Paper SAS255-2014

Managing Large Data with SAS® Dynamic Cluster Table Transactions

Guy Simpson, SAS Institute Inc., Cary, NC

ABSTRACT

Today's business needs require 24/7 access to your data in order to perform complex queries to your analytical store. In addition, you might need to periodically update your analytical store to append new data, delete old data, or modify some existing data. Historically, the size of your analytical tables or the window in which the table must be updated can cause unacceptable downtime for queries. This paper describes how you can use new SAS® Scalable Performance Data Server 5.1 cluster table features to simulate transaction isolation in order to modify large sections of your cluster table. These features are optimized for extremely fast operation and can be done without affecting any on-going queries. This provides both the continuous query access and periodic update requirements for your analytical store that your business model requires.

INTRODUCTION

The SAS Scalable Performance Data Server is designed to meet the storage and performance needs for processing large amounts of SAS data. As the size of data grows, the demands for processing the data quickly increase, and the storage architecture must change to keep pace with business needs.

Hundreds of sites worldwide use the SAS Scalable Performance Data Server, which hosts data warehouses in excess of 30 terabytes. The SAS Scalable Performance Data Server provides high-performance data storage for customers from industries such as banking, credit card, insurance, telecommunications, health care, pharmaceutical, transportation, and brokerage and government agencies.

Customers have been utilizing dynamic cluster tables, introduced in the SAS Scalable Performance Data Server 4.3, as a means to effectively manage their data. A cluster table is a collection of regular, partitioned, SPD Server tables, which are presented to the end-user application as a single table through a metadata layer acting similar to a view. The SPD cluster itself is a read-only structure. This means that inserts, updates or deletes, which can be done against a regular table, are not possible with the cluster unless the cluster is broken prior to the operation, prohibiting queries on the cluster while the maintenance is being done.

Today’s business models now typically require 24/7 access to their store to perform complex queries, along with the need to periodically update their store to append new data, delete old date, or modify existing data. These requirements cannot allow even the smallest downtime to perform maintenance on an SPD Cluster. To satisfy the 24/7 access requirement, The SAS Scalable Performance Data Server 5.1 has added new Cluster Table features that simulate transaction isolation to modify large sections of your cluster table. These new features are optimized for extremely fast operation and can be done without affecting on-going queries. This provides both the continuous query access and periodic update requirements that today’s business models require.

CLUSTER TABLE OVERVIEW

The SAS Scalable Performance Data Server provides a virtual table structure, which is called a *clustered data table* (Figure 1). A *cluster* contains a number of members, where each member is a SAS Scalable Performance Data Server table with the same attributes. The clustered data table uses a layer of metadata to manage the members.

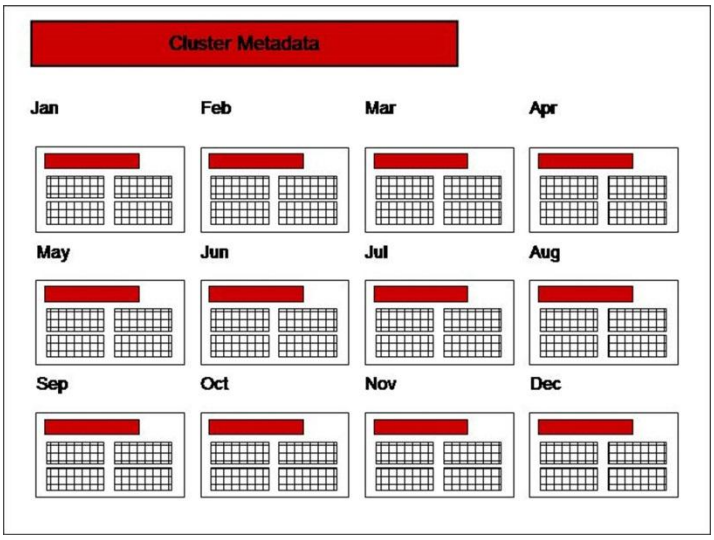

Figure 1. Clustered Data Table

The virtual table structure in the SAS Scalable Performance Data Server offers flexible storage that enables users to organize tables based on values (including SAS date, time, or datetime values) that are contained in numeric columns. Introduced in the SAS Scalable Performance Data Server 4.3, this type of organization is called a *dynamic cluster table*.

