Cube Storage and Aggregation

Objectives

- Understand basic storage.
- View aggregation designs.
- Customize aggregation designs.
- Understand advanced storage.
- Implement proactive caching.
- Use relational and SSAS partitions.
- Customize cube and dimension processing.

The files associated with this chapter are located in the following folders:

- `{Install Folder}\CubeStorage`
- `{Install Folder}\CubeStorageLab`
- `{Install Folder}\CubeStorageLabCompleted`
Basic Storage: MOLAP

This chapter discusses methods for storing SSAS data and metadata, but first you’ll learn about metadata: the kind of information it comprises and the format in which that information is created.

After you complete the design work for your cubes and dimensions in BIDS, you will build and process these structures by using menus in BIDS. Once you have processed your cube, you can view the cube in the BIDS browser or in any other client application.

Build verifies the structure of the metadata; process makes the metadata and data files available to the SSAS server that you specify in the <SolutionName> Property Pages, Deployment Configuration Property section, Target subsection, Server property.

The metadata for SSAS in this version of the product is produced in the form of an XML dialect. This dialect is an open standard for the OLAP community. It is called XML for Analysis, or XMLA.

About XMLA

You can think of XMLA as an SSAS dialect of XML that is similar to SOAP (Simple Object Access Protocol, which is used with Web services). Used in conjunction with the SSAS query language (Multidimensional Expression Language or MDX), XMLA is the language of cube metadata. For an example of XMLA, see Figure 1.

XMLA is also used to perform basic administration such as backup or restore on an SSAS database.
TIP: To generate the XMLA for an SSAS object, you can use SQL Server Management Studio (SSMS). There you connect to SSAS and right-click the object you want to script. Then select the appropriate Script option such as, **Script Dimension as > CREATE to> New Query Editor Window**, etc.

Once you understand the concept of XMLA, the next step is to understand what type of information is data in SSAS, and what type of information is metadata.

Data is the content (or the rows) in the fact table. Metadata is everything else—cube definitions, dimension names, attribute names, hierarchy names, and, most importantly, dimensional attribute values. Here’s a brief list with some of the samples available in SSAS to help explain this concept more fully:

- **Data**: All fact table rows from Fact Internet Sales, Fact Internet Sales Reason, and so on.
- **Metadata**: Names and all attributes for the Customer dimension called Customer; the Customer dimensional hierarchy called Customer Geography; the Customer Attribute called Marital Status; and the Customer attribute values Married, Single, or Unknown.
Three Storage Modes

When you design storage for your cube or partition, your primary concern is how to store the data (the fact table rows). You can select from the following three methods for SSAS cube data storage.

**NOTE** The ability to create multiple partitions for a measure group requires the Enterprise Edition of SSAS. Although at the time of writing Microsoft has not updated the SSAS Performance Guide for SSAS 2008, several sources recommend limiting partitions to approximately 2 GB in most environments for SSAS 2005 SP2.

- **MOLAP (Multidimensional OLAP):** Stores a copy of the fact table rows (or facts) within the Analysis Services database. This is not a one-for-one storage option. Due to the efficient storage mechanisms used for cubes, the stored data is significantly smaller than the original data; i.e., if you have 1 GB in your fact table, plan for around 200 MB of storage on SSAS.

  MOLAP is the default storage option. The SSAS query engine is optimized to run queries against MOLAP data. Queries to MOLAP data execute significantly faster than queries against the other two storage modes. Additionally, processing is typically faster on MOLAP cubes than on ROLAP cubes. This is due to the time required to write the aggregations to the relational data source.

- **HOLAP (Hybrid OLAP):** Does not make a copy of the facts on the SSAS server, but reads this information from the data source. Additionally, HOLAP stores aggregations in the optimized SSAS database. This option offers a compromise between MOLAP and ROLAP and is often used for very large quantities of data—particularly in situations where some of the data could be considered archival (meaning that it is queried infrequently). Processing time is typically quicker than either MOLAP or ROLAP. This is because the fact data does not need to be copied to the SSAS database while aggregations are written to the more efficient SSAS database rather than the relational data store. Although HOLAP queries are typically slower than MOLAP queries, when accessing aggregates rather than detailed data, HOLAP queries outperform ROLAP queries.

- **ROLAP (Relational OLAP):** Does not make a copy of the facts on SSAS, but reads this information from the relational database source. Unlike HOLAP, ROLAP writes the aggregations to the relational database source. Although ROLAP typically provides the slowest query performance and often the slowest processing performance, it offers data access that is the closest to real time that SSAS can offer.
NOTE With one exception (a ROLAP dimension), metadata is always stored within the SSAS multidimensional structure. ROLAP dimensions are discussed later in this chapter.

You can view the storage mode for a cube or a dimension by using the built-in report viewer in SSMS. There are other ways to get this information, but this is probably the simplest.

**Try It Out!**

Use the following steps to connect to SSAS in SSMS and see how to generate a report that will allow you to view metadata about the particular item you’ve selected to work with.

1. Select **Start|All Programs|Microsoft SQL Server 2008|SQL Server Management Studio** to open SSMS.

2. In the Connect to Server dialog box, select **Analysis Services** from the drop-down list next to the Server type label; type in your SSAS server name next to the Server name label, and then click **Connect** to connect to your SSAS instance.

3. In the Object Explorer window, click on the + symbol next to the name of your SSAS instance to expand the tree. Continue to click the nested + signs until you’ve drilled down to the **Cubes** folder of the Adventure Works DW 2008 database.

4. Right-click on the **Adventure Works** cube and then select **Reports|Standard Reports|General** to generate a metadata report about the cube. This report, shown in Figure 2, lists the storage mode.
About Aggregations

When you design storage for cubes and dimensions, the number of aggregations and the storage of aggregations are major considerations. An OLAP aggregation is a calculated, stored intersection of some fact table values (or facts). Aggregations are calculated at a level higher in a defined rollup hierarchy than is the load of the fact table. The MDX Sum function is the default aggregation; however, you can change the aggregation behavior as your business requirements dictate. Other MDX aggregation functions include Count, Min, Max, DistinctCount, AverageOfChildren, and more.

