Microsoft

Automated Partition Management for Analysis Services Tabular Models

Microsoft BI Technical Article

Writer: Christian Wade, Senior Program Manager, Microsoft Corp.

Contributor: Owen Duncan, Senior Content Developer, Microsoft Corp.

Published: December 2016

Applies to: Microsoft SQL Server 2016 Analysis Services, Microsoft Azure Analysis Services

Summary: This whitepaper and associated samples describe partition management automation by using the Tabular Object Model (TOM).

Contents

Introduction

Analysis Services tabular models can store data in a highly-compressed, in-memory cache for optimized query performance. This provides fast user interactivity over large data sets.

Large datasets normally require table partitioning to accelerate and optimize the data-load process. Partitioning enables incremental loads, increases parallelization, and reduces memory consumption. The Tabular Object Model (TOM) serves as an API to create and manage partitions. TOM was released with SQL Server 2016 and is discussed in more detail here. Model Compatibility Level 1200 or above is required.

This document describes how to use the AsPartitionProcessing TOM code sample for automated partition management with minimal code changes.

The sample,

- Is intended to be generic and configuration driven.
- Works for both Azure Analysis Services and SQL Server Analysis Services tabular models.
- Can be leveraged in many ways including from an SSIS script task, Azure Functions and others.

Note: Loading data into the in-memory cache is often referred to as *processing*. This terminology is used by this document.

Partitioning Strategy & Assumptions

Rolling-window pattern

AsPartitionProcessing follows the rolling-window pattern, which is common in traditional Analysis Services implementations. The data is kept within a predefined date range and incremented as necessary. This maintains memory usage within a predictable range over time.

Partition granularity

Yearly, monthly and daily partition granularities can be configured. Choice of granularity is influenced by various factors including how much data is required to be incrementally refreshed and how much processing time is acceptable. For example, if only the last 3 days need to be refreshed daily, it may be beneficial to use daily granularity.

Mixed granularity for a table can also be configured for scenarios such as near-real time refresh at low grain coupled with historical, static partitions at higher granularity. This results in fewer partitions for a table, but also increases management overhead to ensure partition ranges are defined correctly. Unless there are hundreds of partitions or more, there is normally no significant query-performance penalty resulting from keeping the partitions at the lowest grain.

Parallelization

Initial setup processing is sequential. Incremental processing can be performed in parallel.

Initial setup will create and process the partitions for the first time based on the configuration. This is performed one partition at a time to avoid running out of memory (data is not fully compressed during processing). For a large data set, the initial load may typically take a few hours depending on factors such as the query performance of the source system.

Incremental processing can be configured to execute as a fully parallelized operation for all tables within a model. It can also be configured to process different tables one at a time. The reason to consider processing one table at a time is again to work within memory constraints. When processing multiple partitions within a single table, they are always done in parallel.

Online & offline processing

Incremental processing can be performed as an online operation, or offline for less memory usage; it is configuration driven. Online incremental processing requires a copy of the data to be maintained in memory for queries until the new data is ready, and then switches to the new data. When processing multiple tables, keeping the model online can be less efficient because it often requires recalculation of the same calculated columns, relationships and indexes multiple times. Offline processing has the benefit of performing this recalculation just once at the end of the processing window.

Non-partitioned table processing

The sample can be configured to process non-partitioned tables in addition to partitioned ones. This avoids having to set up a separate process to refresh non-partitioned tables.

Table omission

It is possible to configure that some tables in the model are not refreshed at all during normal incremental processing. Tables that may not require frequent processing often include the date dimension, categorical dimensions, and facts that may be defined annually such as budget.

Configuration & logging database

Traditional Analysis Services implementations that require partitioning often use a configuration and logging database. AsPartitionProcessing is meant to work in this way, although this is optional. It can be set up to log messages to other targets. This enables easy partition configuration, and diagnosis of issues resulting from automated processing operations.

Date key format

Date keys in source table are assumed to be integers formatted as yyyymmdd, which is common for data warehouses and marts. If this is not available, it should be possible to derive such a column in a database view.

Getting Started

Requirements

Before you get started, you'll need these tools:

SQL Server 2016 with latest service pack - Install the database engine and SSAS in tabular mode. You can download and install the free SQL Server 2016 Developer Edition here.

SQL Server Data Tools – Download and install the latest version here.

SQL Server Management Studio - Download and install the latest version here.

Visual Studio 2015 – Download and install the free Community Edition here.

AsPartitionProcessing solution

Get the AsPartitionProcessing solution here.

1. Open the solution in Visual Studio and build the project. The hint path for the client library DLLs is the following (assuming installation is on C:\ drive):
   C:\Program Files (x86)\Microsoft SQL Server\130\SDK\Assemblies
2. Ensure AsPartitionProcessing.SampleClient is set as the startup project.

AdventureWorks

The quickest way to understand the code sample is to run it on the AdventureWorksDW sample database. The backup file, AdventureWorksDW.bak, is included in the solution.

The tabular project, AsPartitionProcessing.AdventureWorks**,** is also provided in the solution. It should be used instead of the version from CodePlex because partitioning has been removed from the Internet Sales and Reseller Sales tables. Instead, these tables each have a single partition with the same name as the table, which is the default when you create a new table in SSDT. This partition acts as the *template partition* used by the AsPartitionProcessing sample.

