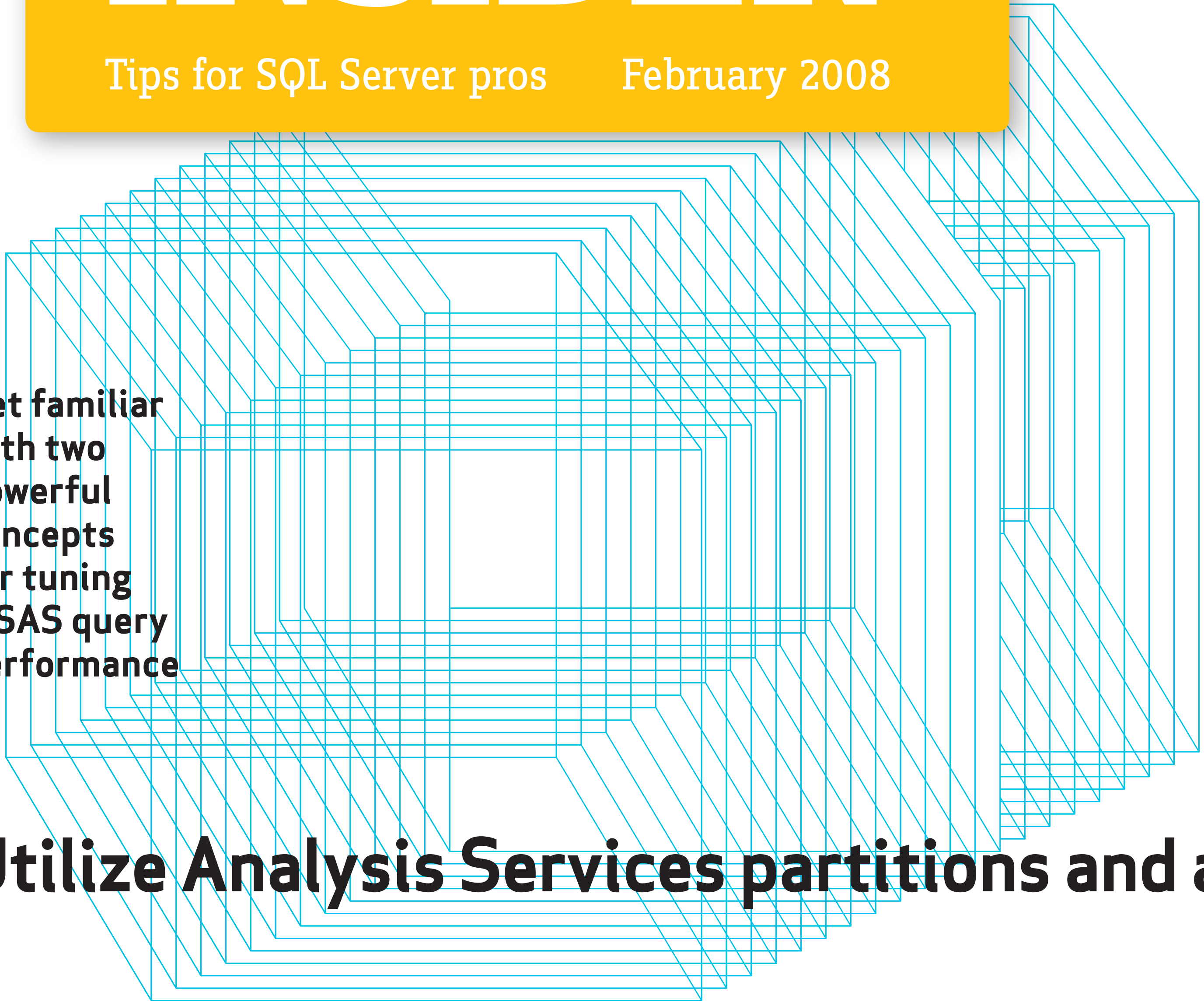
SQL Server INSIDER

Tips for SQL Server pros February 2008



Get familiar with two powerful concepts for tuning MSAS query performance

Utilize Analysis Services partitions and aggregations

Utilize Analysis Services partitions and aggregations

BY BAYA PAVLIASHVILI

Microsoft Analysis Services, or MSAS, allows you to build basic multi-dimensional structures, dimensions and cubes easily. If you have fact tables with 50,000 rows or so, your cube can perform fine without any tuning efforts. Once your cubes grow to a decent size, however, and your fact tables have hundreds of millions of rows, you should augment the trivial functionality so your application can continue providing satisfactory response times for user queries.

There are two powerful concepts for tuning MSAS query performance: partitions and aggregations. The primary focus of this article is MSAS 2005; the same concepts apply to other versions of the software as well, but their implementation details could change from one release to the next.

