**Choosing a Tabular or Multidimensional Modeling Experience in SQL Server 2012 Analysis Services**

**Microsoft Business Intelligence Technical Article**

**Authors**

**Hitachi Consulting:**

**Liz Vitt – Author**

**Scott Cameron – Author**

**Hilary Feier - Reviewer**

**Microsoft:**

**T.K. Anand - Reviewer**

**Ashvini Sharma - Reviewer**

**Published: May 2012**

**Applies to:** SQL Server 2012 Analysis Services

**Summary:** This white paper provides practical guidance to help BI professionals and decision makers decide whether SQL Server 2012 Analysis Services tabular or multidimensional modeling provides the best fit for your next BI solution.

**Copyright**

This document is provided “as is.” Information and views expressed in this document, including URL and other Internet website references, may change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2012 Microsoft Corporation. All rights reserved.

Contents

[Introduction 5](#_Toc323533771)

[BISM Modeling Primer 5](#_Toc323533772)

[Multidimensional Modeling 5](#_Toc323533773)

[Tabular Modeling 6](#_Toc323533774)

[BISM Client Analysis Tools 7](#_Toc323533775)

[Data Model 7](#_Toc323533776)

[Data Relationships 7](#_Toc323533777)

[One-to-Many Relationships 7](#_Toc323533778)

[Many-to-Many Relationships 7](#_Toc323533779)

[Reference Relationships 8](#_Toc323533780)

[Hierarchies 9](#_Toc323533781)

[Standard Hierarchies 9](#_Toc323533782)

[Ragged Hierarchies 9](#_Toc323533783)

[Parent-Child Hierarchies 9](#_Toc323533784)

[Additional Modeling Features 10](#_Toc323533785)

[Business Logic 12](#_Toc323533786)

[Row-Level Transformations 12](#_Toc323533787)

[Aggregated Values 13](#_Toc323533788)

[Calculations 13](#_Toc323533789)

[Business Logic Scenarios 15](#_Toc323533790)

[Hierarchy Logic 15](#_Toc323533791)

[Custom Rollups 15](#_Toc323533792)

[Semiadditive Measures 16](#_Toc323533793)

[Time intelligence 17](#_Toc323533794)

[KPIs 17](#_Toc323533795)

[Currency Conversion 17](#_Toc323533796)

[Named Sets 17](#_Toc323533797)

[Data Access and Storage 19](#_Toc323533798)

[Performance and Scalability 19](#_Toc323533799)

[Multidimensional Models 19](#_Toc323533800)

[Tabular Models 20](#_Toc323533801)

[Programmability 22](#_Toc323533802)

[Security 22](#_Toc323533803)

[Row/Attribute-Level Security 23](#_Toc323533804)

[Dynamic Security 23](#_Toc323533805)

[Cell-Level and Advanced Security 24](#_Toc323533806)

[Summary 25](#_Toc323533807)

[For More Information 29](#_Toc323533808)

# Introduction

Data modeling is a discipline that has been practiced for many years by BI professionals with one common goal: organizing disparate data into an analytic model that effectively and efficiently supports the reporting and analysis needs of the business. As data modeling evolves through the years with new technologies and tools, organizations face the growing challenge of effectively blending their modeling paradigms in a seamless and coherent manner that not only satisfies diverse analysis needs but also provides a common analysis experience to the business.

With the release of SQL Server 2012, Microsoft addresses this goal and challenge with the introduction of the BI Semantic Model (BISM), a single model that can support a broad range of reporting and analysis while blending two Analysis Services modeling experiences behind the scenes:

* Multidimensional modeling, introduced with SQL Server 7.0 OLAP Services and continuing through SQL Server 2012 Analysis Services, enables BI professionals to create sophisticated multidimensional cubes using traditional online analytical processing (OLAP).
* Tabular modeling, introduced with PowerPivot for Microsoft Excel 2010, provides self-service data modeling capabilities to business and data analysts. The tabular modeling experience is more accessible to these users, many who have spent years working with data in desktop productivity tools like Excel and Microsoft Access. In SQL Server 2012, tabular modeling has been extended to enable BI professionals to create tabular models in Analysis Services or to import a tabular model from PowerPivot into Analysis Services. Note that a PowerPivot model cannot be imported into an Analysis Services multidimensional model.