Deploy and process the AdventureWorks tabular model.

SampleClient

AsPartitionProcessing.SampleClient is a console application with a reference to the AsPartitionProcessing class library. It can easily be converted to work for customer projects. Alternatively, it provides sample client code to execute from an SSIS package, Azure Function, or other mechanism.

In Program.cs, note the ExecutionMode constant.

```
const SampleExecutionMode ExecutionMode = SampleExecutionMode.InitializeInline;
```

This means the InitializeAdventureWorksInline method will be executed to initialize parameters.

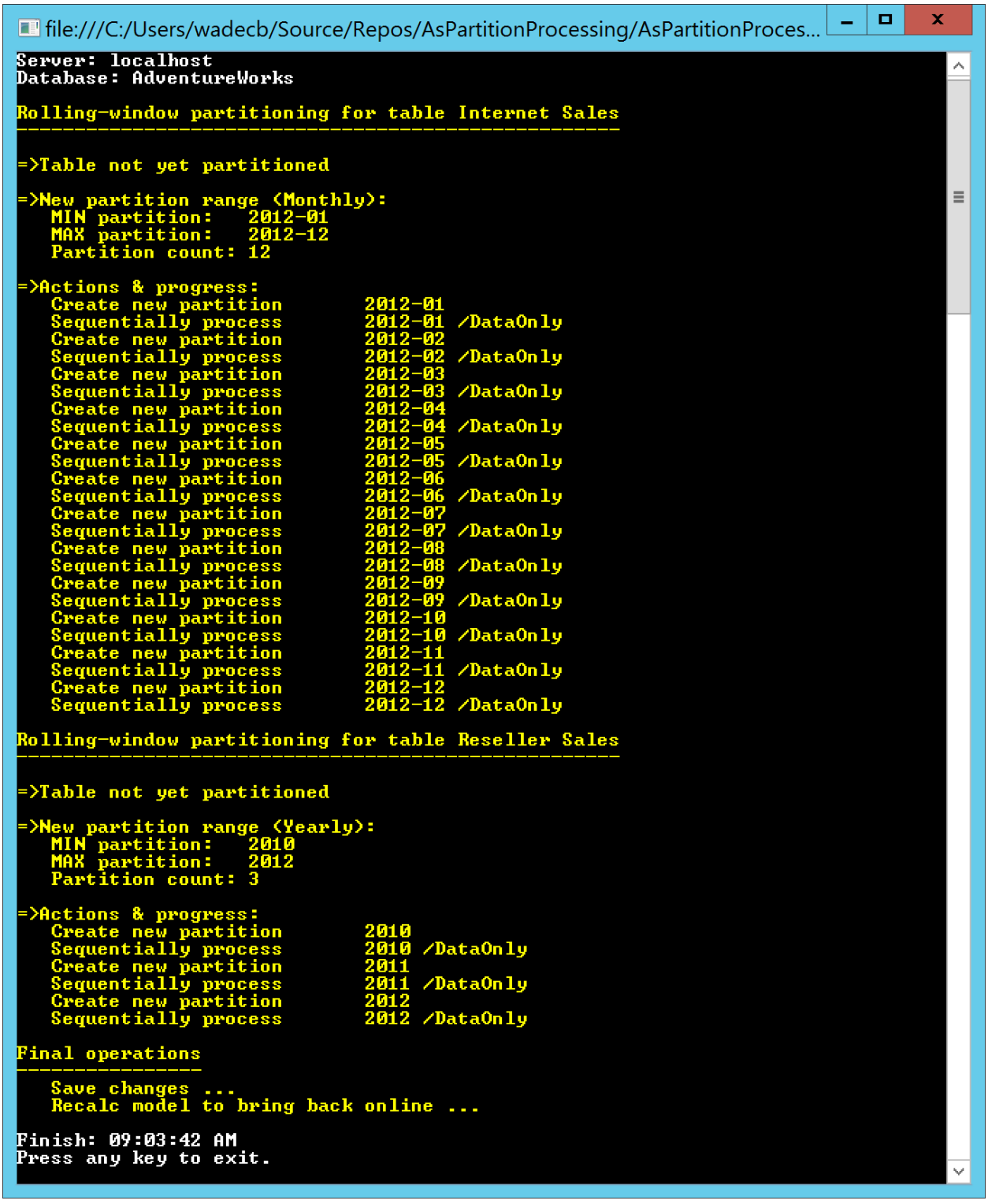
```
ModelConfiguration partitionedModel = new ModelConfiguration(
    modelConfigurationID: 1,
    analysisServicesServer: "localhost",
```

```
    analysisServicesDatabase: "AdventureWorks",
    initialSetUp: true,
    incrementalOnline: true,
    incrementalParallelTables: true,
    integratedAuth: true,
    userName: "",
    password: "",
    tableConfigurations:
    new List<TableConfiguration>
    {
        new TableConfiguration(
            tableConfigurationID: 1,
            analysisServicesTable: "Internet Sales",
            partitioningConfigurations:
            new List<PartitioningConfiguration>
            {
                new PartitioningConfiguration(
                    partitioningConfigurationID: 1,
                    granularity: Granularity.Monthly,
                    numberOfPartitionsFull: 12,
                    numberOfPartitionsForIncrementalProcess: 3,
                    maxDate: Convert.ToDateTime("2012-12-01"),
                    sourceTableName: "[dbo].[FactInternetSales]",
                    sourcePartitionColumn: "OrderDateKey"
                )
            }
        ),
        new TableConfiguration(
            tableConfigurationID: 2,
            analysisServicesTable: "Reseller Sales",
            partitioningConfigurations:
            new List<PartitioningConfiguration>
            {
                new PartitioningConfiguration(
                    partitioningConfigurationID: 2,
                    granularity: Granularity.Yearly,
                    numberOfPartitionsFull: 3,
                    numberOfPartitionsForIncrementalProcess: 1,
                    maxDate: Convert.ToDateTime("2012-12-01"),
                    sourceTableName: "[dbo].[FactResellerSales]",
                    sourcePartitionColumn: "OrderDateKey"
                )
            }
        )
    }
);
```