Before reviewing the methods for using partitions and aggregations effectively, it is useful to become familiar with nomenclature

BAYA PAVLIASHVILI

is a database consultant helping his customers develop highly available and scalable applications with SQL Server and Analysis Services. Throughout his career he has managed database administrator teams and databases of terabyte caliber. Pavliashvili's primary areas of expertise are performance tuning, replication and data warehousing. He can be reached at baya@bayasqlconsulting.com.

related to measure groups and to consider the typical cube lifecycle. A data warehouse analytics project by its very nature is never "complete." As with many technology efforts, building an MSAS system involves gathering requirements, building databases and perhaps coding a custom front-end application. But after the initial development phase, measure groups and cubes prove to be somewhat different from other projects.

Once you build an MSAS measure group, you must process it — meaning that you have to populate it with data residing in a relational database. For those of you who are new to the latest release of MSAS, each cube can include data based on multiple fact tables; a measure group is a subset of a cube that references a single fact table. Each cube can have one or multiple measure groups. So essentially, a measure group in MSAS 2005 is equivalent to a cube in previous versions of MSAS. Processing also involves writing data into multi-dimensional files — if you use multi-dimensional OLAP — and building bitmap indexes and aggregations.

Aggregations are pre-calculated summary values that Microsoft Analysis Services could use at query execution time, rather than scanning the partition files and deriving summary values on the fly. By default, each measure group is created with a single partition that is built upon a relational fact table. However, you can build additional partitions and change existing partitions as needed. The same fact table can be used for multiple partitions and, alternatively, each partition can be bound to a different fact table or view. MSAS 2005 also allows you to bind partition definitions to a query.

Measure groups are equivalent to cubes in earlier versions of Analysis Services. In Analysis Services 2005, a cube can have one or multiple measure groups.

As you add new data to the relational warehouse, you must re-process the measure group to reflect data changes. Processing measure groups based on fact tables with millions or billions of rows can take a long time. Fortunately, if you have multiple partitions within the measure group, you might not have to process all partitions. Each partition is independent of all other partitions. It can have a different fact table, different set of aggregations and a separate process. MSAS also allows you to process multiple partitions in parallel.

Historical data in a data warehouse is normally static. That is you're probably not interested in making changes to records that were created 15 years ago. If indeed this is the case (meaning you don't want to make changes to old data), you would need to fully process only newly created partitions and perhaps incrementally process some of the existing partitions.

On the other hand, a measure group is a continuously evolving entity that requires considerable effort, even after it is exposed to the end users. You might change the structure of your measure group — perhaps by adding or removing dimensions — or you might discover data loaded into the historical partitions was inaccurate and must be reloaded. It is completely possible you might need to reprocess the entire measure group several times.

After you create the initial set of aggregations, you might find they're not sufficient for some frequently executed queries. In that case, options exist to design additional aggregations or change the partitioning strategy. As your analysts become savvier, their querying patterns are likely to change, so you might need to change your measure groups accordingly.

As you can tell, delivering an application to the users is only a start. An Analysis Services database is a continually growing entity. You have much work to do to ensure your solution performs well for a wide variety of queries that users might submit to Analysis Services.

Analysis Services partitions allow you to apply the divide-and-conquer principle to your data.

USE PARTITIONS EFFECTIVELY

Analysis Services partitions allow you to apply the divide-and-conquer principle to your data. When your users submit Multi Dimensional eXpressions, known as MDX, queries to your cube, MSAS examines the storage engine cache. If the data isn't available in cache, then MSAS examines any existing aggregations to see if they can be helpful in resolving the query.

Finally, if neither the aggregations nor cache can satisfy the query, MSAS scans partition data files. Scanning a small partition file will be faster than scanning a large file or multiple files. Your goal then is to partition your measure group so that each query scans only a single partition or few

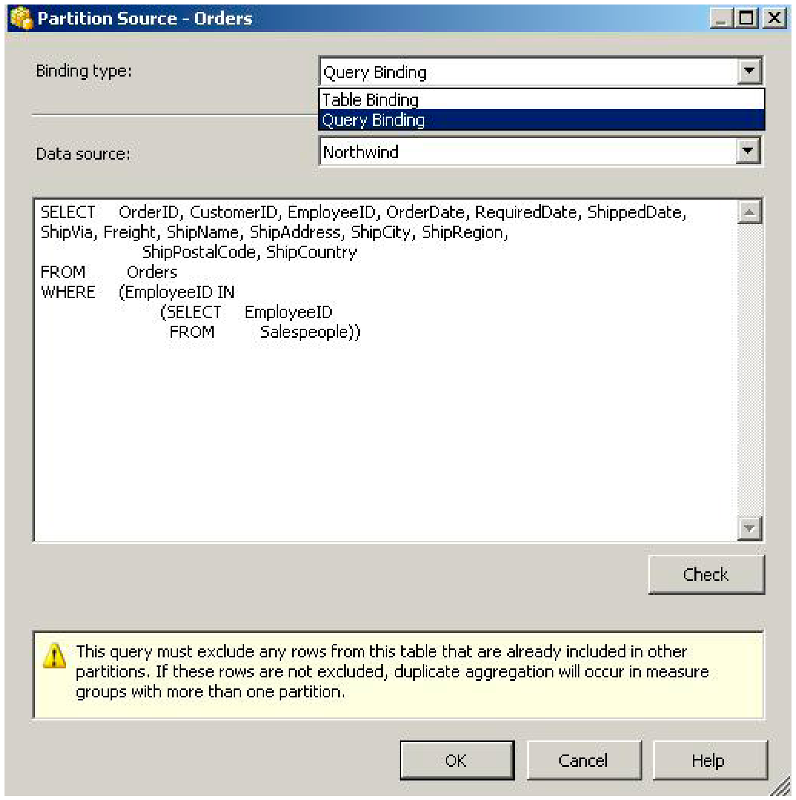
Partitioning a measure group by month instead of year could make queries dramatically faster.