NOTE Generally, you can change the default aggregation in one of two ways. The first is to simply set the value of the AggregateFunction property for a Measure to one of the listed MDX functions, as shown in Figure 3. The second way is to write a custom MDX script.
Figure 3. You can use any of the MDX aggregate functions in the drop-down list.

For example, if your fact table holds a fact (such as sales amount) for each product sold on each day, by each employee, a possible aggregation would be to sum (and store) that information at the week level, or for each week for all rows in the fact table.

An important difference between OLAP aggregations and materialized (or stored) indexes on calculated columns in relational data is that SSAS can use aggregations that were built on any level of a hierarchy in response to a query. The result of this behavior is that full cube aggregation is never needed to optimize queries; rather it’s typical to aggregate only around 20-30%.

An example of this would be to write a query that looks for the sales amount for each product, for each employee by year (from the scenario above. If there were aggregations at the week level, SSAS would use them to calculate the results of the query for the yearly results rather than read each row of the fact table (assuming it was loaded at the day level).

The storage type you select will affect the storage location for any aggregations. By default, MOLAP with no aggregations is the default storage mode for all measure groups. Should you choose to add any aggregations with MOLAP, they will be stored in the native SSAS format on SSAS as well. For HOLAP, only aggregations, not data (or fact table rows), are stored on SSAS. For ROLAP, aggregations are “written back” to the relational database.

NOTE Although you can configure ROLAP and HOLAP partitions with the Standard Edition of SSAS, this edition only supports a single partition for each measure group. Additionally, proactive caching is not available in the Standard Edition. More information on proactive caching is provided later in this chapter.
Viewing Aggregation Designs

New in SSAS 2008, the Aggregations subtab in the Cube Designer allows you to better manage aggregations for each cube partition. Additionally, SSAS 2008 provides you with the ability to create multiple aggregation designs and then assign the appropriate design to the measure group. The Aggregations subtab is shown in Figure 4.

As you can see in Figure 4, this subtab uses the blue squiggly line to indicate an AMO best practice warning, similar to other designer subtabs within BIDS. In this case, the best practice states that the 50 percent performance gain design should be deleted because it is not assigned to any partitions.

In addition to the Standard view shown in Figure 4, the Advanced view, shown in Figure 5, allows you to fine tune individual aggregations. In this view, you select the Measure Group and Aggregation Design that you want to manage from the drop-down lists.
Figure 5. The Advanced View of the Aggregations subtab allows you to fine tune aggregations.

From the Aggregations subtab toolbar, when the Standard view is active, you can select the Design Aggregations button to launch the Aggregation Design Wizard. This wizard is discussed in detail later in this chapter.

**Try It Out!**

Use the following steps to examine the process used in BIDS to view and modify storage settings. You’ll then take a look at the Aggregation Design Wizard.

1. Select **Start|All Programs|Microsoft SQL Server 2008|SQL Server Business Intelligence Development Studio** to open BIDS.

2. Select **File|Open|Analysis Services Database**, then in the Connect To Database window, click on your server instance name that corresponds with the Adventure Works DW 2008 database, and then click **OK**. This opens the Adventure Works DW 2008 SSAS sample database. If the server and database are not listed in the selection box, type the name of your server in the Server box in the Connect to the existing database section, and then select the **Adventure Works DW 2008** database from the Database drop-down list.

3. In the Solution Explorer window, double-click on **Adventure Works** to open the cube in the BIDS Cube Designer.

4. Click the **Partitions** subtab to navigate to this work area.
5. Click on the chevrons next to Sales Reasons (1 Partition) to display the details for the Sales Reasons partition, as shown in Figure 6. Note that this partition was processed with the default storage setting of MOLAP with no aggregation design.

![Adventure Works sample cube](image)

Figure 6. The Adventure Works sample cube contains many partitions.

6. Click the Storage Settings link to open the Partition Storage Settings – Internet_Sales_Reasons dialog box. Note that the slider is set to MOLAP. Click on the slider and drag it to Real-time ROLAP, and then click OK.

7. Click the Aggregations subtab in the Cube Designer.

8. On the Aggregations page, on the left side of the design window, click the + symbol next to Sales Reasons (0 Aggregation Design) to expand the measure group.


10. On the Review Aggregation Usage page, review the options and notice that you can decide whether or not each attribute will be considered in the aggregation design, and then click Next.

11. Click the Count button on the Specify Object Counts page.

**TIP:** You will normally select Count within the Aggregation Design Wizard to allow SSAS to automatically count the distinct rows in the Sales Reason source fact table and related dimensions. The exception to this is when you have very large fact or dimension tables. Allowing the wizard to do this count can be very time consuming. When this is the case, you can manually enter the information into the appropriate locations on the Specify Object Counts page.
12. Leave BIDS open and then open SSMS. When prompted, select Database Engine from the Server type, enter the name of your relational database instance that holds the AdventureWorksDW2008 database, configure the appropriate Authentication for your environment, and then click Connect.

13. In SSMS, select File|Open|File, on the Open File page, select theRowCount.sql file located in the Samples folder for this chapter, and then click Open.

14. Review and execute the RowCount.sql script, and then review the results. Compare the results of the script to the row counts on the Specify Object Counts page in BIDS. The Specify Object Counts page is shown in Figure 7.

15. Close SSMS, do not save the changes if prompted, and return to BIDS.

16. In BIDS, on the Specify Object Counts page, click Next.

17. On the Set Aggregation Options page, click the radio button next to Performance gain reaches, accept the default of 30%, and click Start. Once the wizard finishes suggesting aggregations, notice that only one aggregation is selected, which optimizes the performance gain to the 50% level. Click Next.
18. On the Completing the Wizard page, name the aggregation design **50 percent performance gain**, click **Process now**, and then click **Finish**. If prompted to save changes, click **Yes**. After a minute or two, the Process Partition dialog box appears; click **Run**.

19. The Process Progress dialog box appears, and after a minute or two, you should see a “Process succeeded” message in the Status box. Click **Close** in the Process Progress dialog box.