Place a breakpoint at the Main method and step through the code in the AsPartitionProcessing.SampleClient project to understand how to interact with the methods exposed by the class library. The PerformProcessing method is the key method.

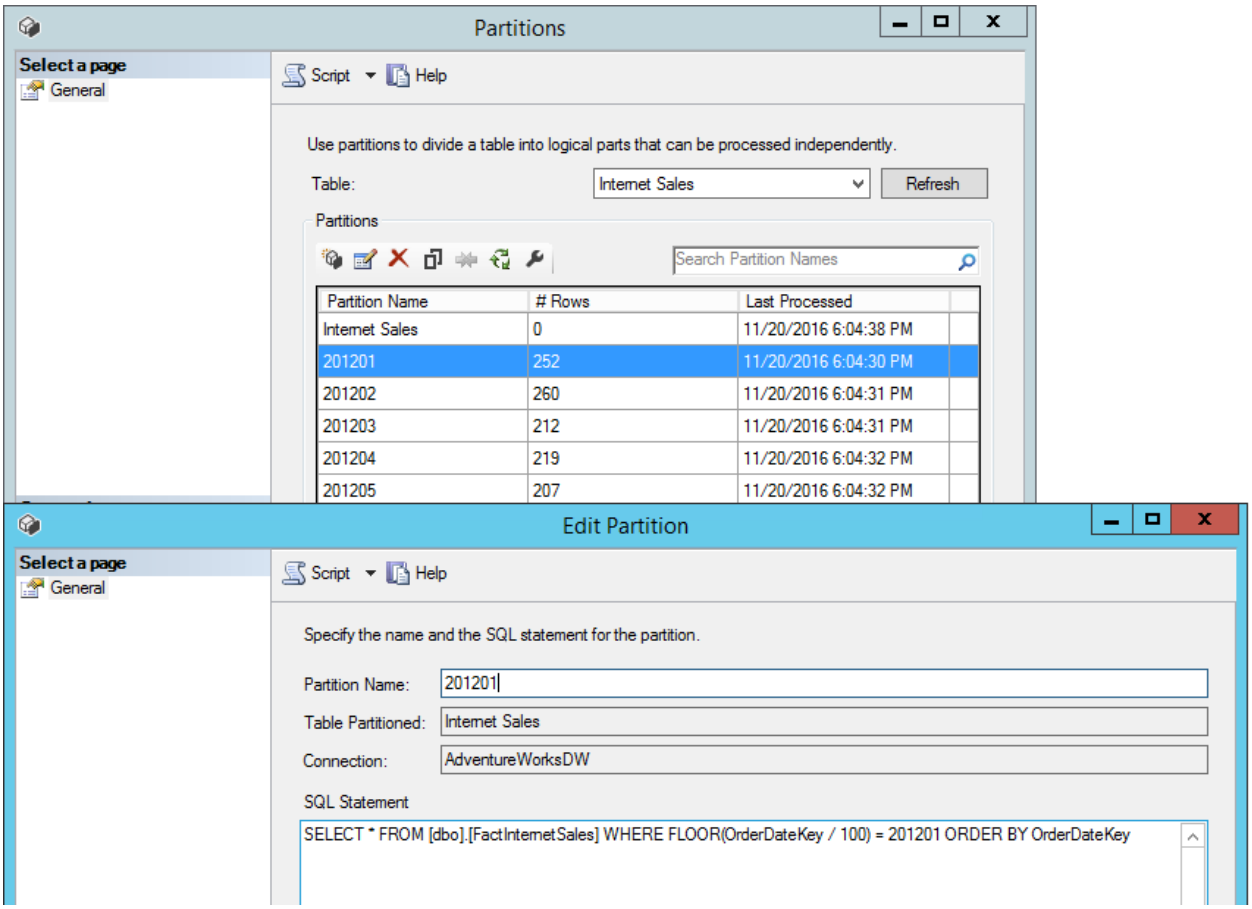
```
PartitionProcessor.PerformProcessing(modelConfig, LogMessage);
```

The console output should be displayed like this:



Use SSMS to inspect the partitions created. Partition source queries take the simple form:

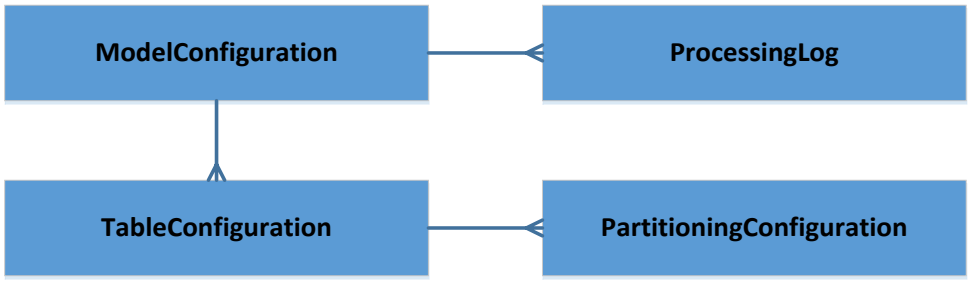
SELECT * FROM <*source table*> WHERE <*partition filter*>



Configuration & Logging Database

Typically, partitioning configuration and logging is done using a database. AsPartitionProcessing contains the CreateDatabaseObjects.sql script to create the necessary tables, and contains the methods for reading and writing to the database.

Data model



ModelConfiguration

Configuration information for an AS tabular model:

Column	Description

ModelConfigurationID	Primary key.
AnalysisServicesServer	Name of the Analysis Services instance. Can be SSAS or an Azure AS URL.
AnalysisServicesDatabase	Name of the Analysis Services database.
InitialSetUp	True for initial set up to create partitions and process them sequentially. False for incremental processing. See Partitioning Strategy & Assumptions section above for more information.
IncrementalOnline	When initialSetUp=false, determines if processing is performed as an online operation, which can require more memory, but allows users to query the model during processing. True to keep the model online (process Full). See Partitioning Strategy & Assumptions section above for more information.
IncrementalParallelTables	When initialSetUp=false, determines if separate tables are processed in parallel. Note: partitions within a table are always processed in parallel. True to process tables in parallel. See Partitioning Strategy & Assumptions section above for more information.
IntegratedAuth	Should always be set to true for SSAS implementations that will run under the current process account. For Azure AS, normally set to false.
UserName	Only applies when integratedAuth=false. Can be used for Azure AD UPNs to connect to Azure AS.
Password	Only applies when integratedAuth=false. Can be used for Azure AD UPNs to connect to Azure AS.

TableConfiguration

Configuration information for a table within an AS tabular model:

Column	Description
TableConfigurationID	Primary key.
ModelConfigurationID	Foreign key to ModelConfiguration table.
AnalysisServicesTable	Name of the partitioned table in the tabular model.
DoNotProcess	Set to true to exclude the table from processing. This can be used to dynamically include/exclude tables. For example, near-realtime processing during the day requires only a few tables to be processed; overnight processing may process all tables.

PartitioningConfiguration

Configuration information for partitioning of a table within an AS tabular model.:

Column	Description
PartitioningConfigurationID	Primary key.
TableConfigurationID	Foreign key to TableConfiguration table.
Granularity	Partition granularity, which can be Yearly, Monthly or Daily. Daily = 0, Monthly = 1, Yearly = 2
NumberOfPartitionsFull	Count of all partitions in the rolling window. For example, a rolling window of 10 years partitioned by month would require 120 partitions.
NumberOfPartitionsForIncrementalProcess	Count of *hot partitions* where the data can change. For example, it may be necessary to refresh the most recent 3 months of data every day. This only applies to the most recent partitions.
MaxDate	The maximum date that needs to be accounted for in the partitioning configuration.
SourceTableName	Name of the source table in the relational database.
SourcePartitionColumn	Name of the source column from the table in the relational database.

ProcessingLog

Log of partitioning execution:

Column	Description
ProcessingLogID	Primary key.
ModelConfigurationID	Foreign key to ModelConfiguration table.
ExecutionID	GUID generated for the execution run.
LogDateTime	Date and time the message was logged.
Message	The log message.

Sample Configuration

The SampleConfiguration.sql script initializes the configuration for AdventureWorks. The script can be modified for use in customer implementations. Execute the script to initialize the database.

```
INSERT INTO [dbo].[ModelConfiguration]
VALUES(
     1                          --[ModelConfigurationID]
    ,'localhost'                --[AnalysisServicesServer]
    ,'AdventureWorks'           --[AnalysisServicesDatabase]
    ,1                          --[InitialSetUp]
    ,1                          --[IncrementalOnline]
    ,1                          --[IncrementalParallelTables]
    ,1                          --[IntegratedAuth]
);

INSERT INTO [dbo].[TableConfiguration]
VALUES(
```

```
     1                          --[TableConfigurationID]
    ,1                          --[ModelConfigurationID]
    ,'Internet Sales'           --[AnalysisServicesTable]
    ,0                          --[DoNotProcess]
),
(
     2                          --[TableConfigurationID]
    ,1                          --[ModelConfigurationID]
    ,'Reseller Sales'           --[AnalysisServicesTable]
    ,0                          --[DoNotProcess]
);

INSERT INTO [dbo].[PartitioningConfiguration]
VALUES(
     1                          --[PartitioningConfigurationID]
    ,1                          --[TableConfigurationID]
    ,1                          --[Granularity]   1=Monthly
    ,12                         --[NumberOfPartitionsFull]
    ,3                          --[NumberOfPartitionsForIncrementalProcess]
    ,'2012-12-01'               --[MaxDate]
    ,'[dbo].[FactInternetSales]'--[SourceTableName]
    ,'OrderDateKey'             --[SourcePartitionColumn]
),
(
     2                          --[PartitioningConfigurationID]
    ,2                          --[TableConfigurationID]
    ,2                          --[Granularity]   2=Yearly
    ,3                          --[NumberOfPartitionsFull]
    ,1                          --[NumberOfPartitionsForIncrementalProcess]
    ,'2012-12-01'               --[MaxDate]
    ,'[dbo].[FactResellerSales]'--[SourceTableName]
    ,'OrderDateKey'             --[SourcePartitionColumn]
);
```