partitions at most.

So how do you partition a measure group effectively? The natural inclination is to use a date or periodicity dimension for partitioning. Your users might ask you to show them all sales for the year 2007 and compare them to corresponding sales in 2006. If you partition your measure group by year, you might have just a few partitions.

So far, so good. But what if query performance is still suboptimal after partitioning by year? Indeed, if the fact table contains millions of records for each year, scanning even a couple of these partition files might take a while. No problem. If you can convince your users to examine data for one month or a few months at a time, you can partition the measure group by month instead of by year. This could make queries dramatically faster. Normally, you process only a single partition or a few partitions at a time. But as their number grows, so does the complexity involved in managing partitions.

You can edit partition definitions through Business Intelligence Development Studio, or BIDS.



```
SELECT    OrderID, CustomerID, EmployeeID, OrderDate, RequiredDate, ShippedDate,
ShipVia, Freight, ShipName, ShipAddress, ShipCity, ShipRegion,
                ShipPostalCode, ShipCountry
FROM      Orders
WHERE     (EmployeeID IN
               (SELECT    EmployeeID
                FROM       Salespeople))
```

This query must exclude any rows from this table that are already included in other partitions. If these rows are not excluded, duplicate aggregation will occur in measure groups with more than one partition.

FIGURE 1: *Use either table-binding or query-binding to define the partition source.*

Navigate to the Partitions tab, click a partition you're interested in and examine the source property. Here you can specify the fact table or view which the partition should be based upon.

You could also specify the query defining the partition. As shown in **FIGURE 1**, you can use either table-binding or query-binding to define the partition source.

How does MSAS know which partition file to scan for each query? Each partition has a property called a "partition slice," where you can specify the attributes associated with the partition. For example, you could define the partition slice for calendar year 2007 as follows:

```
[Date] [calendar] [calendar year] [2007]
```

Microsoft Analysis Services doesn't check your data to ensure the partition slice definition is valid. Cube architects need to ensure that the slice property correctly identifies the subset of cube data stored in each partition.

It is essential to realize you're not limited to date or periodicity dimensions for partitioning. In fact, you can use any dimension and any attribute in the measure group for defining your partitions.

Suppose you use SQL Profiler to review the queries your users execute against the measure group and you find that they're almost always specific to a particular region or country. You could further partition the cube by the "country" attribute. This way, queries retrieving data for 2007 USA sales go against one partition file, and queries examining Canadian sales for the same year scan another partition file.

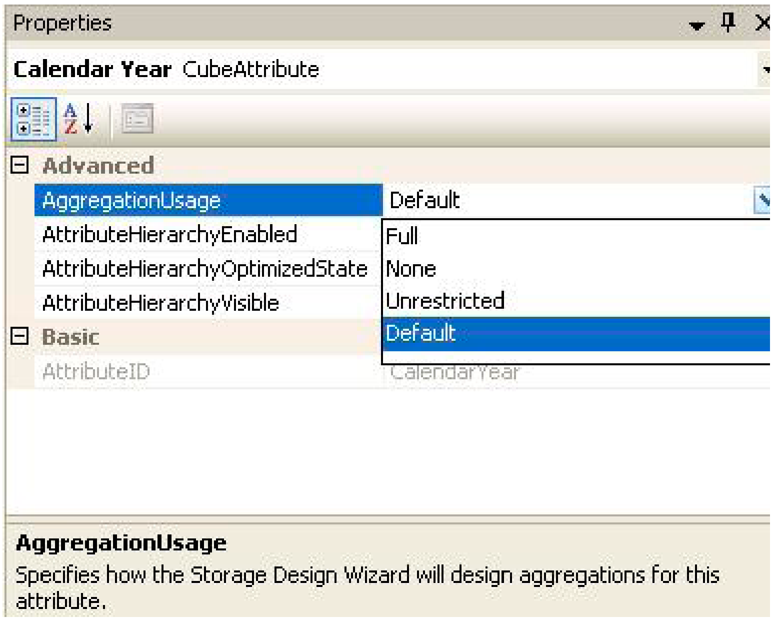
It is essential to realize you're not limited to date or periodicity dimensions for partitioning. You can use any dimension and any attribute to define partitions in the measure group.