Commands for Creating and Undoing a Cluster

Clusters are simple structures, and creating or undoing/deleting a cluster takes only seconds. The three most basic commands are CLUSTER CREATE, CLUSTER UNDO, and CLUSTER DELETE. You execute all of these commands within PROC SPDO.

The basic CLUSTER CREATE command requires four options:

1. The name of the cluster table (*cluster-table-name*) that will be created.
2. A list of the SAS Scalable Performance Data Server tables that will be included in the cluster (using the MEM= option).
3. The maximum number of member tables (using the MAXSLOT= option), that the cluster will have. If not specified the cluster table supports dynamic growth, expanding as necessary to support member tables.
4. An optional parameter which allows the cluster metadata, and all of its member tables to be permanently deleted.

The following shows the syntax for PROC SPDO with a CLUSTER CREATE command.

```
PROC SPDO LIBRARY=<domain-name>;
       CLUSTER CREATE <cluster-table-name>
              MEM|MEMBER=<membername>
              MEM|MEMBER=<membername>
              ...
              [MAXSLOT=<max-slot-num-spec>]
              [DELETE=YES|NO]
;
```

Where:

- <cluster-table-name> is the name of the cluster table to be created
- <membername> is a list of member tables for the cluster
- MAXSLOT is an optional maximum number of member tables for the cluster
- DELETE=YES is optional to allow the cluster to be deleted

When you run the CLUSTER CREATE command, it initiates the creation of a new layer of metadata, which contains information about the SAS Scalable Performance Data Server tables that are included in the cluster. The SAS Scalable Performance Data Server tables listed in the command are hidden from direct access by users, making the data accessible through the cluster only. Underneath the top layer of metadata, nothing is changed. Each table still contains the same columns, rows, and indexes. You can see the tables in a cluster in output from the CONTENTS procedure.

The following example illustrates creating a cluster table *MyCluster* with 6 member tables.

```
PROC SPDO LIBRARY=&domain
       CLUSTER CREATE MyCluster
              MEM = table1
              MEM = table2
              MEM = table3
              MEM = table4
              MEM = table5
              MEM = table6
QUIT;
```

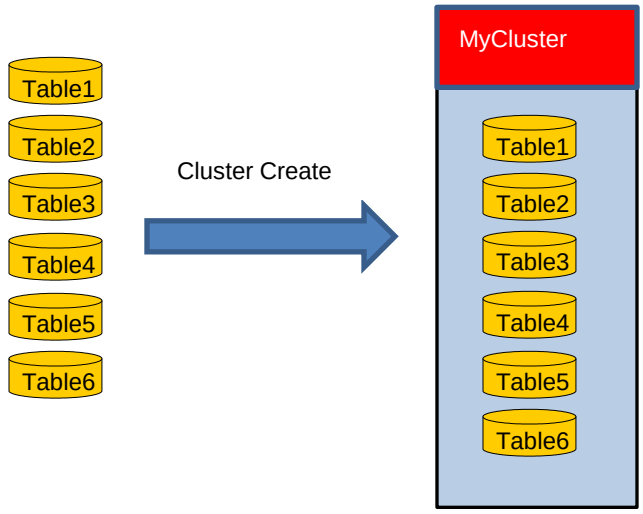

Figure 2. Create a Cluster

The CLUSTER UNDO command removes the cluster metadata. When the metadata layer is removed, you can view the tables in the cluster as normal SAS Scalable Performance Data Server tables. The following example shows the syntax for the CLUSTER UNDO command.

```
PROC SPDO library=<domain-name>;
       CLUSTER UNDO <cluster-table-name>;
QUIT;
```

Where <cluster-table-name> is the name of the cluster table to be undone.

The following example illustrates undoing cluster table MyCluster.

```
PROC SPDO LIBRARY=domain-name
       CLUSTER UNDO MyCluster;
QUIT;
```