Database connection info

Connection information to the configuration and logging database can be set in App.config in the userSettings section.

```
<userSettings>
    <AsPartitionProcessing.SampleClient.Settings>
        <setting name="ConfigServer" serializeAs="String">
            <value>localhost</value>
        </setting>
        <setting name="ConfigDatabase" serializeAs="String">
            <value>AsPartitionProcessing</value>
        </setting>
        <setting name="ConfigDatabaseIntegratedAuth" serializeAs="String">
            <value>True</value>
        </setting>
    </AsPartitionProcessing.SampleClient.Settings>
</userSettings>
```

Test Different Configurations

In this section, we will update the configuration, execute the sample, and view the log messages.

In Program.cs, change the ExecutionMode constant to be assigned InitializeFromDatabase.

```
const SampleExecutionMode ExecutionMode = SampleExecutionMode.InitializeFromDatabase;
```

Execute the SampleClient application, and the log query. If the SampleClient application was previously run from the Getting Started section, messages will be shown saying the partitions already exist and are processed.

Latest execution log query

In addition to the console output, the following query on the configuration and logging database shows the execution results. This can be used to test the different configurations below.

```
SELECT [Message]
FROM [dbo].[ProcessingLog]
WHERE ExecutionID =
(    SELECT MAX([ExecutionID]) FROM [dbo].[ProcessingLog]
     WHERE [LogDateTime] = (SELECT MAX([LogDateTime]) FROM [dbo].[ProcessingLog])
)
ORDER BY [LogDateTime]
```

Incremental mode

Run the following UPDATE statement to switch to incremental mode:

```
UPDATE [dbo].[ModelConfiguration] SET [InitialSetUp] = 0
```

Execute SampleClient application, and the log query. The following results should be shown. Only the specified number of most recent partitions are processed as an online operation.

```
Start: 12:41:02 PM
Server: localhost
Database: AdventureWorks

Rolling-window partitioning for table Internet Sales
----------------------------------------------------

=>Current partition range (Monthly):
   MIN partition:   2012-01
   MAX partition:   2012-12
   Partition count: 12

=>New partition range (Monthly):
   MIN partition:   2012-01
   MAX partition:   2012-12
   Partition count: 12

=>Actions & progress:
   Parallel process partition 2012-10 /Full
   Parallel process partition 2012-11 /Full
   Parallel process partition 2012-12 /Full
```

```
Rolling-window partitioning for table Reseller Sales
----------------------------------------------------

=>Current partition range (Yearly):
   MIN partition:   2010
   MAX partition:   2012
   Partition count: 3

=>New partition range (Yearly):
   MIN partition:   2010
   MAX partition:   2012
   Partition count: 3

=>Actions & progress:
   Parallel process partition 2012 /Full

Final operations
----------------
   Save changes ...

Finish: 12:41:07 PM
```

Increment partition range

Run the following UPDATE statement to increment the partition range by one period.

```
UPDATE [dbo].[PartitioningConfiguration] SET [MaxDate] = '2013-01-01'
```

Execute SampleClient application, and the log query. The following results should be shown. The oldest partition is removed from both tables, a new one is added and the most recent partitions are processed.

```
Start: 12:47:26 PM
Server: localhost
Database: AdventureWorks

Rolling-window partitioning for table Internet Sales
----------------------------------------------------

=>Current partition range (Monthly):
   MIN partition:   2012-01
   MAX partition:   2012-12
   Partition count: 12

=>New partition range (Monthly):
   MIN partition:   2012-02
   MAX partition:   2013-01
   Partition count: 12

=>Actions & progress:
   Remove old partition       2012-01
   Parallel process partition 2012-11 /Full
   Parallel process partition 2012-12 /Full
   Create new partition       2013-01
   Parallel process partition 2013-01 /Full

Rolling-window partitioning for table Reseller Sales
----------------------------------------------------
```

```
=>Current partition range (Yearly):
   MIN partition:   2010
   MAX partition:   2012
   Partition count: 3

=>New partition range (Yearly):
   MIN partition:   2011
   MAX partition:   2013
   Partition count: 3

=>Actions & progress:
   Remove old partition       2010
   Create new partition       2013
   Parallel process partition 2013 /Full

Final operations
----------------
   Save changes ...

Finish: 12:47:34 PM
```

Offline processing

Run the following UPDATE statement to perform offline processing to potentially use less memory during processing.

```
UPDATE [dbo].[ModelConfiguration] SET [IncrementalOnline] = 0
```