20. Click **Close** in the Process Partition dialog box.

21. Leave BIDS open for the next Try It Out!.

**TIP:** So, which type of storage is best? Why is MOLAP with 0% aggregations the default mechanism? The reason for this default is that the SSAS query engine is highly optimized for calculating aggregates on large quantities of data—it is really more like Excel’s calculation engine than that of SQL Server (or any other relational database). The latter is designed to fetch subsets of data efficiently, not necessarily to sort, filter, or aggregate that data. For many business situations, the MOLAP storage mode provides optimal query performance and the best overall solution.

However, because only fairly small cubes will run optimally at 0% aggregations, you probably will want to add some aggregations. You’ll learn more about how to do that in the next section of this chapter.
Customizing Aggregations

BIDS provides you with several tools and wizards to help you easily add aggregations. And, as explained previously, cubes need a relatively small percentage of aggregations to produce a large increase in query performance. Another advantage in the SSAS environment is that the process of designing aggregations for an SSAS cube is much simpler than designing indexes for a relational database.

**NOTE** You can design aggregations for an entire measure group or for one or more specified partitions. In addition, you can assign the same aggregation design to more than one partition in the same measure group. Partitions are available only in the Enterprise Edition of SSAS.

Properly designed aggregations improve cube response times; however, aggregations also increase cube processing times as well as the storage space that the cube requires on disk. A common mistake is over-aggregation.

**TIP:** Some SSAS cubes do not require any aggregations to function acceptably. If your cube is quite small (under 1 GB), and if you have a small number of end users (100 or fewer), then you may decide not to add any aggregations.

SSAS includes two primary tools to help you automatically create aggregation designs: the Aggregation Design Wizard and the Usage-Based Optimization Wizard.

**The Aggregation Design Wizard**

The Aggregation Design Wizard (which you used in the last Try It Out!), is available in BIDS (and SSMS). Once you’ve designed aggregations, you can save the settings and process later, or process immediately using the new settings.
Within the Aggregation Design Wizard, you have the ability to decide how the aggregation designer will determine attribute usage when designing the aggregations. You can choose between the following options:

- **Default:** The designer applies a pre-programmed, default rule based on the type of dimension and attribute.

- **Full:** The designer ensures that aggregations for the cube will include the selected attribute or a related attribute that is lower in the attribute chain. If an attribute has a large number of members, you should avoid using this setting because it may keep aggregations from being designed due to excessive size.

- **None:** The designer cannot include this attribute in any aggregations.

- **Unrestricted:** The designer has no restrictions when you choose this option.

The **Set All to Default** button on the Review Aggregation Usage page resets all attributes to the Default setting.

Here’s a review of the options for designing aggregations when you use the Aggregation Design Wizard:

- **Estimated storage reaches:** Fill in a storage limit, as MB or GB, and SSAS will design aggregations up to that size limit for storage on disk.

- **Performance gain reaches:** Fill in a percentage increase in query performance speed and SSAS will design aggregations until that threshold is met. A good starting value for this is 20%.

- **I click stop:** Select the “stop point” based on the total size of aggregations as suggested by the wizard.

- **Do not design:** This is self-explanatory!

Of these options, Performance gain reaches will provide you with the best results. Remember, 100% aggregation is not required, nor is it advisable.

**The Usage-Based Optimization Wizard**

The Usage-Based Optimization Wizard automatically creates aggregation designs with more granularity and is available in SSMS and in BIDS (on the Partitions subtab). With this tool you ask BIDS to define aggregations for specific queries, based on a combination of configurable input parameters.
You must configure the following three SSAS server properties before running the wizard. To update these properties, connect to the SSAS server using SSMS, then right-click the server instance name, and select Properties.

- **QueryLogConnectionString**: Set this property to the database connection string where you want to store the query log table (called OlapQueryLog by default). The Usage-Based Optimization Wizard uses the data that you store in this table (similar to the way that the database tuning advisor for OLTP uses a trace table).

- **CreateQueryLogTable**: Set this property to True. You can optionally change the default name of the query log table for the database you previously defined. This property is set to OlapQueryLog by default.

- **QueryLogSampling**: The default is to capture only 1 of 10 queries. You will probably want to set this to 1 for your test environment sampling purposes. If you need to maintain a query log in a large production environment, you need to balance performance and log size against an accurate representation of the queries generated against your OLAP instance.

Figure 8 shows the Analysis Services Properties window for SSAS, where you set these values.

After you update these three properties and allow the server to collect a sufficient number of queries, you can run the wizard by connecting to SSAS in SSMS, right-clicking on any cube partition in Object Explorer, and then selecting Usage Based Optimization. Every cube will have at least one partition. You’ll learn more about partitions later in this chapter.
Once you start the wizard, you’ll ask SSAS to design aggregations based on any combination of the following parameter values: beginning date, ending date, specific user, and percentage of queries (which returns the most frequent queries.) After you complete this step, the wizard presents you with a list of queries that fall within those configured parameters.

Figure 9 shows a sample list of queries, from which you select the queries you want SSAS to design aggregations for.

![Figure 9](image)

Figure 9. In the Usage-Based Optimization Wizard you select the queries for which you want to design aggregations.

After accepting the queries by clicking Next, the following two pages resemble those in the Aggregation Design Wizard that you used in the previous Try It Out! The first of these pages in the Specify Object Counts page where you can either provide the row counts or you can allow the Wizard to count the rows. The default options on the Set Aggregation Options page differ from the Aggregation Design Wizard. In the Usage-Based Optimization Wizard, the default (and recommended) option is to **Design aggregations until Performance gain reaches 100%**. The only exception to this recommendation would be when disk space or processing constraints prevent the recommended aggregations.

New in SSAS 2008, the Completing the Wizard page allows you to either create a new aggregation design, or to merge and append the new aggregations to an existing aggregation design. You still have the option of whether or not you want to process the partition immediately.
Using Profiler

SQL Server Profiler is another interesting tool that you can use to help design aggregations more intelligently. You may be familiar with Profiler’s ability to capture whatever traffic and information you specify in the trace definition for OTLP databases. With SQL Server 2005 and 2008, you can use Profiler to capture activity on SSAS. This enables you to easily see which MDX queries are consuming the most resources on the server or are taking the longest to run. You can then design aggregations for the queries you identify using Profiler.