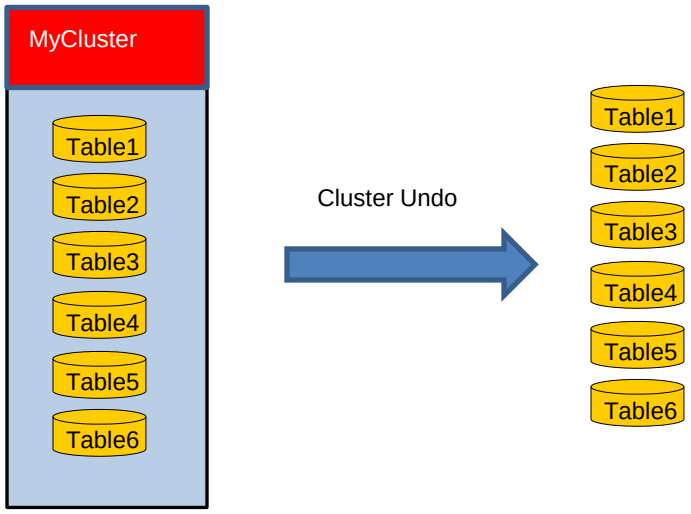

Figure 3. Undo a Cluster

The CLUSTER DELETE command removes the cluster metadata and permanently deletes the member tables. The CLUSTER DELETE can be done only on a cluster that was created with the DELETE=YES option. The following example shows the syntax for the CLUSTER DELETE command.

```
PROC SPDO library=domain-name;
       CLUSTER UNDO <cluster-table-name>;
QUIT;
```

Where <cluster-table-name> is the name of the cluster table to be undone.

Rules for Managing Data Via Dynamic Cluster Tables

Dynamic cluster tables are read-only structures. Therefore, you cannot perform updates, insertions, deletions, or modifications of rows for data that is stored in a dynamic cluster. In addition, you cannot create indexes, append rows, or change column labels and formats. To work with individual member tables, you must use the UNDO command to remove the cluster metadata and access the tables as a normal SAS Scalable Performance Data Server table. Once you make the changes, you can re-create the cluster using the CLUSTER CREATE command.

Cluster Table Transactions

The basic cluster commands to create and undo the cluster are sufficient if your cluster does not need to be modified by adding new members, deleting members, or modifying existing members. To do these operations would require undoing the cluster to either modify existing members, add new members or not including existing members when the cluster is recreated. However, there is a problem with this approach. Undoing the cluster makes it unavailable for access while the maintenance is being done. Additionally, undoing a cluster requires exclusive access, and cannot be done while any query on the cluster is in progress. Therefore, performing the cluster maintenance would require scheduling some query downtime on the cluster to provide sufficient time for queries to complete, as well as scheduling downtime on the cluster while the maintenance is being done. This makes doing cluster maintenance unacceptable for customer sites that require 24 by 7 query access to their data.

To solve this problem, the SAS Scalable Performance Data Server 5.1 introduces the concept of cluster member table transactions, features that simulate transaction isolation to modify large sections of your cluster table concurrently **without** affecting on-going queries to the cluster. Cluster table transactions work on a member table units. Cluster table append or modify transactions allow the user to begin their cluster table transaction by creating a new cluster member table that can be committed to the cluster by either adding the new member, or replacing an

existing member of the table. Cluster table delete transactions are committed by the user removing existing members of the cluster table.

The combination of adding new members, replacing existing members, or removing existing members gives the user all of the facilities to manage updates to the cluster. Each of the cluster member transactions can be done on a cluster table while it is being queried. Queries started before the cluster transaction is committed will complete the query in the state of the cluster when their query began. Subsequent queries started after the cluster transaction is committed will query the cluster with the changes.

Adding Member Tables to a Cluster

Use the CLUSTER ADD command to add new members to a cluster table. The Cluster Add can be done concurrently with on-going queries on the cluster. Any on-going query on the cluster will use the cluster member tables when the query began. Subsequent queries to the cluster table will see the newly added members.

The CLUSTER ADD command requires two options:

1. The name of the cluster table (*cluster-table-name*).
2. A list of member tables that will be added to the cluster (using the MEM= option).