The goal of this white paper is to provide practical guidance to help you decide which SQL Server 2012 Analysis Services modeling experience – tabular or multidimensional - is the best fit for your next BI solution. The product descriptions and recommendations in this paper are based on SQL Server 2012 Analysis Services, which was released in March 2012. Product features and recommendations may change as Analysis Services multidimensional and tabular modeling evolves in future versions of SQL Server.

# BISM Modeling Primer

Before diving into the detailed differences between multidimensional and tabular modeling, let’s begin with a brief primer on each of the BISM modeling experiences provided by SQL Server 2012 Analysis Services.

## Multidimensional Modeling

At its core, multidimensional modeling creates cubes composed of measures and dimensions based on data contained in a relational database. To use this paradigm, the Analysis Services server must be configured to operate in multidimensional mode, its default setting. In this mode, the OLAP engine uses the multidimensional model to preaggregate large volumes of data to support fast query response times. The OLAP engine can store these aggregations on disk with multidimensional OLAP (MOLAP) storage or store them in the relational database with relational OLAP (ROLAP) storage.

The key characteristics of multidimensional modeling include:

* **Rich Data Model –** The multidimensional model of SQL Server 2012 Analysis Services is in its sixth release and provides extensive functionality to model measures and dimensions from both simple and complex datasets commonly found in enterprise data warehouses. More complex datasets typically include advanced features like many-to-many relationships, parent-child hierarchies, and localization. The multidimensional model provides this functionality out of the box.
* **Sophisticated Analytics –** The multidimensional model also provides an advanced calculation and query language called Multidimensional Expressions (MDX). Using MDX you can create sophisticated business logic and calculations that can operate anywhere in the multidimensional space to accomplish financial allocations, time-series calculations, or semiadditive metrics.

Although comprehensive data modeling and sophisticated analytics are important benefits of multidimensional modeling, they often come with the tradeoff of longer development cycles as well as decreased ability to quickly adapt to changing business conditions. In addition, the multidimensional experience tends to require advanced modeling and MDX skills.

## Tabular Modeling

Tabular modeling organizes data into related tables. If you want to use tabular modeling, Analysis Services must be configured to operate in tabular mode. In tabular mode, you can use the xVelocity (formerly Vertipaq) in-memory engine to load tabular data into memory for fast query response, or you can use DirectQuery to pass queries to the source database to leverage its query processing capabilities.

The key characteristics of tabular modeling include:

* **Familiarity -** Working with tabular data is familiar to many audiences who regularly work with stored in tables in relational databases, Excel, or Access. In addition, calculations are written using Data Analysis Expressions (DAX), a formula language that is considered an extension of the Excel formula language. As such, the skills required to build tabular models are more common or easily learned compared to the skills required to build multidimensional models.
* **Flexibility-** Because there is no rigid organization of data into measures and dimensions; tabular modeling can quicken development cycles, requiring less upfront data preparation and design rigor than multidimensional models. This data architecture can also be more accommodating of data modeling changes over time when it is necessary to update relationships and calculations according to changing business needs.

While familiarity and flexibility are key benefits of tabular modeling, they also come with tradeoffs. For example, tabular modeling may not be suited for those solutions that have highly complex datasets or require sophisticated business logic. Users of the DAX language may often be able to create DAX formulas to provide analytic functionality otherwise unavailable in the tabular model. In these cases, however, it may be more suitable and efficient to use the advanced capabilities natively provided by multidimensional modeling.