As with any type of query tuning, adding aggregations is only half of the solution. The other method of improving query performance is to rewrite the query in a more optimal way. While this is possible, it is done much less often in the world of SSAS than, for instance, rewriting T-SQL queries for SQL Server. The reason for this is the inherent complexity of MDX and the inability to control the code sent from front-end applications such as Microsoft Excel.

Try It Out!

Here you’ll use Profiler to look at the query output in SSAS.

1. In the Cube Designer in BIDS, click the Browser subtab. You’ll use this interface to generate some queries to the cube (which will be captured by Profiler). Leave BIDS open.

2. Select Start|All Programs|Microsoft SQL Server 2008|Performance Tools|SQL Server Profiler.


4. The Connect to Server dialog box opens; select Analysis Services as the server type, type in your server’s instance name, and then click Connect.
5. The Trace Properties dialog box opens. On the General tab, configure the trace per Figure 10 and accept the defaults for the remaining settings.

![Trace Properties](image)

Figure 10. Name your trace and enter a path if you want to save the file.

6. In the Trace Properties dialog box, click the **Events Selection** tab. Review the events and columns selected for the default trace. Note that the events are specific to SSAS.

7. Click the **Organize Columns** button. In the Organize Columns dialog box, click **Duration**, click **Up** until **Duration** is under the Groups heading as shown in Figure 11, and then click **OK**.

![Organize Columns](image)

Figure 11. You can group the results by Duration by using the Organize Columns dialog box.
8. In the Trace Properties window, click **Run** to start the trace.

9. Return to the **Browser** subtab in BIDS and generate some queries against the Adventure Works cube by dragging and dropping measures and dimensions onto the pivot table surface.

10. Return to the Profile and then click the **Stop Trace** (red) button on the toolbar. Scroll through the data that Profiler captured. Expand some of the groups and highlight a Query End row to view the query generated by the browser. The longest running queries are at the bottom of the output. Figure 12 shows part of a sample trace.

![SQL Server Profiler](Image)

<table>
<thead>
<tr>
<th>Duration</th>
<th>EventSubclass</th>
<th>EventId</th>
<th>EventClass</th>
<th>EventProperties</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Query Subcube</td>
<td>2</td>
<td>Non-cache data</td>
<td>00000009,000,000,0001000000000000000...</td>
</tr>
<tr>
<td>31</td>
<td>Query Subcube</td>
<td>2</td>
<td>Non-cache data</td>
<td>00000009,000,000,0001000000000000000...</td>
</tr>
<tr>
<td>31</td>
<td>Query Subcube</td>
<td>2</td>
<td>Non-cache data</td>
<td>00000009,000,000,0001000000000000000...</td>
</tr>
<tr>
<td>31</td>
<td>Query Subcube</td>
<td>2</td>
<td>Non-cache data</td>
<td>00000009,000,000,0001000000000000000...</td>
</tr>
</tbody>
</table>

Figure 12. You can use SQL Profiler to trace activity on SSAS.

If you click on (or filter for) **EventSubclass type 0**, you can see the detail of the MDX query that you selected.

In addition, you can extract specific types of SSAS items from a trace. To do this, select **File|Export|Extract SQL Server Analysis Service Events**. This is a simple way to “grab” all of the MDX queries that a particular trace captured.
Both of these techniques can help you determine which MDX queries would best benefit from added aggregations.

TIP: You can rearrange the order of the columns in Profiler by simply dragging and dropping them in the UI.

11. Close Profiler and BIDS without saving the changes.
Advanced Storage: MOLAP, HOLAP, or ROLAP

If storing cube data and metadata entirely in MOLAP results in the best query performance, why would you want to use any other type of storage?

Typically, you will be interested in these options (HOLAP or ROLAP) only if your cube is relatively large—250 GB or greater, for example—or if storage space is generally problematic for you. Other reasons might include cube processing times that are excessive for your particular business needs, or specific business rules or requirements that regulate the storage of data.

TIP: Before changing storage settings for a cube, you may want to review aggregation design. Be sure not to over-aggregate your cube (as discussed in the “Customizing Aggregations” section). Over-aggregation results in excessive disk storage space and slow cube processing times.

Using Partitions with Advanced Storage Options

Although you can change the storage type for an entire cube, it is more common to change the physical storage design for a portion (or partition) of the cube. (Remember that the ability to create partitions in a cube requires the Enterprise Edition of SSAS.)

To view the source of a partition in BIDS, click the Partitions subtab on the cube design surface, and then click in the Source column cell for the partition you are interested in. A build button will appear; click that button and you'll be able to verify the source of the partition. The options that you can configure vary depending on the binding type that you choose.

You can configure the Binding type field to Table Binding or Query Binding. For a Query Binding, you must select the Data Source and a query that defines the columns and rows to be included in the partition. If you are creating more than one partition for the Measure Group, you must define a WHERE clause that restricts the rows returned so that they are unique to this partition. Figure 13 shows the Query Binding for the first Reseller Sales partition from the Adventure Works sample cube.
Cube Storage and Aggregation

Figure 13. SSAS cube partitions that are query bound must specify a WHERE clause to make the rowset for the partition unique.

For a table bound partition, you will configure the data source or data source view (DSV) to look in. You must also select the source table from the available tables list. You can use the Filter tables box and the Find tables button to help locate the required table more easily in a large DSV.

You can adjust the SSAS storage method on a partition-by-partition basis. To do this, select the partition of interest on the Partitions subtab, and then click the Storage Settings link below it. This opens the Partition Storage Settings dialog box (see Figure 14). Here you can adjust the storage mode (and other advanced storage settings) by dragging the slider to one of the listed settings, or by selecting Custom setting and then configuring the required options.
Figure 14. You can adjust the storage settings by dragging the slider in the Partition Storage Settings dialog box.

Although the slider provides a brief explanation of the settings, you probably need a more complete explanation to effectively select something other than the default of simple MOLAP. Note that Proactive Caching is enabled by default for all storage modes other than the default MOLAP. (Proactive Caching, which requires the Enterprise Edition, is discussed in the next section of this chapter.) Here’s a list that describes the impact of changing the default setting in the Measure Group (or Partition) Storage Settings - <measure group name> dialog box.

- **MOLAP** (default): Source data is copied; metadata and aggregations are stored in MOLAP format on SSAS; proactive caching is not used.