The following shows the syntax for PROC SPDO with a CLUSTER ADD command.

```
PROC SPDO LIBRARY=<domain-name>;
       CLUSTER ADD <cluster-table-name>
              MEM|MEMBER=<membername>
              MEM|MEMBER=<membername>
              …
;
```

Where:

- <cluster-table-name> is the name of the cluster table to add members to
- <membername> is a list of member tables to add to the cluster

The following example illustrates using cluster add to add new member tables table7 and table8 to MyCluster.

```
PROC SPDO LIBRARY=&domain
       CLUSTER ADD MyCluster
       MEM = table7
       MEM = table8
QUIT;
```

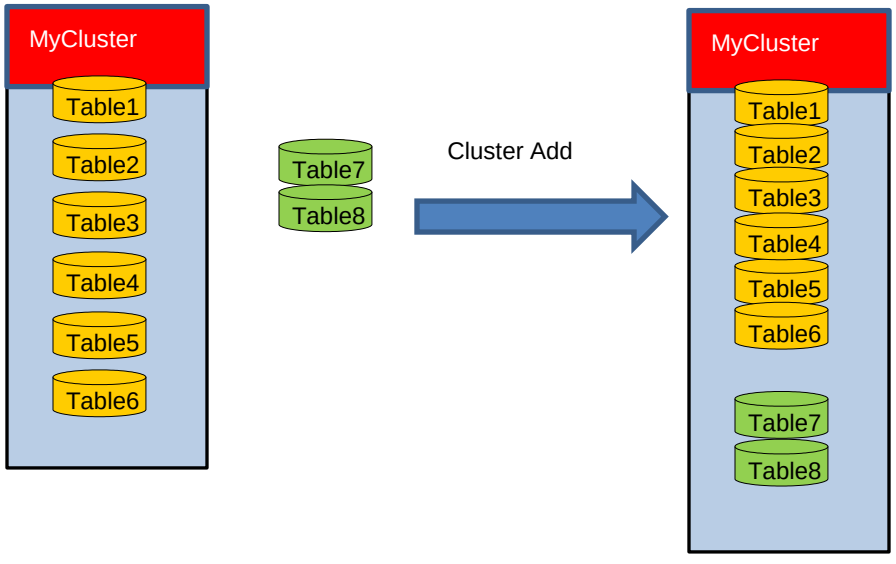

Figure 4. Add Tables to a Cluster

Removing Member Tables From a Cluster

Use the Cluster Remove command to remove existing members from a cluster table. The Cluster Remove can be done concurrently with on-going queries on the cluster. Any on-going query on the cluster will see the member tables when the query began. Subsequent queries to the cluster table will not see the removed members.

The CLUSTER REMOVE command requires two options:

1. the name of the cluster table (*cluster-table-name*) that will be created
2. a list of member tables that will be removed from the cluster (using the MEM= option)

The following shows the syntax for PROC SPDO with a CLUSTER REMOVE command.

```
PROC SPDO LIBRARY=<domain-name>;
       CLUSTER REMOVE <cluster-table-name>
              MEM|MEMBER=<membername>
              MEM|MEMBER=<membername>
              ...
;
```

Where:

- <cluster-table-name> is the name of the cluster table
- <membername> is a list of member tables to remove from the cluster

The following example illustrates using cluster remove to remove member tables table7 and table8 from MyCluster.

```
PROC SPDO LIBRARY=&domain
       CLUSTER REMOVE MyCluster
              MEM = table7
              MEM = table8
QUIT;
```