# BISM Client Analysis Tools

Whether you choose multidimensional or tabular modeling, it is important to note that you can use client tools that generate either MDX or DAX to query the model. Excel and SQL Server Reporting Services are examples of client tools that generate queries using MDX, and Power View is an example of a client tool that generates queries using DAX. There are two exceptions to this guidance.

* Power View is an interactive data exploration and visualization tool that is a feature of the SQL Server 2012 Reporting Services Add-in for Microsoft SharePoint Server 2010 Enterprise Edition. If you want to use Power View or any other analysis client that uses DAX to query a BISM, you need to use a tabular model. Future versions of SQL Server are expected to provide the ability to use DAX to query multidimensional models so that they can be accessed by a client tool like Power View.
* Tabular models that have been configured to use DirectQuery require a client tool that generates DAX queries such as Power View. Future versions of SQL Server are expected to enable tabular models configured to use DirectQuery to accept MDX queries.

# Data Model

The characteristics of your data model are a core consideration in your choice of modeling experience.

## Data Relationships

A fundamental requirement of any data model is correctly representing how the data elements within that model interrelate and connect, much like the pieces of a jigsaw puzzle. Both tabular models and multidimensional models require you to define relationships among your source data tables. Common relationships that you see in data modeling are one-to-many, many-to-many, and reference relationships.

### One-to-Many Relationships

In a one-to-many relationship, a single record from one table relates to multiple records from another table. An example of a one-to-many relationship is a customer who has multiple sales orders. Both tabular and multidimensional data models natively handle one-to-many relationships.

### Many-to-Many Relationships

In a many-to-many relationship, many records from one table relate to many records from a second table. For example, a single customer has a one-to-many relationship with sales orders; however, each customer can be categorized into one or more customer profiles (such as Sports Enthusiast, Casual Gamer, and Fitness Expert.). Analyzing orders by customer profile is a many-to-many challenge where double-counting may occur: An order for one bicycle by a customer who is both a Sports Enthusiast and a Fitness Expert could easily be counted twice when orders by customer profile are summed to get total orders. Typically many-to-many relationships are managed by breaking them down into two one-to-many relationships using a bridge or intermediate table as pictured in Figure 1.



**Bridge / Intermediate Table**to assign the customer profile



**Customer Table**

**Sales Order Table**

Figure 1 –Many-to-Many Example

In multidimensional models, you can define and build many-to-many relationships directly in the data model by identifying the bridge table and then relating that bridge table to other tables in your model. When aggregating, Analysis Services applies a distinct sum to ensure that data totals are correctly summarized and not improperly inflated.

SQL Server 2012 Analysis Services tabular models do not support the definition of many-to-many relationships. However, you can use the DAX language to create formulas that handle the many-to-many challenge.

### Reference Relationships

A data model may contain a set of common attributes that are related to multiple entities. For example, geographical attributes are related to customers, suppliers, and stores. In multidimensional modeling you must create a dimension containing the common attributes and then create reference dimension relationships to each of the related dimensions. In tabular modeling there is no need to create reference relationships. In a tabular model, all you need to do is create relationships between the table containing the common attributes and the tables containing the related entities.

## Hierarchies

Hierarchies categorize data into a tree structure to facilitate drill-down analysis.

### Standard Hierarchies

Standard hierarchies are composed of ordered levels that come from columns in your source data. For example, a product hierarchy may organize products into subcategories, which can be further organized into categories. In this case, you would have a hierarchy with three levels, where each level comes from a separate column in your source data. Simple hierarchies like the product hierarchy described here are supported in both tabular and dimensional models.