- **MOLAP** (non-default): Source data is copied; metadata and aggregations are stored in MOLAP format on SSAS; proactive caching is enabled. This includes scheduled, automatic, medium, and low-latency MOLAP.

- **HOLAP**: Source data is *not* copied; metadata and aggregations are stored in MOLAP format on SSAS; proactive caching is enabled. This method is often used for archive partitions or in situations where a large amount of historical data is queried infrequently.

- **ROLAP** (for a cube): Source data is *not* copied, but resides in the relational database source; aggregations are also stored in the source database (for a dimension); proactive caching is enabled for the real-time ROLAP setting.

**NOTE** ROLAP storage is most often used when business needs warrant a “near real-time” OLAP solution—more about this in the section on proactive caching.
ROLAP Dimensions

Do you have any really huge dimensions, with millions and millions of members? If your business case includes this, you can choose to use ROLAP dimensions.

You typically use this feature only for dimensions with a huge number of members; an example might be the customer dimension.

**NOTE** Although ROLAP dimensions increase the flexibility of your cube, they’re not often used in production BI solutions. The reason is that any queries to the relational database source will always be significantly slower than queries to MOLAP data or metadata.

Using ROLAP dimension storage means that the dimensional attribute values will *not* be copied to and stored on SSAS; rather, they will be retrieved directly from the relational source table or tables. To set a dimension as a ROLAP dimension, you’ll use the properties window for that dimension. There you’ll change the StorageMode property from the default MOLAP to ROLAP.

**WARNING!** When using a ROLAP dimension in a cube that uses MOLAP storage, any schema change to the relational database must be immediately followed by an immediate processing of the cube.
Implementing Proactive Caching

Traditional OLAP solutions included some latency because the source data often had to undergo an ETL (Extract, Transform, and Load) process before it was made available to the OLAP cube(s). The typical latency was one day. This restriction was unacceptable to some businesses.

Microsoft developed the Proactive Caching feature set to overcome this restriction. The concept is to produce “near real-time” OLAP. Like other advanced features, you can only deploy proactive caching to servers using the Enterprise Edition. Another way to understand this concept is to think that configuring proactive caching is the method by which you manage the MOLAP cache; this is shown (conceptually) in Figure 15.

NOTE One of the reasons that queries in SSAS are so much faster than queries to a relational data store is the use of the MOLAP cache. When you consider configuring manual settings to manage cache refreshes, you can use the Analysis Services: Query object in System Monitor to measure which queries are being answered by cache hits vs. disk calls.

![Figure 15. Enabling Proactive Caching allows you to manually manage the MOLAP cache refresh and update settings.](image)

It’s important to understand that the nearer you get to real-time results for your queries, the more overhead you add to the system. That’s why SSAS has six standard options that you can choose from when configuring proactive caching (using the Partition Storage Settings dialog box). In addition, you have the possibility of still more finely grained control via the Storage Options dialog box in this section, and still more control by modifying individual values in the Properties window for any particular measure group.
WARNING! Proactive caching is not for every BI solution. Using it effectively necessitates that you either read your OLTP data directly as the source for your cube, or you read a replicated copy of your data. Another option is to read your OLTP data using the new snapshot isolation level. To use any of these other options, your source data must be very clean. If you need to cleanse, validate, and/or consolidate during ETL processing, proactive caching is not the best choice for your solution.

Let’s start with a more complete explanation of the choices available in the Partition Storage Settings dialog box (shown in Figure 14) as they relate to proactive caching settings. The Measure Group Storage Settings dialog box presents the same options, simply applying them to a particular measure group, rather than a partition.

TIP: When you change the storage settings for a measure group, it does not affect any of the current partition settings. It will only affect the default setting for new partitions created in that measure group, in the future.

The first choice you’ll make is whether to use MOLAP, HOLAP, or ROLAP data storage for your cube. In most cases, due to the superior query performance, you’ll select some version of MOLAP. The proactive caching configuration choices for MOLAP are as follows:

- **Scheduled MOLAP**: Updates the MOLAP cache per a schedule (whether the source data changes or not); the default is once daily. This sets the rebuild interval to one day.
- **Automatic MOLAP**: Updates the cache whenever the source data changes. It configures the silence interval (refer to the TIP box) to 10 seconds and a silence override interval of 10 minutes.
- **Medium-latency MOLAP**: In this setting, the outdated caches are dropped periodically (the default is a latency period of 4 hours). The cache is updated when data changes (defaults are a silence interval of 10 seconds and a 10-minute silence override interval). Additionally, the *Bring online immediately* setting is enabled, allowing users to query the data from the relational data source while the MOLAP cache is being rebuilt. If this option were not selected, queries are not processed until the cache rebuild is completed.
- **Low-latency MOLAP**: In this setting, outdated caches are dropped periodically (the default is a latency period of 30 minutes). The cache is updated when data changes (defaults are a silence interval of 10 seconds and a 10-minute silence override interval). Additionally, the *Bring online immediately* setting is enabled.
Implementing Proactive Caching

TIP: To understand the silence interval property, think of this question: “How long should the cache wait to refresh itself if there are no changes to the source data?” To understand the silence override interval property, think of the question “What is the maximum amount of time that the cache should wait to start rebuilding itself after having received a notification that the source data has been modified?”

If you select HOLAP or ROLAP, the following standard proactive caching settings are available:

- **HOLAP:** In this setting, outdated caches are dropped immediately (configures the latency period to 0 seconds). The cache is updated when data changes (defaults are a silence interval of 0 seconds and no silence override interval).

- **ROLAP:** In this setting, the cube is always in ROLAP mode, and all updates to the source data are immediately reflected in the query results—the latency period is set to 0 seconds.

Selecting the Options button in the Storage Settings dialog box allows you to adjust the cache settings, options, and notification values. Figure 16 shows these options.

![Storage Options](image)

Figure 16. The Storage Options dialog box presents options dependent on the storage mode you’ve selected: MOLAP, HOLAP, or ROLAP.
If you are using ROLAP or MOLAP with the ProactiveCaching property enabled, the OnlineMode property set to Immediate, and if your partitions align with a frequently queried “slice” of your cube, (for example the year 2004), you can configure the Slice property of the partition to improve performance. The Slice property tells the query engine what data is stored on a partition, so it does not have to query the partition to determine what data it holds. For example, suppose you have two partitions: one holds 2003 data and the other holds 2004 data. If you configure the Slice property and a query requests data for 2003, the server knows that the data is on the first partition and does not have to query the other partition. For MOLAP partitions without proactive caching, this happens automatically, so you do not need to manually configure the Slice property.