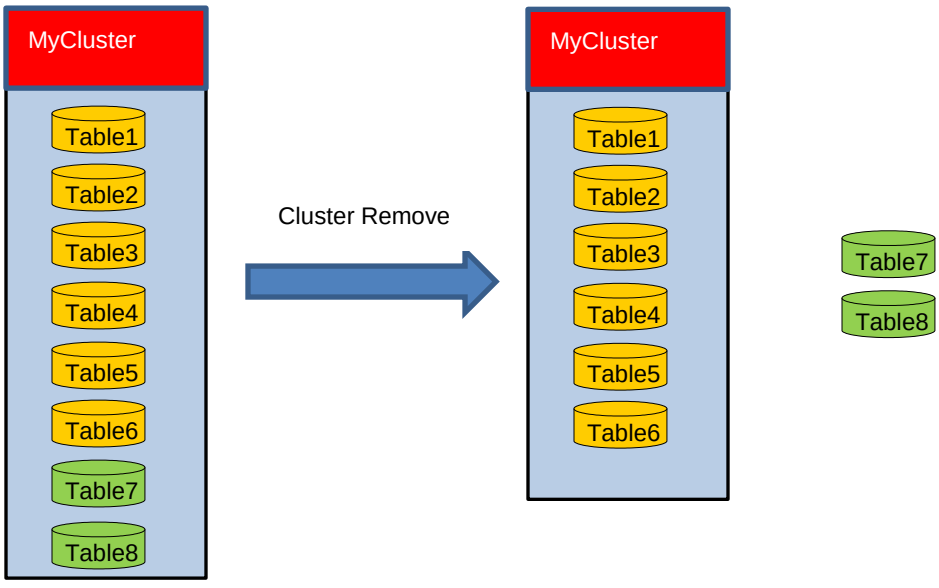

Figure 5. Remove Tables from a Cluster

Replacing Member Tables in a Cluster

Use the Cluster Replace command to replace existing members from a cluster table with new members. The Cluster Replace can be done concurrently with on-going queries on the cluster. Any on-going query on the cluster will see the member tables when the query began. Subsequent queries to the cluster table will see the replaced members.

The CLUSTER REPLACE command requires two options:

1. The name of the cluster table (*cluster-table-name*) that will be created
2. A list of member tables to replace existing member tables in the cluster (using the MEM= option)

The following shows the syntax for PROC SPDO with a CLUSTER ADD command.

```
PROC SPDO LIBRARY=<domain-name>;
       CLUSTER REPLACE <cluster-table-name>
       OLDMEM=<membername> NEWMEM=<membername>
       OLDMEM=<membername> NEWMEM=<membername>
       ...
;
```

Where:

- <cluster-table-name> is the name of the cluster table to replace members
- OLDMEM=<membername> NEWMEM=<membername>is a list of member tables to replace

The following example illustrates using cluster replace to replace member table table1 with table7, and member table2 with table8 from MyCluster.

```
PROC SPDO LIBRARY=&domain
       CLUSTER REMOVE MyCluster
              OLDMEM = table1 NEWMEM=table7
              OLDMEM = table2 NEWMEM=table8
QUIT;
```

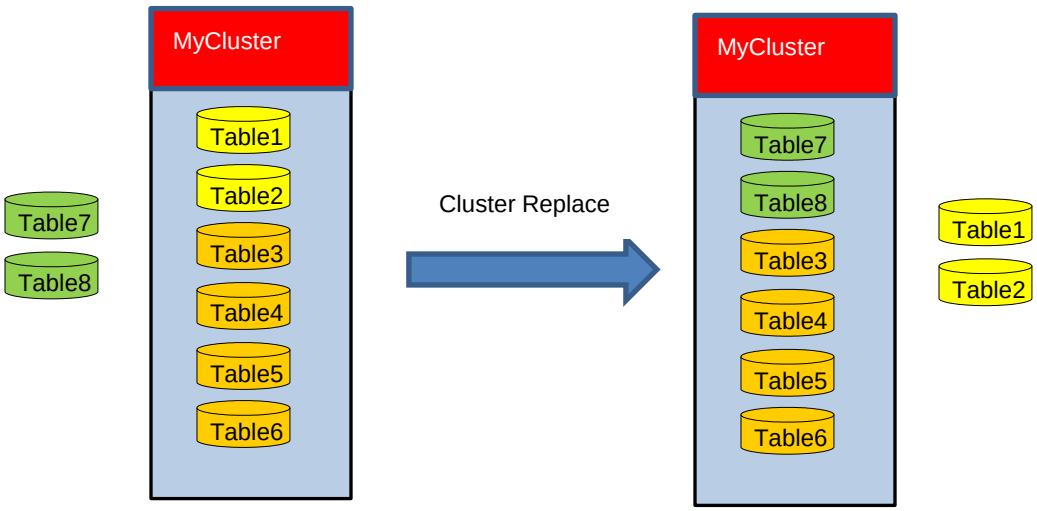

Figure 6. Replace Tables in a Cluster

What Happens to Cluster Members that are Removed or Replaced?

When you remove a member from a cluster by using either the cluster delete or cluster replace command, the member table is made visible to use again. However, because the member table can be in the process of being used by an on-going cluster read operation, the member is marked so that it cannot be updated. When you are sure that no on-going cluster command is using the table, you can use the cluster fix command to allow it to be updated.

The following shows the syntax for PROC SPDO with a CLUSTER ADD command.

```
PROC SPDO LIBRARY=<domain-name>;
       CLUSTER FIX MEM=<table-name>
;
```

If you want to delete the member table after it is removed from the cluster, the scenario depends on whether your implementation is UNIX or Windows. For a UNIX implementation you can delete the table (using proc datasets delete, or using proc sql drop) while there is an on-going query of the cluster that was using the member table. The UNIX file system will no longer make the table visible so no subsequent queries can be done. However, the UNIX file system will maintain a shadow version of the table until the SPDS Server closes all references of the table, thus allowing on-going queries to complete. When all on-going queries have finished, UNIX will free up the disk space for the table.