Execute SampleClient application, and the log query. The following results should be shown. The partitions are processed using RefreshType of DataOnly, and a Recalc operation is performed to bring the model back online.

```
Start: 12:53:19 PM
Server: localhost
Database: AdventureWorks

Rolling-window partitioning for table Internet Sales
----------------------------------------------------

=>Current partition range (Monthly):
   MIN partition:   2012-02
   MAX partition:   2013-01
   Partition count: 12

=>New partition range (Monthly):
   MIN partition:   2012-02
   MAX partition:   2013-01
   Partition count: 12

=>Actions & progress:
   Parallel process partition 2012-11 /DataOnly
   Parallel process partition 2012-12 /DataOnly
   Parallel process partition 2013-01 /DataOnly

Rolling-window partitioning for table Reseller Sales
```

```
------------------------------------------------

=>Current partition range (Yearly):
   MIN partition:   2011
   MAX partition:   2013
   Partition count: 3

=>New partition range (Yearly):
   MIN partition:   2011
   MAX partition:   2013
   Partition count: 3

=>Actions & progress:
   Parallel process partition 2013 /DataOnly

Final operations
----------------
   Save changes ...
   Recalc model to bring back online ...

Finish: 12:53:27 PM
```

Sequential table processing

Run the following UPDATE statement to process tables sequentially for less memory usage. This setting will take longer to process the model.

```
UPDATE [dbo].[ModelConfiguration] SET [IncrementalParallelTables] = 0
```

Execute SampleClient application, and the log query. The following results should be shown. Changes are saved for each table individually before moving onto the next table.

```
Start: 01:04:25 PM
Server: localhost
Database: AdventureWorks

Rolling-window partitioning for table Internet Sales
----------------------------------------------------

=>Current partition range (Monthly):
   MIN partition:   2012-02
   MAX partition:   2013-01
   Partition count: 12

=>New partition range (Monthly):
   MIN partition:   2012-02
   MAX partition:   2013-01
   Partition count: 12

=>Actions & progress:
   Parallel process partition 2012-11 /DataOnly
   Parallel process partition 2012-12 /DataOnly
   Parallel process partition 2013-01 /DataOnly
   Save changes for table Internet Sales ...

Rolling-window partitioning for table Reseller Sales
----------------------------------------------------
```

```
=>Current partition range (Yearly):
   MIN partition:   2011
   MAX partition:   2013
   Partition count: 3

=>New partition range (Yearly):
   MIN partition:   2011
   MAX partition:   2013
   Partition count: 3

=>Actions & progress:
   Parallel process partition 2013 /DataOnly
   Save changes for table Reseller Sales ...

Final operations
----------------
   Recalc model to bring back online ...

Finish: 01:04:33 PM
```

Non-partitioned table processing & table omission

Run the following INSERT statement to create table configurations for the Customer, Product and Sales Quota tables.

```
INSERT INTO [dbo].[TableConfiguration]
VALUES(
     3                          --[TableConfigurationID]
    ,1                          --[ModelConfigurationID]
    ,'Customer'                 --[AnalysisServicesTable]
    ,0                          --[DoNotProcess]
),
(
     4                          --[TableConfigurationID]
    ,1                          --[ModelConfigurationID]
    ,'Product'                  --[AnalysisServicesTable]
    ,0                          --[DoNotProcess]
),
(
     5                          --[TableConfigurationID]
    ,1                          --[ModelConfigurationID]
    ,'Sales Quota'              --[AnalysisServicesTable]
    ,1                          --[DoNotProcess]
);
```

Sales Quota has DoNotProcess equal to 1, so it will be excluded from processing. This flag can be used to dynamically include and exclude tables. For example, certain tables can be processed during the day for near-real time requirements and other tables processed overnight.

The Customer and Product tables will be processed. They do not have entries in the PartitioningConfiguration table, so they will be treated as non-partitioned tables and processed at the table level.

Other tables in the model that do not have table configuration entries are omitted from processing. Some tables may have no ongoing processing requirements. For example, the date-dimension table and categorical dimension tables typically have no need to be processed daily, so they can be left out altogether.

Execute SampleClient application, and the log query. The following results should be shown. The Customer and Product non-partitioned tables are processed.

```
Start: 08:55:05 PM
Server: localhost
Database: AdventureWorks

Rolling-window partitioning for table Internet Sales
----------------------------------------------------

=>Current partition range (Monthly):
   MIN partition:   2012-02
   MAX partition:   2013-01
   Partition count: 12

=>New partition range (Monthly):
   MIN partition:   2012-02
   MAX partition:   2013-01
   Partition count: 12

=>Actions & progress:
   Parallel process partition 2012-11 /DataOnly
   Parallel process partition 2012-12 /DataOnly
   Parallel process partition 2013-01 /DataOnly
   Save changes for table Internet Sales ...

Rolling-window partitioning for table Reseller Sales
----------------------------------------------------

=>Current partition range (Yearly):
   MIN partition:   2011
   MAX partition:   2013
   Partition count: 3

=>New partition range (Yearly):
   MIN partition:   2011
   MAX partition:   2013
   Partition count: 3

=>Actions & progress:
   Parallel process partition 2013 /DataOnly
   Save changes for table Reseller Sales ...

Non-partitioned processing for table Customer
---------------------------------------------
   Process table Customer /DataOnly
   Save changes for table Customer ...

Non-partitioned processing for table Product
--------------------------------------------
   Process table Product /DataOnly
   Save changes for table Product ...
```

```
Final operations
---------------
   Recalc model to bring back online ...

Finish: 08:55:17 PM
```

Merging partitions

Merging of partitions may be useful in mixed-granularity scenarios. For example, merging historical days into a month, or merging historical months into a year. Care must be taken when merging partitions to ensure it is done correctly. Once merged, it is not possible to unmerge.