Notification Settings for Proactive Caching

You can adjust the notification settings (regarding data changes in the base OLTP store) by using the Notifications subtab in the Storage Options dialog box. Three types of notifications are possible using this property sheet:

- **SQL Server**: For this option, you specify tracking tables in the relational source database. This option uses trace events and requires that either the service account for SSAS or the data source have ALTER TRACE or system administrator privileges.

- **Client Initiated**: For SQL Server, you specify tracking tables in the relational source database. Use this option when notification of changes will be sent from a client application to SSAS.

- **Scheduled Polling**: For this option, you specify the polling interval time value and whether you want to enable incremental updates, and add at least one polling query. Each polling query (or queries) is also associated with a particular tracking table.
Using Partitions: Relational or SSAS

If you are working with the Enterprise Edition of SQL Server, you now have the ability to do relational table partitioning in your relational database source. This strategy can complement partitioning that you choose to do using SSAS (that is, cube partitions), or you can choose to partition only on the relational side. The consideration then becomes, which type of partitioning is appropriate for your BI solution?

Relational Table Partitioning in SQL Server 2008

Table partitioning is the ability to position data from one table in different physical locations (disks), while that data appears to originate from the same logical table. This simplifies management of very large databases (VLDBs)—in particular, management of very large tables. The large tables that are of interest here are, of course, fact tables.

It is not uncommon for fact tables to contain millions or tens of millions of rows. Relational table partitioning can simplify administrative tasks and general management of these often large data sources.

How to Implement OLTP Partitioning

Although relational table partitioning is relatively simple, it requires several steps to implement.

TIP: For more information about OLTP partitioning, see Partitioned Tables and Indexes in SQL Server Books Online.

Here’s a conceptual overview of the technique:

1. Identify the tables that are the best candidates for partitioning. For OLAP projects, as mentioned, this will generally be the fact tables.

2. Identify the value (or column) to be used for partitioning; this is usually a date field. You must implement a unique constraint on this column of the tables that will participate in partitioning.

3. Implement the physical architecture necessary to support partitioning; i.e., install the physical disks.
4. Create **Filegroups** in the database for each of the new physical disks or arrays.

5. Create .ndf files (or **secondary database files**) for the SQL Server 2008 database where the tables to be partitioned are contained and associate these .ndf files with the Filegroups you created in Step 4.

6. Create a **partition function**. This creates the “buckets” to distribute the sections of the table into, usually by date range; i.e., from xxx to yyy date—most often monthly or annually.

7. Create a **partition scheme**. This associates the buckets you created previously to a list of Filegroups, one filegroup for each time period—month or year.

8. Create the **table** (this is usually the fact table) on the partition scheme that you created earlier; this “splits” the table into the buckets you created.

### Other Capabilities of OLAP Partitions

One other consideration in the world of partitions relates to SSAS cube partitions. With the Enterprise Edition of SSAS, it is also possible to define cube partitions as **local** (the default) or **remote**.

The primary reason to consider using remote partitions is to do a kind of “load balancing” in the SSAS environment. The following additional considerations apply to remote partitions.

Remote partitions can use MOLAP, HOLAP, or ROLAP storage. They can also use proactive caching. Remote partitions store information on the remote server. Servers holding remote partitions must have SSAS installed and must be at the same service pack level as the server holding the SSAS database. The type of information that they store remotely depends on the storage method you select.

- **MOLAP**: Data and aggregations for the remote partition are stored on the remote server.
- **HOLAP**: Aggregations for the remote partition are stored on the remote server.
- **ROLAP**: Nothing is stored on the remote server.
Cube and Dimension Processing Options

Now that you’ve learned about storage, aggregations, partitions, and caching, it’s time to review cube and dimension processing option types. Dimensions must be completely and correctly processed either before, or at the beginning of a cube process. The best way to understand this is to remember that dimensional data is the metadata or the structure of the cube itself—so the metadata must be available before the data can be loaded into the cube.

NOTE During development you will usually perform a Full Process type of cube processing whenever you need to view the results of a change that you’ve made. This option completely erases and rebuilds all data and metadata. For a few customers, this simple method of updating the cube can be used in production as well. What happens there is a complete overwrite on rebuild. This is only practical for small cubes—a maximum size of a couple of GBs.

Usually you will choose to use the more granular processing options in production situations. The first consideration is the ability to separate processing of dimensions from the cube.

In Analysis Services, you can process SSAS databases, cubes, measure groups, partitions, dimensions, and data mining structures and models. A set of processing options is provided for each object type. To minimize processing time, you can choose to process an object that you have modified along with other objects affected by that modification.

Here’s a review of the process for processing via BIDS: simply right-click on the cube name in the Solution Explorer in BIDS, and then select Process. You’ll see the dialog box shown in Figure 17.

![Figure 17. The Process Partition dialog box in BIDS provides you with several processing options.](image-url)
A more complete explanation follows of the available selections for process options for both cubes and dimensions. Certain options are available only for cubes or for dimensions, as noted in the following list:

- **Default**: SSAS detects the current state of the cube or dimension, and then does the type of processing (full or incremental) that is necessary to return the cube or dimension to a completely processed state.

- **Full**: SSAS completely reprocesses the cube or dimension. In the case of a cube, this includes all the objects within it; for example, dimensions. (Full process is required when a structural change has been made to a cube or dimension.) In the case of a dimension, reprocessing would occur, for example, when an attribute hierarchy is added, deleted, or renamed.

- **Data**: SSAS processes data only (including dropping and repopulating existing data) and does not build any aggregations or indexes.

- **Unprocess**: SSAS drops the data in the cube or dimension. If there are any lower-level dependent objects—for example, dimensions in a cube—the data for those objects is dropped as well.

- **Index**: SSAS creates or rebuilds indexes and aggregations for all processed partitions. This option results in an error on unprocessed objects.