For a Windows implementation, you cannot delete the table while there is an on-going query of the cluster using the member table. Windows will return a file system error if the table is in use. You can retry the delete until the delete succeeds, meaning there are no longer any on-going cluster queries using the table.

Example using Cluster Table Transactions

Consider a case where a company has 18 months of historical data they want to analyze, from January 2012 through June 2013. Each month, a new batch of data is to be added to the analysis, up to a 24 month period. When the 24 month period is loaded, as a new month is added to the analysis, the oldest month is to be removed to maintain the 24 month period.

Requirements include:

- An historical month of data may require some updates.
- A maximum of 24 months of historical data.
- 24/7 access to the data.
- Very fast modifications.
- A query must see the entire modification (add/update/delete).

SAS Dynamic Cluster tables with the SAS Scalable Performance Data Server 5.1 can be used to meet all of the above requirements.

Initial load of the data

To initially load the 18 months of historical data, an ETL job is run to break the data up into 18 tables. Each table must have a data column that includes the month and day of row. This allows monthly queries to affect only one cluster member. The ETL job can be run in parallel to create the per-month tables concurrently.

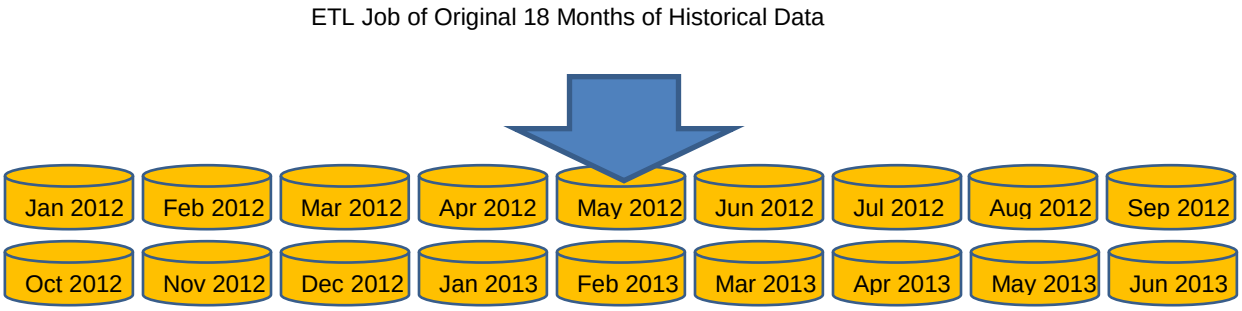

Figure 7. Load Monthly Data

At the completion of the ETL job, use PROC SPDO with the CLUSTER CREATE command to create dynamic cluster *SalesHistory*.

```
PROC SPDO LIBRARY=domain-name;
       CLUSTER CREATE SalesHistory
              MEM=Jan2012  MEM=Feb2012 MEM=Mar2012
              MEM=Apr2012  MEM=May2012 MEM=Jun2012
              MEM=Jul2012  MEM=Aug2012 MEM=Sep2012
              MEM=Oct2012  MEM=Nov2012 MEM=Dec2012
              MEM=Jan2013  MEM=Feb2013 MEM=Mar2013
              MEM=Apr2013  MEM=May2013 MEM=Jun2013
;
```

Monthly Additions for Months 18-24

For the next six months, as the new monthly data arrives, an ETL job is run to create a new monthly table that is added to the SalesHistory cluster using the CLUSTER ADD command. The example shows adding member table Jul2013. Adding member tables Aug2013 through Dec2013 would follow the same logic.

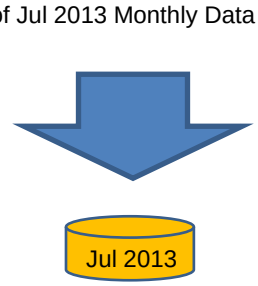

Figure 8. Create New Monthly Table

At the completion of the ETL job, use PROC SPDO with a CLUSTER ADD command to add member table *Jul2013* to dynamic cluster *SalesHistory*.

```
PROC SPDO LIBRARY=domain-name;
       CLUSTER ADD SalesHistory
       MEM=Jul2013
;
```

Ongoing queries started before the CLUSTER ADD will complete without interruption using the state of the cluster when the query began.

