Paper 330-2008

Avoid Growing Pains: New Cube Update Features You Should Know About

Ann Weinberger, SAS, Cary, NC
Janet Tierney, SAS, Cary, NC

ABSTRACT

You may have heard about the new cube update feature coming in SAS® 9.2 but may not be sure of just what it entails. Learn exactly what cube update is and how to get started with it. These "how-to's" will be discussed:

- distinguishing between managed and in-place update
- adding new data as well as new hierarchical members
- updating existing member properties
- changing cube captions

INTRODUCTION

After a SAS® OLAP cube is created, it is possible to update the data for the cube without completely recreating it. The SAS® 9.2 OLAP Server enables you to incrementally update SAS OLAP cubes. An *incremental update* involves adding cell data and members to an existing SAS OLAP cube. Incremental updates of a cube are generally faster than rebuilding the cube from the combined set of input data and update data.

There are several decisions you must make before you create your cube that will make the incremental update process run smoother. You will need to decide how to structure your cube and the input data as well as choose a method of update. The update procedure itself has many options that you can use to control what parts of the cube will be updated. Once the cube is updated, there are maintenance tasks that you should perform periodically. As you update the cube, query performance may start to degrade. This is due to the extra racks of data that are added during the update. One maintenance task is to coalesce the racks of data into a single aggregation table in order to restore performance.

BEFORE THE UPDATE

NON-UPDATEABLE DIMENSIONS

There are several decisions you must make before you create your cube that will allow the incremental update process to run smoother. First, you must decide which of your dimensions can be updated with new members. As a cube is created, a list of members is created from the dimension tables. When you update a cube, you can add new members to the dimension in addition to adding new cell data. By default, each dimension in a cube can have new members added during update. However, at build time, you can mark a dimension as non-updateable by adding the NONUPDATEABLE keyword to the DIMENSION statement. Marking a dimension as non-updateable will save space in the cube aggregation files.

```
Proc olap data=input.salesdata cube=salescube;
metasvr olap_schema="SASApp – OLAP Schema";
dimension geography nonupdateable hierarchies=(geography);
hierarchy geography levels=(country state);
level country;
level state;
dimension products hierarchies=(products);
hierarchy products levels=(type group item);
level type;
level group;
level item;
measure actualSales column=actual statistic=sum;
```

In this example, the products dimension can have new members added during an update while the geography dimension cannot. Use of the NONUPDATEABLE keyword is recommended for dimensions that have a static list of

members. If a dimension is marked as non-updateable, new members may be added only by fully recreating the cube.

INPUT DATA

When you update a SAS OLAP cube, you should consider how the input data will be stored and made available for the cube update process. All incremental updates must be loaded from tables that are similar in structure and content to the tables that were originally used to build the cube. The tables must also be registered in the metadata repository. The data or fact table used in the incremental update should contain only the new fact data that you want to add to the cube. This can be done by creating a table separate from the original data or fact table. Or, you can add the new data to the original table and then subset the table based on a timestamp or other subset criteria. If the cube were built from a star schema, the dimension tables used during update should contain the new members to match any new fact keys. The new member data can be contained in a new dimension table or added to the original dimension table.

One important thing to note is that the data used to update the cube must be available along with the original data in order to rebuild the cube. You may want to make a structural change to a cube, such as add a new dimension, hierarchy, or measure. In order to do this, you must rebuild the cube. To rebuild a cube in its entirety, all of the data used as input to the cube (since the original cube was created) must be available in a single joined table or a view of the separate input and update tables.

Note: A cube must have a MOLAP NWAY aggregation in order to be updated. Therefore, cubes created from a fully summarized or NWAY table cannot be updated incrementally.

TYPES OF UPDATE

There are two ways to update your SAS OLAP cubes: *in-place cube update* and *managed cube update*.

The simplest and most straightforward method for updating a cube is to update it in place. With this approach, a new generation of the cube is created and the previous generation of the cube is automatically deleted. The new data in the cube will be available for new queries against the cube. There will be no interruption in the online availability of the cube. However, there will also be no time to test the data in the cube before it is made available.