- **Structure** (cubes and mining structures only): For an unprocessed cube, SSAS processes the cubes and if necessary, any contained dimensions. Otherwise, it simply creates cube definitions. If applied to a mining structure, it populates the mining structure, but does not process any mining models associated with it.

- **Clear Structure** (mining structures only): Removes all training data from a mining structure.

- **Incremental** (measure groups and partitions only): SSAS adds newly available fact data and processes to only the affected partitions. This is the most common option used in day-to-day production.

- **Script Cache** (cubes only): Deprecated feature, which means that is should not to be used in new code and should be removed from existing code as soon as possible.

- **Update** (dimensions only): SSAS forces an update of dimension attribute values. For flexible relationships, aggregations and indexes on related partitions will be dropped, possibly forcing a reread of all data to update object attributes.
TIP: Aggregation process behavior in dimensions depends on the RelationshipType property of the attribute relationships in the cube. After an incremental update, aggregations for flexible relationships are dropped and must be manually reprocessed. For rigid relationships, the aggregations are persisted after an incremental update. This is why it is important to define relationships as rigid when the relationships between the levels are not going to change. In addition, if you set the dimension processing mode (for the dimension) to LazyAggreggations, aggregations are reprocessed as a background task and end users can browse the cube while this processing occurs.

You can take the following optimization step to reduce processing times for your dimensions: Turn off the AttributeHierarchyOptimizedState property for dimensional attributes that end users view infrequently. This is especially important to do for attributes with a high portion of unique values. An example would be an e-mail address field. Although a user may want to drill through to see this attribute, they would not want to pivot the cube upon this value.

To do this, set the property value to NotOptimized, so SSAS will not create supplementary indexes (those that are created by default) for this particular attribute on dimension or cube process. This can result in slower query times, so be careful with changing this property. Figure 18 shows this setting.

![Properties](image)

Figure 18. Setting the AttributeHierarchyOptimizedState to NotOptimized for certain attributes can improve dimension processing times.
The final consideration for processing cubes and dimensions is whether you need to adjust any of the processing options. You access these options with the Change Settings button in the Process Cube dialog box. Here you have two tabs to work with. On the first tab, **Processing Options**, you can set the following values:

- **Parallel or Sequential processing:**
  - If parallel, the maximum number of parallel tasks.
  - If sequential, a transaction mode of one or multiple transactions.

- **Writeback table:** Choose between three values:
  - **Create**: Create the writeback table if one doesn’t exist. An error occurs if the table already exists.
  - **Create always**: Create the writeback table if one doesn’t exist. Overwrite an existing table if one does exist.
  - **Use existing**: Use the existing table if one exists. An error occurs if the table does not exist.

- **Process affected objects**: Turn this off or on.

The second tab, **Dimension key errors**, allows you to configure the behavior of errors during processing. Figure 19 shows this tab of the property sheet.

![Change Settings](image)

Figure 19. The Dimension key errors tab in the Change Settings dialog box allows you to specify error configurations for processing.
Although you have probably processed test cubes before reading this chapter, you may have gained a bit more insight into what actually happens when you run the process action. Figure 20 shows the output from the Process Cube dialog box. You can also direct this process information to a log file. In production, you would normally automate the cube/dimension processing via SSIS packages, using the Analysis Services Processing tasks that are included with SSIS.

Figure 20. The Process Progress dialog box provides a complete list of each processing step.

If you need to troubleshoot processing errors, you can view and copy the actual SELECT statement issued by SSAS by double-clicking the query in question. This will open the query in a small window as shown in Figure 21.

Figure 21. You can view and copy the script issued by SSAS during processing.
Summary

- For a fair number of BI projects, the default storage option—MOLAP with no aggregations—is adequate.

- Although HOLAP and ROLAP can be more efficient in terms of storage space, queries to these types of structures (cubes or partitions) execute much more slowly than queries directed to MOLAP structures.

- It is quite common to add aggregations to your cubes. Use the tools included in BIDS to help design and implement aggregations appropriately. Refrain from over-aggregating, as this increases cube processing times and storage space, and does not improve query response times.

- If your BI project includes a very large amount of data (more than 250 GB) you may want to use either partitions or a non-default storage method (HOLAP or ROLAP) to improve data management or cube processing times.

- Cube partitions require the Enterprise Edition of SSAS.

- Use ROLAP dimensions when the underlying data is constantly changing and you do not want to have the overhead associated with dimension processing. Queries to ROLAP dimensions run much more slowly than queries to MOLAP or HOLAP dimensions.

- If your business requirements necessitate “near real-time” OLAP, you can implement proactive caching, which is available only in the Enterprise Edition. To be able to use this feature, your relational database source data must be very clean.

- The Analysis Services Processing task is integrated into SSIS so that you can process by using SSIS packages.
Questions

1. True/False: The default storage mode for SSAS is MOLAP with 20% aggregations.
2. What is an aggregation?
3. What is a ROLAP dimension?
4. True/False: All versions of SSAS support the creation of cube partitions.
5. What is proactive caching?
6. In production situations, which SQL Server tool is generally used to automate cube processing?
Answers

1. True/False: The default storage mode for SSAS is MOLAP with 20% aggregations.
   False. The default storage mode is MOLAP. However, 0% aggregations is also the default. It is most typical to add aggregations, usually at the 20-30% level.

2. What is an aggregation?
   An OLAP aggregation is a saved, calculated intersection of fact values. Aggregations are created at higher levels in dimensional hierarchies than the initial load of the fact table. An example is a sales amount by day that is aggregated by week.

3. What is a ROLAP dimension?
   A ROLAP dimension is an OLAP dimension that is set for ROLAP storage. What this means is that the dimensional attribute values are read directly from the relational database source, rather than being loaded (as MOLAP data) into SSAS. ROLAP dimensions are typically used for very large, (i.e., millions of members) dimensions.

4. True/False: All versions of SSAS support the creation of cube partitions.
   False. The only version of SSAS that supports the creation of cube partitions is the Enterprise Edition.

5. What is proactive caching?
   Proactive caching is the ability to provide near real-time data to users of an OLAP cube by configuring the update behavior of the MOLAP cache.