In Program.cs, change the ExecutionMode constant to be assigned MergePartitions.

```
const SampleExecutionMode ExecutionMode = SampleExecutionMode.MergePartitions;
```

Run the following UPDATE statement to ensure the daily configuration range does not overlap with the new partition at year granularity.

```
UPDATE dbo.[PartitioningConfiguration] SET
  NumberOfPartitionsFull = 1,
  NumberOfPartitionsForIncrementalProcess = 1
WHERE PartitioningConfigurationID = 1 --Internet Sales
```

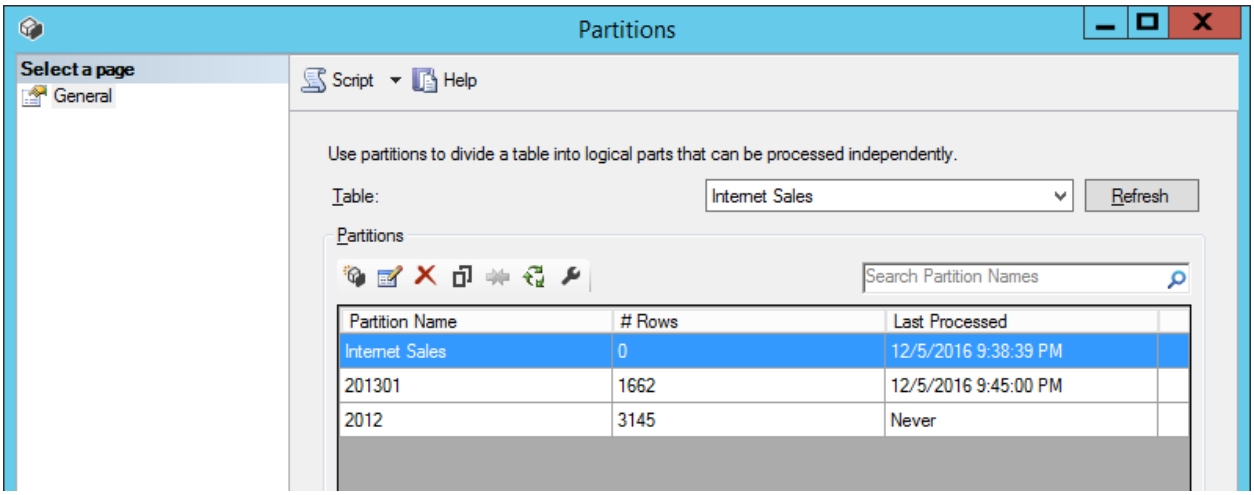
Execute SampleClient application, and the log query. The following results should be shown. The months in 2012 are merged into the year 2012.

```
Merge partitions into 2012 for table Internet Sales
---------------------------------------------------

=>Actions & progress:
   Create new merged partition 2012 for table Internet Sales
   Partition 201202 to be merged into 2012
   Partition 201203 to be merged into 2012
   Partition 201204 to be merged into 2012
   Partition 201205 to be merged into 2012
   Partition 201206 to be merged into 2012
   Partition 201207 to be merged into 2012
   Partition 201208 to be merged into 2012
   Partition 201209 to be merged into 2012
   Partition 201210 to be merged into 2012
   Partition 201211 to be merged into 2012
   Partition 201212 to be merged into 2012
   Save changes for table Internet Sales ...

Finish: 10:03:38 PM
```

Inspect the new partition structure in SSMS.



Mixed granularity configurations

For mixed granularity scenarios, it may be necessary to set up multiple configurations. This allows automated removal of old partitions – at different granularities – that fall out of range. Care must be taken to ensure partitioning configurations are set correctly to avoid integrity issues.

In Program.cs, change the ExecutionMode constant back to InitializeFromDatabase.

```
const SampleExecutionMode ExecutionMode = SampleExecutionMode.InitializeFromDatabase;
```

Run the following UPDATE and INSERT statements to configure mixed granularity for the Reseller Sales table. The model configuration is set back to initial set-up mode to create the new partition ranges.

```
UPDATE [dbo].[ModelConfiguration] SET [InitialSetUp] = 1

INSERT INTO [dbo].[PartitioningConfiguration]
VALUES(
     3                          --[PartitioningConfigurationID]
    ,2                          --[TableConfigurationID]
    ,1                          --[Granularity]   1=Monthly
    ,3                          --[NumberOfPartitionsFull]
    ,0                          --[NumberOfPartitionsForIncrementalProcess]
    ,'2014-03-01'               --[MaxDate]
    ,'[dbo].[FactResellerSales]'--[SourceTableName]
    ,'OrderDateKey'             --[SourcePartitionColumn]
),
(
     4                          --[PartitioningConfigurationID]
    ,2                          --[TableConfigurationID]
    ,0                          --[Granularity]   2=Daily
    ,3                          --[NumberOfPartitionsFull]
    ,1                          --[NumberOfPartitionsForIncrementalProcess]
    ,'2014-04-03'               --[MaxDate]
    ,'[dbo].[FactResellerSales]'--[SourceTableName]
    ,'OrderDateKey'             --[SourcePartitionColumn]
);
```