To perform an in-place update, you use the UPDATE_INPLACE option along with the ADD_DATA option.

```
proc olap data=mylib.newdata cube=prodcube add_data update_inplace;
metasvr olap_schema=prodSchema;
run;
```

The update in-place process does **not** create a new metadata registration when the update is complete. The existing metadata registration points to the new cube generation, and the old cube generation is then deleted. Furthermore, you do not have to disable the cube prior to beginning the in-place update.

A managed cube update maintains, as a separate cube, the generation that is created during an incremental cube update. With the managed cube update process, you determine and control the sequence of events that result in a cube update. Initially, a new generation of the cube is created. You may then determine any additional modifications and updates to that generation. This may include administrative tasks, including running verification tests against the cube, and making any necessary updates to the cube's security. Next, you switch out the old cube generation with the new one. And finally, you decide whether you want to delete the previous cube generation or make it available online or offline. The steps to perform a managed update are:

1. Create a new generation of the cube using the ADD_DATA option. You must specify either a new cube name using the OUTCUBE option or a new OLAP schema for the cube using the OUTSCHEMA option. You may also specify both OUTCUBE and OUTSCHEMA.
2. Perform any tests or data validation that is needed. Make any additional modifications that are required.
3. Disable the production version of the cube. After you have created the new generation and verified it, you should disable the current production cube in order to switch out the old cube generation with the new one. To disable a cube, you can use SAS® OLAP Cube Studio, the SAS® OLAP Server Monitor plug-in to SAS® Management Console, or the PROC OLAPOPERATE DISABLE CUBE statement. When a cube is disabled, it is taken offline and is not available for new queries. Existing queries will continue to run to completion. The cube will not show up in the list of available cubes in client sessions.

4. Rename the cubes. After you have disabled the production cube, you can use the PROC OLAP RENAME option to “switch” the cube generation. First, rename the original cube to another name. Then, rename the new cube generation to the original cube name. You can also rename the cube in SAS OLAP Cube Studio.
5. Enable the new production cube. After you have switched the cube generations, you can enable the new cube generation. To enable a cube, you can use SAS OLAP Cube Studio, the SAS OLAP Server Monitor plug-in to SAS Management Console, or the PROC OLAPOPERATE ENABLE CUBE statement. This enables queries to resume on the server for the new production cube.

If your data can be made available immediately without testing or other modifications, then an in-place update of the cube would be appropriate. Managed update of a cube is appropriate when you want to verify the updated contents of the cube or make additional changes to the cube such as updating the security. If you choose to use managed update of the cube, you will need to decide how many generations of the cube you want to maintain. You should also choose a naming convention for the cube and its generations that will allow you distinguish among the different generations.

DURING THE UPDATE

Several new options have been added to PROC OLAP for cube update. These options control what type of update you will do and what parts of the cube will be updated. The primary options that are used during incremental cube update are:

ADD_DATA
: indicates that you will be adding new cell data to the cube. It is used when doing either an in-place or managed update of the cube.

OUTCUBE
: specifies a new name for the new cube. This option is used during managed update to give the name for the next generation of the cube.

OUTSCHEMA
: specifies a new OLAP schema for the new cube. This option is used during managed update to put the new cube in a different OLAP schema than the original cube. Since the cube is in a different OLAP schema, the new generation can have the same name as the old generation of the cube.

UPDATE_INPLACE
: indicates that the cube will be updated in place. The new generation of the cube is created and named the same as the old generation. The old generation is deleted.

There are two other options that can be used to control what parts of the cube will be updated: UPDATE_DIMENSION and UPDATE_DISPLAY_NAMES.