With such a partitioning strategy, region-specific queries could be considerably faster and queries examining sales across countries will not be affected. A minor drawback of splitting partitions is that you might have to modify your code so it processes multiple partitions each month — or whatever frequency you use for processing partitions — as opposed to only processing a single partition.

In general, it's good to limit each partition to fewer than 20 million rows. If your cube has many dimensions, then you should examine the size of partition files. Typically you should try to keep partition files under 500 MB.

BUILDING USEFUL AGGREGATIONS

MSAS has storage design and usage-based optimization wizards for building aggregations. Both wizards use a sophisticated algorithm for deciding which aggregations would be most beneficial for each measure group.


Properties

Calendar Year CubeAttribute

Advanced
- AggregationUsage: Default (options: Full, None, Unrestricted, Default)
- AttributeHierarchyEnabled
- AttributeHierarchyOptimizedState
- AttributeHierarchyVisible

Basic
- AttributeID: CalendarYear

AggregationUsage
Specifies how the Storage Design Wizard will design aggregations for this attribute.

FIGURE 2: *The properties window with possible values for the Aggregation Usage property.*

You do have some control over the attributes considered as aggregation candidates. You can use the Aggregation Usage property of each dimension attribute to advise MSAS whether a particular attribute should be considered for aggregations. You can edit this property for each attribute under the Cube Structure tab within BIDS. **FIGURE 2** shows the property window with possible values for the Aggregation Usage property.

The default value of the Aggregation Usage property is — you guessed it — *default*. For a full explanation of how values of this property affect the wizards' behavior, please refer to the Analysis Services 2005 Performance Guide.

MSAS wizards also consider the partition and attribute statistics when building aggregations. Analysis Services does not maintain the partition row counts automatically — it is up to the cube architect or administrator to keep a partition's estimated rows property up to date.

Furthermore, the wizards include a "count" button that allows you to query a data warehouse table or tables and determine the number of records for each partition and dimension attribute. If your data warehouse is large and if you have a rough idea of the number of records, you can type in partition counts within the wizard. There's no need to run a query that could take a long time to complete. The estimated number of rows you type in doesn't have to be exact, but it should be fairly accurate in order to have the wizard exercise its algorithm properly.

MSAS wizards never design an aggregation if its size is greater than or equal to one-third of the fact table size for any given partition because scanning through such an aggregation isn't likely to be any more efficient than scanning the partition file. Here is where aggregations and partitions can work in tandem to provide the optimal performance. If your partition files are too large to accommodate aggregations that could benefit your queries, split the partitions so the aggregation size is less than one-third of the partition size.

AGGREGATION MANAGER

Each partition can have a different set of aggregations. When you create your measure group, you can build an initial set of aggregations for each partition or apply the same basic algorithm — perhaps the performance improvement goal of 15% — to all partitions.

Once your users start querying the cube, use the usage-based optimization wizard to fine-tune aggregations for frequently queried partitions. The UBO wizard makes its decisions based on queries that are recorded in the query log. You must configure the MSAS server so MDX queries are recorded in the query log. Note: To use UBO you must record MDX queries into a SQL Server database table. By examining the query log table, you'll discover it doesn't record the actual MDX queries. Instead, it stores a bitmap of hierarchies that were referenced in each query.

Although UBO and storage design wizards use a smart algorithm for building aggregations, you might want to override the choices the wizards make at times and build aggregations manually. Fortunately, with Analysis Services 2005 Service Pack 2, Microsoft shipped a sample application called Aggregation Manager that gives you complete control over the aggregations designed for your measure groups. This tool is similar to Partition Manager, which was available as an unsupported sample application with MSAS 2000, but Aggregation Manager is more powerful.

Once you connect to an instance of MSAS using Aggregation Manager, you can navigate to the particular measure group of interest. The tool shows you existing aggregation designs and the partitions that use the particular aggregation design. For example, **FIGURE 3** shows that two partitions within