Adding a New Month That Would Exceed the 24-month Requirement

When monthly data for January 2014 is to be added to *SalesHistory*, it cannot be added without breaking the requirement of a maximum of 24 months of historical data. To maintain the 24 month requirement, use PROC SPDO with a CLUSTER REPLACE command to replace the oldest member table of the *SalesHistory* cluster, *jan2012*, with the newest member *jan2014*.

The example shows replacing month Jan2012 with Jan2014. Subsequent months would follow the same logic.

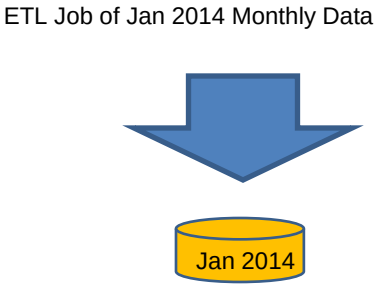

Figure 9. Replace Monthy Data

```
PROC SPDO LIBRARY=domain-name;
       CLUSTER REPLACE SalesHistory
       OLDMEM=Jan2012 NEWMEM=Jan2014
;
```

Ongoing queries started before the CLUSTER REPLACE will complete without interruption using the state of the cluster when the query began.

Updating an Existing Month

The final requirement for the data is that monthly data may need to be periodically updated. As previously mentioned, you cannot modify the rows of a SAS Scalable Performance Dynamic Cluster. To perform the update you would need to replace an existing monthly member table with a modified version of the table using the Cluster Replace command.

The following example shows replacing month Jan2014 with a modified version of that table.

```
/* Create a copy of the member table to be modified, using the date column to
 * only select rows for month Jan2014 in the SalesHistory cluster
 */

Libname foo sasspds "<domain>" host="<hostname>" serv="<service>" user="<user>"
                    Password="<passwd>";
PROC SQL;
create table foo.jan2014_new as select * from foo.SalesHistory where
                 where History_DTE between '01Jan2014'd and '31Jan2014'd;

/* Modify or delete some existing rows in the Jan2014_new table */

Update table foo.jan2014_new ……;
Delete from table foo.jan2014_new where …..;
Quit;

/* Append some new rows to the new member table */
Proc append base=foo.jan2014_new data=foo.jan2014_more_data;
Run;

/* Use Cluster replace to replace the old jan2014 member table with
 * the modified jan14_new member table
 */
```

```
      Proc SPDO lib=foo;
      CLUSTER REPLACE SalesHistory
                      OLDMEM=jan2014 NEWMEM=jan2014_new
;
```

Ongoing queries started before the CLUSTER REPLACE will complete without interruption using the state of the cluster when the query began.

SPDS 5.1 Cluster Table SQL Extensions

The SAS Scalable Performance Data Server 5.1 software introduces SQL language extensions to be able to manage your SPDS clusters. This allows you to manage cluster operations such as creating and deleting a cluster, as well as cluster table transactions to update your cluster table via SAS Proc SQL with explicit pass-through, or 3rd party access via the SAS JDBC or ODBC driver. One main difference between the cluster table SQL extensions and Proc SPDO commands, is the SQL extensions do not support a list of cluster member tables. The SQL extensions only support a single member table.

Syntax for the cluster table SQL extensions is:

1. To create a cluster table:

```
Create cluster <cluster>[delete=yes] with table <table>;
```

2. To undo a cluster table:

```
Undo cluster <cluster>;
```

3. To add a member to a cluster table:

```
Cluster add <cluster> with table <table>
```

4. To remove a member from a cluster table:

```
Cluster remove table <table> from <cluster>
```

5. To replace a cluster member table:

```
Cluster replace table <table> with table <table> from <cluster>
```

CONCLUSIONS

It is no longer acceptable for many sites to allow any downtime for queries to strategic business data. However, it is also generally necessary to periodically update business data to add new information, or modify existing information. The SAS Scalable Performance Data Server 5.1 software provides the means necessary to meet these requirements using SPDS Dynamic Cluster Tables with SPDS 5.1 Cluster table transactions.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Guy Simpson
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
Guy.Simpson@sas.com
http://www.sas.com