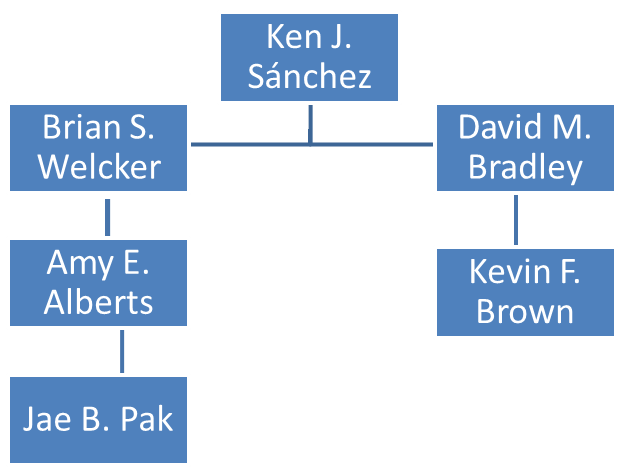
Note that within multidimensional models, there is an added step of creating attribute relationships, which is explicitly identifying the one-to-many relationships between attributes in each dimension. Defining attribute relationships is strongly recommended because they enable more efficient design of precalculated aggregations and because MDX semantics rely on attribute relationships. Tabular modeling is more straightforward because you do not create attribute relationships. Tabular models do not precalculate aggregations and DAX semantics do not rely on identifying the one-to-many relationships between attributes, so in tabular modeling there is no equivalent to multidimensional modeling’s attribute relationships.

### Ragged Hierarchies

Ragged hierarchies occur when a given data element is missing in the hierarchy tree. For example, a ragged product hierarchy occurs if there are products that are never assigned a subcategory, yet they have product category assignments. In these situations, rather than showing the gap in the tree, you may choose to hide the gap to facilitate drill-down analysis. Multidimensional models provide out-of-box support for ragged hierarchies; however, tabular models do not support this capability.

### Parent-Child Hierarchies

Parent-child hierarchies offer a more complicated hierarchical design. The branches in a parent-child hierarchy don’t all have the same number of levels. For example, a parent--child relationship between employee and manager could produce a hierarchy in which some managers only have direct reports while other managers have direct reports who each also have their own direct reports. This kind of hierarchy is modeled by creating a relationship between two columns in a source data table as shown in Figure 2.



**Parent Child Source Data**

**Parent Child Hierarchy Tree**

Figure 2 – Parent Child Hierarchy

Multidimensional models provide out-of-the box functionality that enables you to define and build parent-child hierarchies based on relationships in your source data.

In tabular models, you can leverage DAX functions to create formulas that navigate and use the parent-child structure in calculations. For more information about the use of parent-child hierarchies in tabular models, see [Understanding Functions for Parent-Child Hierarchies in DAX](http://msdn.microsoft.com/en-us/library/gg492192(v=sql.110).aspx) (http://msdn.microsoft.com/en-us/library/gg492192(v=sql.110).aspx).

## Additional Modeling Features

In addition to data relationships and hierarchies, there are additional modeling features that can help you choose the best modeling experience:

* Perspectives enable you to define a subset of a data model for simplified browsing by end users. Perspectives are available in both multidimensional models and tabular models.
* Translations enable multidimensional models to display dimension, attribute, measure, calculated member, and other object names and dimension member values in the language specified by the computer’s localization settings. Enabling this feature requires the model developer to provide the translated object names and to reference the columns in the source data that contain the translated dimension member values. Tabular models do not provide this functionality.
* Actions empower end users to run a Reporting Services report, navigate to a URL, or initiate an external operation based on the context of the cell where the action occurs. For example, using an action, an end user can launch a webpage that displays the company’s product catalog automatically filtered to the product or products that the user was browsing. Actions are natively supported in multidimensional models and many client tools (like Excel and Reporting Services) allow users to execute actions. In SQL Server 2012 the ability to create actions in a tabular model using SQL Server Data Tools is not supported.
* Drillthrough enables you to navigate to the detailed data stored in your model. Drillthrough is available in both multidimensional and tabular modeling. Multidimensional models also enable you to create drillthrough actions so that you can customize the drillthrough experience by specifying the columns that will be returned by the drillthrough action and the cube space where the action will be enabled.
* Write-back is a feature that is typically necessary in budgeting and forecasting applications. In these scenarios, business users typically want to perform “what-if” analysis where they change and update data values in the model and then publish those for others to see. Multidimensional models provide native support for data write-back. In SQL Server 2012 tabular models do not support this functionality.