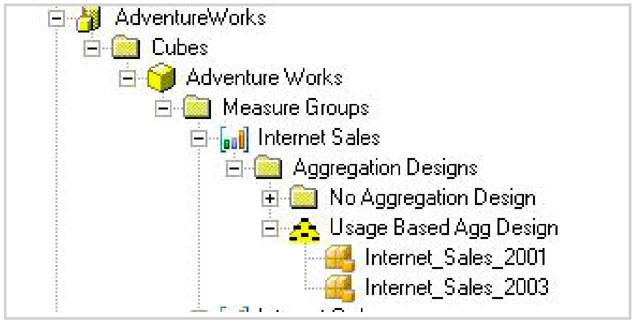
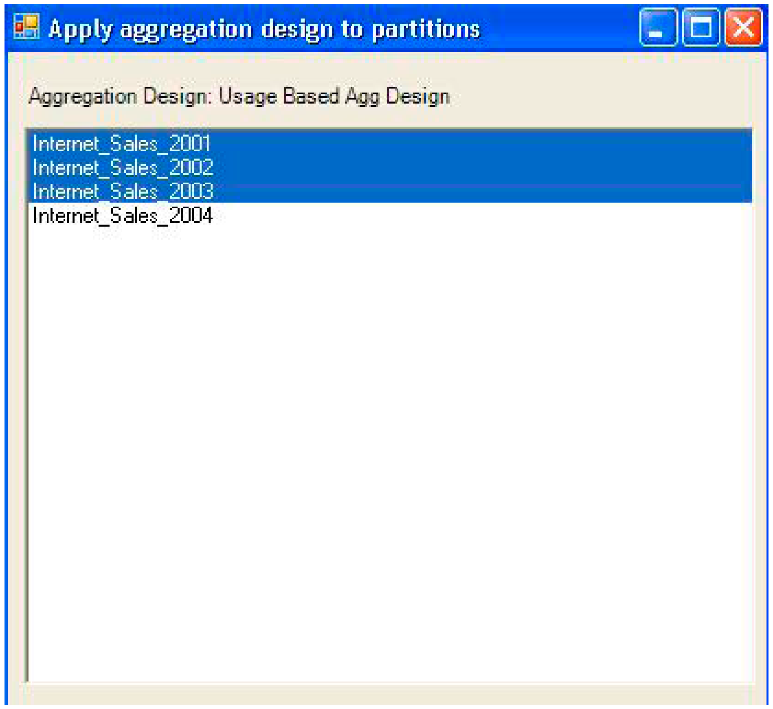

```
AdventureWorks
  Cubes
    Adventure Works
      Measure Groups
        Internet Sales
          Aggregation Designs
            No Aggregation Design
            Usage Based Agg Design
              Internet_Sales_2001
              Internet_Sales_2003
```

FIGURE 3: *Two partitions within the Internet Sales measure group use the aggregation design called "Usage Based Agg Design."*



Aggregation Design: Usage Based Agg Design

Internet_Sales_2001
Internet_Sales_2002
Internet_Sales_2003
Internet_Sales_2004

FIGURE 4: *Choose "change partitions" to apply the same set of aggregations to any number of partitions.*

NOTE: You must right-click the measure group and choose "save to the server" to apply the aggregation designs to the partitions selected from the previous dialog. Alternatively, Aggregation Manager allows you to script the partition to an XMLA file.

There's no need to reprocess partitions fully — you can calculate aggregations by processing indexes on any partition.

the Internet Sales measure group use the aggregation design called "Usage Based Agg Design."

You can right click on the aggregation design and choose "change partitions" to apply the same set of aggregations to any number of partitions using the dialog shown in **FIGURE 4**.

You could subsequently execute the ALTER statement through SQL Server Management Studio or ascmd utility. The ALTER command applies aggregation design to existing partitions. There's no need to reprocess the partitions fully — you can calculate aggregations by processing indexes on any partition.

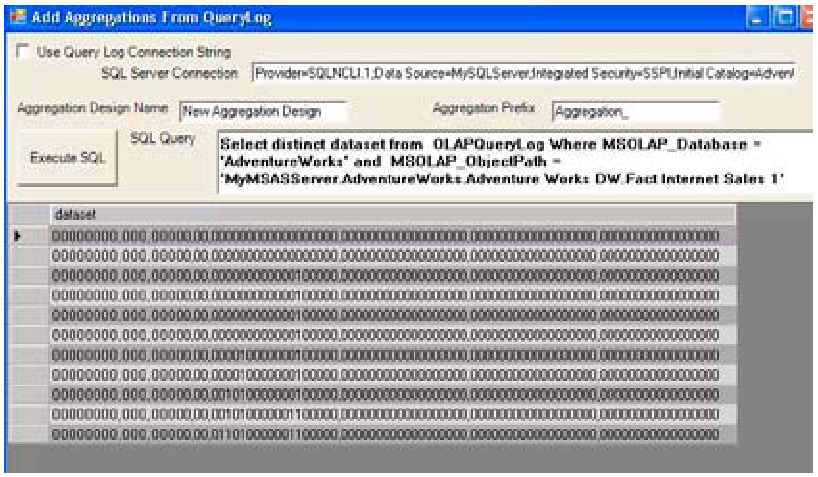
Another useful option within Aggregation Manager allows for the building of aggregations based on the contents of query log. Right click on "Aggregation Designs" and choose "add from query log." The resulting dialog box contains a connection string to the SQL Server database where you store the query log and the default query that retrieves all hierarchy bitmaps from the query log. You can execute the default query or modify it as desired to build aggregations specifically for the MDX queries you wish to tune, as illustrated in **FIGURE 5.**

Yet another alternative is to pick and choose the attributes you wish to include in aggregations. To do so, right click on an existing aggregation design and choose "edit." As shown in **FIGURE 6**, you can choose any attribute from any hierarchy to include in the aggregation design.