Execute SampleClient application, and the log query. The following results should be shown. Reseller Sales has 3 partitions at the year level, 3 at the month level, and 3 at the day level.

```
Start: 09:05:06 PM
Server: localhost
Database: AdventureWorks

Rolling-window partitioning for table Internet Sales
----------------------------------------------------

=>Current partition range (Monthly):
   MIN partition:   2013-01
   MAX partition:   2013-01
   Partition count: 1

=>New partition range (Monthly):
   MIN partition:   2013-01
   MAX partition:   2013-01
   Partition count: 1

=>Actions & progress:
   Partition 2013-01 already exists and is processed
   Save changes for table Internet Sales ...

Rolling-window partitioning for table Reseller Sales
----------------------------------------------------

=>Current partition range (Yearly):
   MIN partition:   2011
   MAX partition:   2013
   Partition count: 3

=>New partition range (Yearly):
   MIN partition:   2011
   MAX partition:   2013
   Partition count: 3

=>Actions & progress:
   Partition 2011 already exists and is processed
   Partition 2012 already exists and is processed
   Partition 2013 already exists and is processed
   Save changes for table Reseller Sales ...

Rolling-window partitioning for table Reseller Sales
----------------------------------------------------

=>Table not yet partitioned

=>New partition range (Monthly):
   MIN partition:   2014-01
   MAX partition:   2014-03
   Partition count: 3

=>Actions & progress:
   Create new partition       2014-01
   Sequentially process       2014-01 /DataOnly
   Create new partition       2014-02
```

```
   Sequentially process       2014-02 /DataOnly
   Create new partition       2014-03
   Sequentially process       2014-03 /DataOnly
   Save changes for table Reseller Sales ...

Rolling-window partitioning for table Reseller Sales
----------------------------------------------------

=>Table not yet partitioned

=>New partition range (Daily):
   MIN partition:   2014-04-01
   MAX partition:   2014-04-03
   Partition count: 3

=>Actions & progress:
   Create new partition       2014-04-01
   Sequentially process       2014-04-01 /DataOnly
   Create new partition       2014-04-02
   Sequentially process       2014-04-02 /DataOnly
   Create new partition       2014-04-03
   Sequentially process       2014-04-03 /DataOnly
   Save changes for table Reseller Sales ...

Non-partitioned processing for table Customer
---------------------------------------------
   Process table Customer /DataOnly
   Save changes for table Customer ...

Non-partitioned processing for table Product
--------------------------------------------
   Process table Product /DataOnly
   Save changes for table Product ...

Final operations
----------------
   Recalc model to bring back online ...

Finish: 09:05:22 PM
```

Validation of date ranges for mixed granularity

Todo

Error if overlapping ranges.

Error if more than 1 granuarity defined per partitioning configuration.

Error if for example month upper boundary is Mar-2016, but there are partitions at month level greater than that (Apri-2016), or will mess up mixed granularity

Other Options & Considerations

Custom logging

The LogMessage method in Program.cs is passed as a delegate into the PerformProcessing method, so it can easily be changed for custom logging requirements.

```
private static void LogMessage(string message, ModelConfiguration partitionedModel)
{
    //Can provide custom logging code here
    ...
```

Fragmentation

Partitioned tables may suffer from fragmentation over time. When a partition is removed from a table, the table dictionary entries are retained despite having no rows of data. Defragmentation removes the unused dictionary entries. It is not necessary to perform defragmentation on non-partitioned tables because they are processed at the table level. Defragmentation of large tables can be an expensive, sometimes time-consuming, operation.

Dictionary and table size can be monitored using community tools such as VertiPaq Analyzer and SSAS Memory Usage Report.

The code sample provides the following methods for defragmentation.

Todo

Locking

Todo

Model deployment

When deploying new versions of partitioned tabular models that already exist on the target environment, it is necessary to be aware of the partitioning process already in-place. As shown by this code sample, partitions are normally created and managed by a separate process. This means the version of the tabular model from source control does not contain the partitions. A simple deployment process such as right-click, Deploy from SSDT will lose the partitions and all the data within them. Two deployment tools that support retaining partitions are BISM Normalizer and the Analysis Services Deployment Wizard. Both these tools support command-line execution for automated deployment. Detailed discussion on this topic including the pros and cons of these tools is outside the scope of this document.

AsPerfMon

Another code sample that may be useful in conjunction with AsPartitionProcessing is the AsPerfMon tool, which is available here.

AsPerfMon can be used to check real-time memory usage during processing. It splits memory usage by database, which is informative when multiple databases share the same server.

This is especially useful for Azure AS since you can't use Task Manager or create Performance Monitor counters. Similar functionality is provided by the Metrics section in the control blade for an Azure AS server in the Azure Portal. By using Metrics, you can check usage for the past day or week. AsPerfMon is for real-time monitoring during processing.

AsPerfMon works by polling the DISCOVER_OBJECT_MEMORY_USAGE Data Management View.