6. In production situations, which SQL Server tool is generally used to automate cube processing?
   In production situations, cube and dimension processing is generally automated using custom SSIS packages that include the OLAP cube and dimension processing task.
Lab 6:
Cube Storage and Aggregation
Lab 6 Overview

In this lab you’ll learn how to use BIDS to design and implement storage for a cube.

To complete this lab, you’ll need to work through two exercises:

- Add a Cube Partition
- Design Aggregations and Process the Cube

There are starter files associated with this lab in the CubeStorageLab folder. The solution files are in the CubeStorageLabCompleted folder.
Add a Cube Partition

Objective

In this exercise, you’ll use the BIDS Add a Partition Wizard to create a cube partition.

Things to Consider

The ability to create SSAS cube partitions requires the Enterprise Edition of SSAS. It is typical to create OLAP partitions to reduce the amount of time it takes to process portions of a cube. Another reason is to be able to define different physical locations for parts of the cube.

Step-by-Step Instructions

1. Open the Cube storage CubeStorageLab.sln in BIDS.

2. Double-click the CubeStorage.cube file under the Cubes folder in the Solution Explorer window to open the Cube Designer window in BIDS.

3. Click the Partitions subtab in the Cube Designer to select it. Note that the cube contains two partitions, one for each measure group.

4. Click the New Partition link under the Dim Employee partition. The Partition Wizard opens; click Next. Click DimEmployee in the Available tables section and note that you cannot create a partition because the entire table is bound to an existing partition. Click Cancel.

5. Click below the Source column header in the Dim Employee section to select it. A build button will appear. Click the build button to open the Partition Source dialog box, as shown in Figure 22.

![Partition Source - Dim Employee](image)

Figure 22. The Dim Employee partition is currently bound to an entire table.
6. Change the value of the Binding type to **Query Binding** and add a where clause to the end of the query: `WHERE EmployeeKey < 200`. Click **OK**.

7. Click the link named **New Partition** under the Dim Employee partition to open the Add a Partition Wizard and then click **Next**.

8. On the Specify Source Information page, click the check box next to `dbo.DimEmployee` in the Available tables section and then click **Next**.

9. On the Restrict Rows page, check the box to **Specify a Query**, and then add a where clause to the end of the query: `WHERE EmployeeKey >= 200`. Click **Next**.

10. On the Processing and Storage Locations page, accept the default setting of **Current Server instance** and then click **Next**.

11. On the Completing the Wizard page, accept the default partition name of **Dim Employee 1** and then click the radio button to **design aggregations later** and then click **Finish**. You’ll note that the partition has been designed with the default storage and aggregation settings—MOLAP with 0% aggregations.

12. Make sure that the new partition (**Dim Employee 1**) is selected in the list of partitions for Dim Employee and then click the link named **Storage Settings** to open the Partition Storage Settings dialog box.

**NOTE**  In a production environment, your partition names should describe the subset of data included in the partition, such as Dim Employee EmpKey over 200.

13. Drag the slider from MOLAP to **Low-latency MOLAP**. Note that after you do that, BIDS displays a warning message, shown in Figure 23.

![Warning](image)

Figure 23. Most non-default storage options that are bound to partitions based on queries have additional requirements.

14. Drag the slider to **Scheduled MOLAP**. Note that the warning goes away. Click **OK** to close the dialog box.

15. Leave BIDS open for the next exercise.
Design Aggregations and Process the Cube

Objective

In this exercise, you’ll design aggregations and storage for the cube partition you created in the first exercise.

Things to Consider

It is quite typical to create partitions to take advantage of the fact that each partition can have its own storage mode and aggregation percentage. Usually data is partitioned by time (often year or month); the most current data is usually aggregated at a higher percentage.

This lab is dependent on the successful completion of the first exercise.

Step-by-Step Instructions

1. Continue to work with the cube from the last exercise in BIDS (or use BIDS to open CubeStorageLab.sln in the CubeStorageLab_ex02 folder). Change to the Aggregations subtab of the CubeStorage.cube designer.

2. Right-click the Dim Employee measure group, and select Design Aggregations to open the Aggregation Design Wizard, and then click Next on the welcome screen.

3. On the Select Partitions to Modify page, select the Dim Employee partition and then click Next.

4. On the Review Aggregation Usage page, review the objects listed and click Next to accept the default options.

5. On the Specify Object Counts page, click the Count button to have BIDS count the number of members. After the count is complete, click on the plus sign next to Dim Employees to view the count for each attribute, and then click Next.
6. On the Set Aggregation Options page, click the radio button next to **Performance gain reaches** and verify that 30% is selected, and then click the **Start** button. After the wizard is done designing aggregations, click **Next**.

7. On the Completing the Wizard page, name the aggregation design **30 Percent Optimization**, verify that the radio button next to **Save the aggregations but do not process them** is selected, and then click **Finish**.

8. Review the information presented on the Aggregations subtab and notice that the Dim Employee partition is using the new aggregation design that you just created, while the Dim Employee 1 partition is still unassigned, meaning it is using the default of zero aggregations.

9. In the Aggregations design area on the Aggregations subtab, right-click on the **Dim Employee 1** partition, and select **Assign Aggregation Design**.

10. In the Assign Aggregation Design window, select **30 Percent Optimization** in the Aggregation designs drop-down list, select **Dim Employee 1** in the Destination partitions area, and then click **OK**. Review the changes in the aggregations design area.

11. On the Aggregations subtab toolbar, click the **Advanced View** icon.

12. At the top of the Advanced window, select **30 Percent Optimization** in the Aggregation Design drop-down box. Expand Dim Employee in the main design window and review the aggregation that was created for Department Name.

13. Change to the **Partitions** subtab. In the Dim Employee partition section, click on the **Dim Employee 1** partition in the list to select it.

14. Click on the link named **Storage Settings** to open the Partition Storage Settings window.

15. In the Partition Storage Settings window, accept the previously defined storage setting of Scheduled MOLAP and click **Options**.

16. On the Storage Options page, on the **General** tab, note that the Scheduled MOLAP defaults to setting the MOLAP cache to **Update daily**. Change the Rebuild Interval value to **1 Hours**, and click **OK**.

17. Click **OK** to close the Partition Storage Settings window.

18. Save your solution and close BIDS.