From this screen, you can eliminate redundant and duplicate aggregations. Aggregations are redundant if they include an attribute higher in the relationship chain than another attribute that is also included in the aggregation. For example, if the store attribute is included, then store city and store country attributes would be redundant. This is because MSAS can use the lower level aggregations to resolve queries requesting higher levels within the same hierarchy.

You can take advantage of two powerful features for tuning MSAS query performance. Partitions allow you to divide your data into manageable chunks and could help by limiting the amount of data to be scanned at query execution time. Aggregations are summary values pre-computed during partition processing that MSAS can use to resolve queries rather than scanning partition files.

Using these tools in tandem helps you achieve the best possible performance for your analytical applications.


Add Aggregations From QueryLog
Use Query Log Connection String
Execute SQL

SQL Server Connection: Provider=SQLNCLI.1;Data Source=MySQLServer;Integrated Security=SSPI;Initial Catalog=Advent

Aggregation Design Name: New Aggregation Design

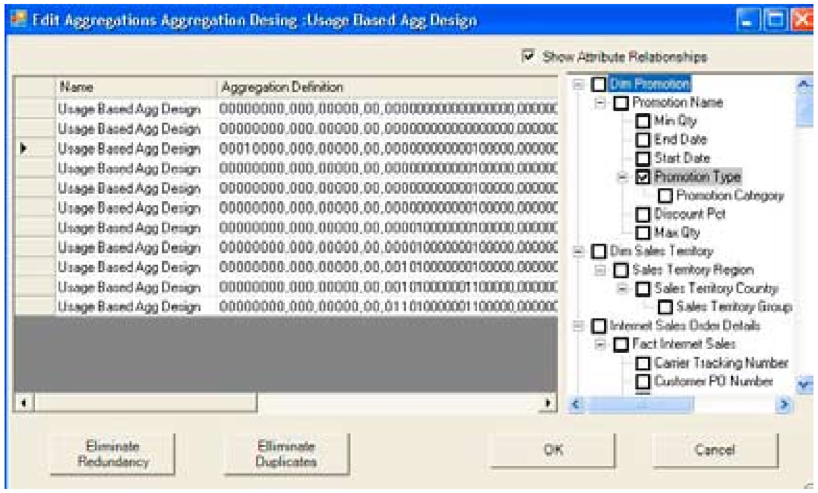
Aggregation Prefix: Aggregation_

SQL Query:

```
Select distinct dataset from  OLAPQueryLog Where MSOLAP_Database =
'AdventureWorks' and  MSOLAP_ObjectPath =
'MyMSASServer.AdventureWorks.Adventure Works DW.Fact Internet Sales 1'
```

```
dataset
00000000,000,00000,00,0000000000000000,0000000000000000,0000000000000000,0000000000000000
00000000,000,00000,00,0000000000000000,0000000000000000,0000000000000000,0000000000000000
00000000,000,00000,00,0000000000010000,0000000000000000,0000000000000000,0000000000000000
00000000,000,00000,00,0000000000010000,0000000000000000,0000000000000000,0000000000000000
00000000,000,00000,00,0000000000010000,0000000000000000,0000000000000000,0000000000000000
00000000,000,00000,00,0000000000010000,0000000000000000,0000000000000000,0000000000000000
00000000,000,00000,00,0001000000010000,0000000000000000,0000000000000000,0000000000000000
00000000,000,00000,00,0001000000010000,0000000000000000,0000000000000000,0000000000000000
00000000,000,00000,00,0010100000010000,0000000000000000,0000000000000000,0000000000000000
00000000,000,00000,00,0010100000110000,0000000000000000,0000000000000000,0000000000000000
00000000,000,00000,00,0110100000110000,0000000000000000,0000000000000000,0000000000000000
```

FIGURE 5: *Execute the default query or modify it as desired to build aggregations specifically for the MDX queries.*


Edit Aggregations Aggregation Desing :Usage Based Agg Design
Show Attribute Relationships
Eliminate Redundancy
Eliminate Duplicates
OK
Cancel

Name	Aggregation Definition
Usage Based Agg Design	00000000,000,00000,00,0000000000000000,00000
Usage Based Agg Design	00000000,000,00000,00,0000000000000000,00000
Usage Based Agg Design	00010000,000,00000,00,0000000000010000,00000
Usage Based Agg Design	00000000,000,00000,00,0000000000010000,00000
Usage Based Agg Design	00000000,000,00000,00,0000000000010000,00000
Usage Based Agg Design	00000000,000,00000,00,0000000000010000,00000
Usage Based Agg Design	00000000,000,00000,00,0001000000010000,00000
Usage Based Agg Design	00000000,000,00000,00,0001000000010000,00000
Usage Based Agg Design	00000000,000,00000,00,0010100000010000,00000
Usage Based Agg Design	00000000,000,00000,00,0010100000110000,00000
Usage Based Agg Design	00000000,000,00000,00,0110100000110000,00000