# Business Logic

Business logic can add tremendous value to any data model in the form of calculations and business rules that enhance data for end-user analysis.

Both tabular and multidimensional modeling offer rich formula languages to implement business logic. Multidimensional modeling leverages MDX and tabular modeling leverages DAX. Before delving into some of the advanced business logic scenarios of each paradigm, it is important to establish a baseline understanding of how business logic can be applied using row-level transformations, aggregated values, and calculations in multidimensional and tabular modeling.

## Row-Level Transformations

You may need to perform calculations and data transformations that are not readily available in your source data. For example, your source data may have a Sales Amount column and a Daily Exchange Rate column, but be missing sales converted to the foreign currency, or your source data may have Employee First Name and Employee Last Name but be missing a concatenated Employee Full Name. Note that in these examples the calculation or data manipulation must occur on row-level, unaggregated data.

In multidimensional modeling, row-level transformations on unaggregated data must be performed before the data is loaded into the model or must be performed when the model is queried. You can transform dimension attributes, like employee names, by either applying the transformation in the data source system or by writing an SQL expression that gets applied when Analysis Services queries the source database. Row-level transformations of numeric data can be performed using an SQL expression before the data is loaded into Analysis Services, or the transformation can be applied using an MDX expression within a Scope statement so that the calculation is applied at the row level. If the transformation is applied before the data is loaded, then Analysis Services can preaggregate the numeric values. If the transformation is applied using a Scope statement, aggregation occurs at query time.

In tabular modeling, row-level transformations are created using calculated columns. When you create a calculated column, you add the column to a specific table in your model and you use DAX formulas to define the column’s values. The formula is then evaluated for every record in that table and is loaded into memory just like any other column in the model. This flexibility allows you to enhance your data directly in the tabular model based on your specific analysis requirements and lessens the need to perform tweaks to upstream data sources that may or may not be able to accommodate your changes in a timely manner. Calculated columns provide a very convenient way to create and persist calculations that must be performed at a detailed level in your data before being aggregated. While this flexibility is powerful, note that calculated columns are not intended to perform the heavy data cleansing or data transformations that you would find in Extract, Transform, and Load (ETL) processes.

## Aggregated Values

In multidimensional modeling, you use measures to create aggregated values. The Analysis Services OLAP engine preaggregates a cube’s measures using aggregate functions like SUM, COUNT, MIN, MAX, and DISTINCT COUNT, and others. During cube processing, each measure is aggregated from bottom to top across all hierarchies. Because this processing happens prior to end-user analysis, the preaggregated measures can provide tremendous benefits to query performance.

When you create a measure in your cube, there is a one-to-one relationship between a cube measure and a numeric column in your source data. As such, in multidimensional modeling, measures are useful when you need to perform bottom-up aggregation of numeric data elements that (1) exist in your source data at the lowest level of detail and (2) require a rollup that leverages one of the native cube aggregate functions.

In tabular modeling, you also use measures to create aggregated values. You create a measure by selecting a column and then specifying the aggregation function (SUM, COUNT, DISTINCT COUNT, MIN, MAX, or AVERAGE), or you can write a DAX expression that specifies function you want to use to aggregate the measure. In tabular modeling, the row-level data is stored in memory and aggregates are calculated at query time. As explained in the next section, in tabular modeling measures can also be used to apply calculations. This can include calculations that are based on multiple aggregated columns.

## Calculations

In multidimensional modeling you use MDX to create calculations. MDX is a both an expression and a query language with functions that natively understand the design of a cube’s dimensions, hierarchies, attribute relationships, and measures. This native understanding enables you to create succinct and powerful expressions that apply business logic across multiple data contexts. You create and store MDX calculations in the cube’s calculation script, where you can control the order in which the logic is applied.

