerReadingID], [EffectiveDate], [EndDate])
go

CREATE STATISTICS [_dta_stat_1915439635_8_1_6] ON [dbo].[VehicleOdometerReading]([EffectiveDate],
[VehicleOdometerReadingID], [OdometerUnitID])
go

CREATE STATISTICS [_dta_stat_1915439635_8_2_1_6] ON [dbo].[VehicleOdometerReading]([EffectiveDate],
[VehicleNumber], [VehicleOdometerReadingID], [OdometerUnitID])
go

CREATE STATISTICS [_dta_stat_1915439635_2_1_6_5] ON [dbo].[VehicleOdometerReading]([VehicleNumber],
[VehicleOdometerReadingID], [OdometerUnitID], [OdometerReadingTypeID])
go

CREATE STATISTICS [_dta_stat_1915439635_2_3_1_8_9] ON [dbo].[VehicleOdometerReading]([VehicleNumber],
[VehicleOwnershipID], [VehicleOdometerReadingID], [EffectiveDate], [EndDate])
go

CREATE STATISTICS [_dta_stat_1915439635_2_1_5_8_6] ON [dbo].[VehicleOdometerReading]([VehicleNumber],
[VehicleOdometerReadingID], [OdometerReadingTypeID], [EffectiveDate], [OdometerUnitID])
go

CREATE STATISTICS [_dta_stat_1915439635_2_3_1_5_8_9] ON
[dbo].[VehicleOdometerReading]([VehicleNumber], [VehicleOwnershipID], [VehicleOdometerReadingID],
[OdometerReadingTypeID], [EffectiveDate], [EndDate])
go

CREATE STATISTICS [_dta_stat_1162317238_3] ON
[dbo].[VehicleRegistrationPropertyTaxSegment]([PropertyTaxSegmentID])
go

CREATE STATISTICS [_dta_stat_1130981093_2_1] ON [dbo].[Plate]([PlateTypeID], [PlateID])
go

CREATE STATISTICS [_dta_stat_1130981093_4_3] ON [dbo].[Plate]([PlateNumber], [PlateConfigurationID])
go

CREATE STATISTICS [_dta_stat_1130981093_1_4_3] ON [dbo].[Plate]([PlateID], [PlateNumber],
[PlateConfigurationID])
go

CREATE STATISTICS [_dta_stat_1130981093_1_4_2] ON [dbo].[Plate]([PlateID], [PlateNumber],
[PlateTypeID])
go

CREATE STATISTICS [_dta_stat_1130981093_2_4_6_1] ON [dbo].[Plate]([PlateTypeID], [PlateNumber],
[Personalized], [PlateID])
go

CREATE STATISTICS [_dta_stat_151501606_4] ON [dbo].[TransactionCredential]([DeviceDocumentClassID])
go
```

```
CREATE STATISTICS [_dta_stat_151501606_3_4] ON
[dbo].[TransactionCredential]([TransactionCredentialTypeID], [DeviceDocumentClassID])
go

CREATE STATISTICS [_dta_stat_151501606_2_3_4] ON [dbo].[TransactionCredential]([TransactionID],
[TransactionCredentialTypeID], [DeviceDocumentClassID])
go

CREATE STATISTICS [_dta_stat_1235365503_1_3] ON
[dbo].[TransactionTitleRegistration]([TransactionTitleRegistrationID], [VehicleRegistrationID])
go

CREATE STATISTICS [_dta_stat_1235365503_4_2] ON [dbo].[TransactionTitleRegistration]([TitleID],
[TransactionID])
go

CREATE STATISTICS [_dta_stat_1235365503_2_1] ON [dbo].[TransactionTitleRegistration]([TransactionID],
[TransactionTitleRegistrationID])
go

CREATE STATISTICS [_dta_stat_1235365503_4_1_2] ON [dbo].[TransactionTitleRegistration]([TitleID],
[TransactionTitleRegistrationID], [TransactionID])
go