- ☐ Dim Promotion
  - ☐ Promotion Name
    - ☐ Min Qty
    - ☐ End Date
    - ☐ Start Date
    - ☑ Promotion Type
      - ☐ Promotion Category
    - ☐ Discount Pct
    - ☐ Max Qty
- ☐ Dim Sales Territory
  - ☐ Sales Territory Region
    - ☐ Sales Territory Country
      - ☐ Sales Territory Group
- ☐ Internet Sales Order Details
  - ☐ Fact Internet Sales
    - ☐ Carrier Tracking Number
    - ☐ Customer PO Number

FIGURE 6: *Choose any attribute from any hierarchy to include in the aggregation design.*

SQL Server Insider PERFORMANCE

Optimize SAN setup for improved SQL Server performance

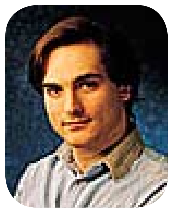
BY SERDAR YEGULALP

STORAGE AREA NETWORKS, or SANs, are widely used to store data for Microsoft SQL Server installations with large databases. SANs are one of the most economical ways to deal with very large data sets. They're designed to scale better in this regard than disk arrays installed directly on the host.

However, setting up SQL Server databases on a SAN requires some awareness on the part of the database administrator about the way SANs work. You can't simply dump a database onto the SAN and expect to get the same results you've been getting.

Here are some pointers about how to get the most out of a SAN setup when using SQL Server.

SERDAR YEGULALP



has been writing about Windows and related technologies for more than 10 years and is a frequent contributor to various sections of TechTarget as well as other publications. He hosts the Web site Windows Insight, where he posts regularly about Windows and has an ongoing features guide to Windows Vista for emigrants from Windows XP.

The first issue to consider is the bandwidth of the data channel. SANs are typically connected to their host over a fiber optic link. While these links are fast — the 20GFC protocol can in theory produce up to 2,000 MB/sec throughput — there's a big difference between their rated speed and the actual speed obtained when connected to a host. If you have an existing SQL Server installation that you're migrating to a SAN setup, find out what the I/O demands are for the existing setup first. Odds are the SAN will be able to comfortably encompass those demands, but make sure that is the case before you make the leap.

Measuring bandwidth demands in SQL Server 2005 isn't too tough — just set up the performance monitoring application to derive logs of SQL Server's I/O usage. You can do this over the course of a day or even an hour of high utilization. This may also give you a chance to flag any previously undetected I/O performance issues that need to be squelched in a high-performance environment. SQLIO is another useful, albeit unsupported, tool for deriving live I/O stats.

Your SAN should be robustly designed — a disk system that complements SQL Server's own needs as far as data consistency goes.

The specific SAN you choose should be a robustly designed product able to host SQL Server data reliably. Microsoft uses the term "stable media" to describe any storage system that can survive a system reboot or failure without losing anything, including pending writes that might be currently held in a cache. The idea is to have a disk system that complements SQL Server's own needs as far as data consistency goes. To be frank, almost any SAN worth spending money on is already going to sport those sorts of features; if it doesn't, then you're not getting much of a SAN.

One useful feature for SQL Server 2005 is the point-in-time snapshot. Use it in conjunction with things like Analysis Services in creative ways (the Analysis Services link goes to an article that tells you how to accomplish this).

Another issue to consider is the way SAN abstracts the physical devices it presents to the system. SANs present their devices to the computer as if they were local disks, but the LUNs the computer sees and the actual disk arrangement can be

Zoning allows specific disks in the SAN to be dedicated to a specific LUN. This means more accurate capacity and performance planning for your SQL Server environment.

radically different. You'll want to know about these things if the plan for your database, table and physical file structure is to take maximum advantage of parallelism — and it better, whenever it can.

For example: If you have a database that you want to place entirely on its own physical spindle, you may be inclined to do this by assigning it to a given LUN as advertised by the SAN. But, if that said LUN is actually split across disks that are shared by another LUN, and you don't know it, then you won't get the performance you need. If you are not responsible for setting up storage on the SAN, consult the person who is and describe your needs to him or her in detail. The folks at the Microsoft SQL Server Development Customer Advisory Team have some suggestions about how to configure LUNs on a SAN for SQL Server, with both rules and exceptions spelled out in detail.

Hilary Cotter has written a series of general suggestions for SANs in SQL Server that are worth noting. One suggestion is particularly applicable here: Use zoning, a feature supported by many SANs. Zoning allows specific disks in the SAN to be dedicated to a specific LUN and, therefore, allows more accurate capacity and performance planning.