Calculated members are the most common MDX calculations. Calculated members are evaluated at query time after the data is preaggregated. Calculated members can be created in any dimension. When they are created in the measures dimension they are often referred to as *calculated measures*. Calculated members can be fairly simple with basic arithmetic operations such as sales per unit (sales / unit) or spend per person (spend / headcount). They can also be more complex when you need to apply specific business rules such as Rolling 3 Period Average Sales or YTD Margin. For example, if you want to calculate sales for the current time period as a percent of the parent time period you can use the following MDX calculation.

[Measures].[Sales Amount] / ([Date].[Calendar].CurrentMember.Parent,[Measures].[Sales Amount])

Creating a calculated member in a dimension other than the measures dimension adds a value to an attribute in the dimension. For example, if you had a dimension attribute that contained a list of colors, you might want to add the calculated member Primary Colors which would sum the values of the colors red, green, and blue. In tabular modeling, creating a measure is similar to creating a calculated member in the measures dimension in a multidimensional model. In tabular modeling you cannot add a value to a column in a table, so tabular modeling does not support the equivalent of creating a calculated member in a dimension other than the measures dimension in a multidimensional model.

Scope assignments are more advanced than calculated measures but they are also more powerful. As mentioned in the “Row-Level Transformation” section earlier, you can use a Scope statement so that calculations are applied at the row level. However, you can also use a Scope statement to specify any range of cube cells where you want to apply a calculation. Scope assignments are compiled ahead of query time and enable Analysis Services to provide an optimized execution path when the calculation is queried. Given their strength, scope assignments can not only do the work of multiple calculated measures but they can also do the work more efficiently. For example, in a budgeting solution, you want to assign next year’s budget for the East region to be 90 percent of their current year’s budget. You want the new West region budget to be 75 percent of their current year’s budget. You want the new South region budget to be 105 percent of their current year’s budget and the new North region budget to be the same as their current year’s budget. Rather than writing a single complex calculated measure with nested IF statements or multiple calculated measures that segregate each budget scenario individually, you can use scope assignments to effectively apply these ratios at the Region level and then aggregate the data totals. For example, if you wanted to convert sales amount into a foreign currency using daily exchange rates you could use the following MDX expression:

Scope([Date].[Date]);

This = [Measures].[Sales Amount] \* [Measures].[Daily FX Rate];

End Scope;

In tabular modeling you use DAX to create calculations. As mentioned previously, in tabular modeling you apply row-level calculations by creating calculated columns. You can also apply calculations when you create a measure by writing a DAX expression. Because you explicitly use a combination of DAX row-level and aggregation functions, measures in tabular models are very flexible. You can apply row-level functions and then apply an aggregation function so that your measure applies calculations before aggregation, or you can apply aggregation functions first and then apply row-level functions so that your measure applies calculations after aggregations.

DAX can dynamically evaluate a formula in different data contexts (not just the current view of an Excel worksheet or a PivotTable) using a special set of functions called FILTER functions. In the broadest sense, these functions have a similar purpose to Analysis Services scope assignments in that they enable you to define and perform a calculation over a specific set of rows. For example, you can use FILTER functions to handle the budgeting example described previously.

## Business Logic Scenarios

Now that you have a good sense of how you can create and apply basic business logic in MDX and DAX, consider the following calculation scenarios to compare and contrast tabular and multidimensional modeling experiences.

### Hierarchy Logic

As stated previously, hierarchies provide a way for business users to drill up or drill down during data analysis. In some situations, it is useful to create calculations that navigate the hierarchy. For example, consider a product dimension where you have product category, product subcategory, and product. For each level in the hierarchy, you want to add a calculation that measures how members of each level contribute to the parent’s sales total. This is called a *percent of parent* calculation given that it must navigate the hierarchy to return the desired value.