CREATE STATISTICS [_dta_stat_1235365503_1_2_3] ON
[dbo].[TransactionTitleRegistration]([TransactionTitleRegistrationID], [TransactionID],
[VehicleRegistrationID])
go

CREATE STATISTICS [_dta_stat_1813867266_1_2] ON [dbo].[Document]([DocumentID], [TransactionID])
go

CREATE STATISTICS [_dta_stat_1813867266_1_3_2] ON [dbo].[Document]([DocumentID], [DocumentTypeID],
[TransactionID])
go

CREATE STATISTICS [_dta_stat_898752392_16] ON [dbo].[VehicleSalesTaxCustomer]([CreatedDate])
go

CREATE STATISTICS [_dta_stat_898752392_4_16] ON [dbo].[VehicleSalesTaxCustomer]([CustomerNumber],
[CreatedDate])
go

CREATE STATISTICS [_dta_stat_666315471_1_2_17] ON [dbo].[Permit]([PermitID], [PermitTypeID],
[Active])
go

CREATE STATISTICS [_dta_stat_666315471_1_4_2_17] ON [dbo].[Permit]([PermitID], [VehicleNumber],
[PermitTypeID], [Active])
go

CREATE STATISTICS [_dta_stat_1074388634_1_17] ON
[dbo].[NonResidentVehicleOwnership]([NonResidentVehicleOwnershipID], [EffectiveDate])
go

CREATE STATISTICS [_dta_stat_1074388634_2_1] ON
[dbo].[NonResidentVehicleOwnership]([VehicleOwnershipID], [NonResidentVehicleOwnershipID])
go

CREATE STATISTICS [_dta_stat_1074388634_17_2_1] ON
[dbo].[NonResidentVehicleOwnership]([EffectiveDate], [VehicleOwnershipID],
[NonResidentVehicleOwnershipID])
go

CREATE STATISTICS [_dta_stat_1586390458_3_1] ON
[dbo].[VehicleAttributeDataChangeFlag]([VehicleNumber], [VehicleAttributeDataChangeFlagID])
go

CREATE STATISTICS [_dta_stat_1586390458_3_2] ON
[dbo].[VehicleAttributeDataChangeFlag]([VehicleNumber], [VehicleAttributeDataChangeFlagTypeID])
```

```
go

CREATE STATISTICS [_dta_stat_1586390458_2_1_3] ON
[dbo].[VehicleAttributeDataChangeFlag]([VehicleAttributeDataChangeFlagTypeID],
[VehicleAttributeDataChangeFlagID], [VehicleNumber])
go

CREATE STATISTICS [_dta_stat_1369601955_10_1] ON [dbo].[Model]([Active], [ModelID])
go

CREATE STATISTICS [_dta_stat_2088224527_1_5] ON [dbo].[PlateApplication]([PlateApplicationID],
[CustomerNumber])
go

CREATE STATISTICS [_dta_stat_2088224527_5_3] ON [dbo].[PlateApplication]([CustomerNumber],
[TransactionID])
go

CREATE STATISTICS [_dta_stat_2088224527_1_3_5] ON [dbo].[PlateApplication]([PlateApplicationID],
[TransactionID], [CustomerNumber])
go

CREATE STATISTICS [_dta_stat_632167285_2_4_5] ON [dbo].[VehicleCharacteristic]([VehicleNumber],
[EffectiveDate], [EndDate])
go

CREATE STATISTICS [_dta_stat_1209601385_10] ON [dbo].[Make]([Active])
go

CREATE STATISTICS [_dta_stat_1209601385_1_10] ON [dbo].[Make]([MakeID], [Active])
go

CREATE STATISTICS [_dta_stat_255886016_13_7] ON [dbo].[TitleBatchPrint]([TitleDocumentType],
[Printed])
go

CREATE STATISTICS [_dta_stat_255886016_7_4] ON [dbo].[TitleBatchPrint]([Printed],
[DeviceDocumentClassID])
go