One final note, which also comes on behalf of Microsoft, is about which RAID level to use on the SAN. For SQL Server data and logs, Microsoft recommends using RAID 10, when possible, for a variety of reasons. It offers better availability than RAID 5 and better support for write-heavy environments — making it a good choice for the temporary database too. The extra cost of implementing RAID 10 is more than worth it if you can afford it. If you can't shell out that much more, RAID 5 is an acceptable substitute in most cases, although it does come with a bit of a hit to performance.

SANs may be one of the most economical ways to deal with very large data sets in SQL Server. But setup is key and requires some planning on the part of database administrators.

SQL Server Insider BACKUP AND RECOVERY

DECISION TIME: Restore a full SQL Server database or one failed filegroup?

BY SERDAR YEGULALP

Both SQL Server 2000 and SQL Server 2005 allow you to restore whole databases and to selectively restore individual files or filegroups. This is useful if one particular file or filegroup has failed and you want to restore it without going through the hassle of restoring the entire database. It's especially useful if the file/filegroup in question isn't very large to begin with or is only a small part of a large database.

Here's a concrete example: If you have a single bad file that's only 50 MB, and your entire database runs to several dozen gigabytes, it makes more sense to restore the single bad file, if possible. One scenario where this sort of thing happens often is when the file or filegroup is on a separate drive and the drive fails. Usually, you'll incur less total downtime by restoring a single file/filegroup, since it cuts down significantly on the amount of data that has to be restored.

Now, why would you not want to restore the single bad file? Here are a few reasons:

1 **YOU NEED TO HAVE TRANSACTION LOG BACKUPS.** If you want to restore a file or filegroup from backups, you'll also have to restore the transaction log backups created with them so that the entire database can be brought to a consistent state. In SQL Server 2000 and 2005, you must use the Full Recovery or Bulk-Logged Recovery modes — not Simple Recovery — to make this possible. SQL Server does make a best effort to determine if a file or filegroup has been modified since the last backup. If it hasn't been, then the transaction logs aren't needed. But, on the whole, count on needing transaction log backups — and if you don't already have a recovery or backup plan in place, which does back up transaction logs, then set one up.

2 **INCONSISTENCIES IN DATA** between the tables in that file or file group and the rest of the database may make it a bad idea. If you have tables that depend on each other, which aren't stored in the same physical file or filegroup — and sometimes that's unavoidable — restoring only one file or filegroup may cause it to fall out of sync with the rest of the database. For instance, if you have one table that's referenced to another table with a JOIN using a view or stored procedure, restoring one without also restoring the other could be problematic.

3 **DOING IT THE OLD WAY... YOU ONLY HAVE ONE FILEGROUP IN THE DATABASE**. If all of your data is stored in just one file or filegroup — which happens if it's not a very large database to begin with — then it makes no sense to attempt a file/filegroup restore.

WHEN TO USE SELECTIVE FILE/FILEGROUP RESTORES

The main reason for performing selective restores of files or filegroups is to make it possible to spot-recover damage to a database that is too big to restore wholesale. On smaller or less heavily trafficked databases and on nonproduction systems or with databases that have only one filegroup, it's scarcely worth the effort to do a selective restore since it's often just as easy to restore the whole database at once.

I've found that most of the time when people want to perform a file/filegroup restore, they are really trying to retrieve a specific table as it was at an earlier point in time. This is not an explicitly supported feature in SQL Server, but there is a way to do it, provided you don't mind manually managing any of the inconsistencies that might arise because of doing it this way. If you have a full database backup handy, you can simply restore that backup as a differently named instance of the same database. Then, roll that database forward with the transaction logs to the point desired — if that's required — and manually copy the table in question to the target database.

I've done variations of this trick myself a few times but only on a table that I knew wasn't strongly interrelated with other tables in the same database. My example involves a chat site that also featured a message board system. I've often had to restore data accidentally deleted from a message board, which — thankfully— is fairly self-contained. The only JOINs made from the data in the message board table were outward, not inward. Therefore, I was comfortable with updating that table with impunity since I knew I wouldn't be leaving that table out of sync with anything else.

In SQL Server 2000 and higher, you can use the PARTIAL clause when you do a RESTORE so that only the filegroup with the needed data is restored. This is useful as both a time-saving and space-saving measure. You don't have to tediously restore everything just to get at one table, and perhaps there simply is no space available to do a full restore.

SQL Server INSIDER

is brought to you by SearchSQLServer.com. The articles "Optimize SAN setup for improved SQL Server performance" and "Decision time: Restore a full SQL Server database or one failed filegroup?" originally appeared on SearchSQLServer.com

EDITORS
Heidi Sweeney
Christine Casatelli

COPY EDITOR
Martha Moore

DESIGN
Ronn Campisi
www.ronncampisi.com