The UPDATE_DIMENSION option specifies whether new members are added to the cube during an update and if the member properties for existing members will be updated. If you use the ADD_DATA option, the default behavior is to add new members and their member properties to the cube from the dimension tables used in the update. However, the member properties for existing members are not updated. If you specify UPDATE_DIMENSION=MEMBERS_AND_PROPERTIES, the member properties for existing members will be updated from values found in the dimension tables.

```
Proc olap cube=prodcube data=input.update add_data update_inplace;
Metasvr olap_schema=prodschema;
Dimension products update_dimension=members_and_properties;
Dimension time update_dimension=members;
```

For a cube loaded from a star schema, it is possible to add new members and update member property values without adding new cell data to the cube. Simply do not specify an input table and omit the ADD_DATA option from the PROC OLAP statement. Then, include a dimension statement for each dimension that you want to update, and add the UPDATE_DIMENSION option. Set the option to MEMBERS to add new members but ignore the member properties. Set the option to MEMBERS_AND_PROPERTIES to add new members and update the member property values for existing members. If the UPDATE_DIMENSION option is set to OFF on a dimension statement, that dimension will not be updated with new members or member properties.

```
Proc olap cube=prodcube update_inplace;
Metasvr olap_schema=prodschema;
Dimension products update_dimension=members_and_properties;
Dimension time update_dimension=members;
```

Since the update of members and member properties will create a new generation of the cube, the UPDATE_DIMENSION option requires either the UPDATE_INPLACE option or the OUTCUBE or OUTSCHEMA options.

The UPDATE_DISPLAY_NAMES option allows you to update the captions and descriptions for dimensions, hierarchies, levels, and measures. These are nonstructural changes to the cube and can be done without adding new data to the cube.

```
Proc olap cube=prodcube update_display_names;
Metasvr olap_schema=prodschema;
Dimension products caption="Product Dimension";
```

AFTER THE UPDATE

DATA MANAGEMENT

Now that the cube is updated, there are various maintenance tasks that can be done. First, you should perform any data management tasks that are needed to provide a complete set of input data should the cube need to be recreated in its entirety. This might involve a merge of the update table with the original input table or creating a view of the original input table and the update tables.

COALESCE AGGREGATIONS

Each time you perform an incremental update of your cube, the new data are aggregated and a new file containing that data is added to each aggregation. That new data portion is called a *rack*. Each cube update adds a new rack to the aggregations. As a result, the more cube updates you do, then the more racks each aggregation will have. Eventually, the number of aggregation racks can impact query performance. When query performance degrades due to the large number of racks, it is time to combine all of the individual rack tables into a single aggregation table. Aggregations are coalesced (combined) in a separate job from adding data to a cube.

The COALESCE_AGGREGATIONS option is used on the PROC OLAP statement to coalesce all aggregations on the cube. Only aggregations with multiple racks will be coalesced. The coalesce process is done in place on the cube, and a new generation of the cube is **not** created. Because the coalesce process updates the cube in place, the cube must be disabled before the coalesce process is executed.

```
Proc olap cube=prodcube coalesce_aggregations;
Metasvr olap_schema=prodschema;
```

You can choose to coalesce individual aggregations by omitting the COALESCE_AGGREGATIONS option on the PROC OLAP statement and instead including individual COALESCE_AGGREGATION statements.

```
Proc olap cube=prodcube;
Metasvr olap_schema=prodschema;
Coalesce_aggregation name=prodaggr;
```

EXPORTING UPDATED CUBES

After you have updated a cube, it may be necessary to export that cube for use in another SAS OLAP environment. You can export only the last generation of an updated SAS OLAP cube. When you import the cube, it is important that you do not have two metadata definitions pointing at the same set of physical files. To prevent this, you can change the path of the newly imported cube to point to a different physical location. The physical cube files can be moved or copied to this new location. Or, if you have the entire set of input and update data in the new environment, you can rebuild the cube after importing the definition.

CONCLUSION

SAS 9.2 OLAP Server provides the ability to incrementally update the data in your cube. You can update your cube in place and make the updated data available immediately. Or, you can use managed update to build separate generations of the cube that can then be “switched” or brought into production. PROC OLAP provides many new options that allow you to control what parts of the cube are updated. After the cube is updated, there are tools available to coalesce or combine the aggregation racks and to export and import the updated cube.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Ann Weinberger
SAS Institute Inc.
100 SAS Campus Dr., R1124
Cary, NC 27513-8617
(919)-531-6607
Ann.Weinberger@sas.com

Janet Tierney
SAS Institute Inc.
100 SAS Campus Dr., R1121
Cary, NC 27513-8617
(919)-531-4936
Janet.Tierney@sas.com