CREATE STATISTICS [_dta_stat_255886016_8_7_3] ON [dbo].[TitleBatchPrint]([PrintedDate], [Printed],
[TitleNumber])
go

CREATE STATISTICS [_dta_stat_255886016_4_5_6_7] ON [dbo].[TitleBatchPrint]([DeviceDocumentClassID],
[ZipCode], [ZipCodeDeliveryPoint], [Printed])
go

CREATE STATISTICS [_dta_stat_255886016_9_8_7_3] ON [dbo].[TitleBatchPrint]([PrintImmediately],
[PrintedDate], [Printed], [TitleNumber])
go

CREATE STATISTICS [_dta_stat_255886016_4_2_7_5_6] ON [dbo].[TitleBatchPrint]([DeviceDocumentClassID],
[TitleID], [Printed], [ZipCode], [ZipCodeDeliveryPoint])
go

CREATE STATISTICS [_dta_stat_939644378_1_2] ON [dbo].[FeeAdjustment]([FeeAdjustmentID], [FeeID])
go

CREATE STATISTICS [_dta_stat_871504171_4_2_3] ON
[dbo].[TransactionSecurityInterest]([SecurityInterestAdded], [TransactionID],
[VehicleOwnershipVestedPartyID])
go

CREATE STATISTICS [_dta_stat_1206125375_9_1] ON [dbo].[Style]([Active], [StyleID])
go

CREATE STATISTICS [_dta_stat_79885389_2] ON [dbo].[PropertyTaxSegment]([PropertyTaxSegmentTypeID])
go
```

```
CREATE STATISTICS [_dta_stat_79885389_1_5] ON [dbo].[PropertyTaxSegment]([PropertyTaxSegmentID],
[PropertyTaxExemptionID])
go

CREATE STATISTICS [_dta_stat_79885389_1_2_5] ON [dbo].[PropertyTaxSegment]([PropertyTaxSegmentID],
[PropertyTaxSegmentTypeID], [PropertyTaxExemptionID])
go

CREATE STATISTICS [_dta_stat_1357793932_7_13] ON [dbo].[PermitCustomer]([CustomerNumber], [Active])
go

CREATE STATISTICS [_dta_stat_1357793932_1_13] ON [dbo].[PermitCustomer]([PermitCustomerID], [Active])
go

CREATE STATISTICS [_dta_stat_1357793932_2_7] ON [dbo].[PermitCustomer]([PermitID], [CustomerNumber])
go

CREATE STATISTICS [_dta_stat_1357793932_13_2_1] ON [dbo].[PermitCustomer]([Active], [PermitID],
[PermitCustomerID])
go

CREATE STATISTICS [_dta_stat_1889935792_1_5] ON [dbo].[TitleStatusHistory]([TitleStatusHistoryID],
[EndDate])
go

CREATE STATISTICS [_dta_stat_1889935792_2_5] ON [dbo].[TitleStatusHistory]([TitleID], [EndDate])
go

CREATE STATISTICS [_dta_stat_1889935792_2_1_5] ON [dbo].[TitleStatusHistory]([TitleID],
[TitleStatusHistoryID], [EndDate])
go

CREATE STATISTICS [_dta_stat_1226317466_3_2] ON [dbo].[PropertyTaxDetail]([RegistrationYear],
[PropertyTaxSegmentID])
go

CREATE STATISTICS [_dta_stat_1226317466_3_1] ON [dbo].[PropertyTaxDetail]([RegistrationYear],
[PropertyTaxDetailID])
go

CREATE STATISTICS [_dta_stat_1226317466_2_1_3] ON [dbo].[PropertyTaxDetail]([PropertyTaxSegmentID],
[PropertyTaxDetailID], [RegistrationYear])
go

CREATE STATISTICS [_dta_stat_415886586_1_2] ON [dbo].[TransactionSnapshot]([TransactionSnapshotID],
[TransactionID])
go
```