Both MDX and DAX provide functions to work with data that is organized into a hierarchy and create calculations like percent of parent; however, the MDX functions tend to be more straightforward and easy to use. For example, in MDX this is the expression that gives percent of parent in the Product dimension.

[Measures].[Sales Amount] /  
([Product].[Product Categories].CurrentMember.Parent, [Measures].[Sales Amount])

This more complex expression is required to create the same percent of parent calculation using DAX.

IF(

ISFILTERED(Product[Product])

,[Sales]/CALCULATE([Sales],ALL(Product[Product]))

,IF(

ISFILTERED(Product[Subcategory])

,[Sales]/CALCULATE([Sales],ALL(Product[Subcategory]))

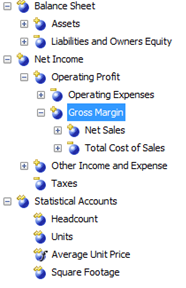
,1

)

)

### Custom Rollups

Although uniform data summarization is applicable in many scenarios, there are also situations where you want to have finer-grained control over how your data rolls up. One example of this is the case of finance models where you have a chart of accounts (usually in a parent-child format) with specific rollup logic required for each account. As shown here, the calculation for Gross Margin is Net Sales minus Total Cost of Sales, and the calculation for Operating Profit is Gross Margin minus Operating Expenses.

Not only do multidimensional models provide native support for parent-child hierarchies, they also provide built-in account intelligence, which enables you to easily apply unary operators and MDX formulas at the account level that drive the data rollup.

In tabular models, parent-child or account intelligence is not built-in, but you can build your own solution using a combination of calculated columns and measures to build out the parent child hierarchy and apply the custom rollup.

### Semiadditive Measures

Generally speaking, semiadditive measures are those that uniformly aggregate across all dimensions except for date. Examples of semiadditive measures include opening balance and closing balance. For these measures, you want to apply special logic to correctly summarize the data by time period. After all, the inventory stock-on-hand balance for the month of March is not the sum of the stock-on-hand for all days in March. In addition, this balance should also correctly work across all date attributes like quarter and year. For example, the Q1 inventory stock-on-hand balance should be the same balance that was reported on March 31st (assuming March 31st is the last day in Q1).

Multidimensional models provide out-of-box support for semiadditive measures with special aggregate functions like First Child, Last Child, FirstNonEmptyChild, and LastNonEmptyChild. If these aggregate functions do not satisfy your specific logic requirements, you can also write custom MDX formulas.

Tabular models provide similar functions such as ClosingBalanceMonth and OpeningBalanceMonth. There are additional functions that apply across other date attributes like quarter and year.

### Time intelligence

Almost every BI solution that you encounter will require time intelligence. Time intelligence includes being able to calculate year-to-date summaries and perform prior year comparisons. Both MDX and DAX provide time-series functions; however, each uses a slightly different data model design.

Multidimensional models provide out-of-the-box time intelligence through the Analysis Services Business Intelligence Wizard. Using this wizard, time calculations can be added to the design of the time dimension and also applied to all measures in the model. Although using the wizard is one way to build time calculations, you can also write your own MDX calculations within the multidimensional model.

In tabular models, although there is no wizard to create time intelligence calculations, you can manually create calculations by creating DAX formulas that leverage a variety of functions including TOTALMTD and TOTALYTD as well as SAMEPERIODSLASTYEAR.

### KPIs

Key performance indicators (KPIs) identify special measures that you want to monitor against a target value using a visual indicator such as a stoplight. Both multidimensional and tabular models provide support for KPIs. Both provide the ability to assign a target for a measure and to use the comparison of actual to target to assess the performance status of the measure. Multidimensional models provide the added ability to assess the KPI’s trend and assign a separate visual indicator to represent how the KPI performs over time.

### Currency Conversion

Currency conversions require you to convert currency data from one or more source currencies into one or more reporting currencies. For example if your organization processes sales transactions in EUR, JPY, and USD, in order to consolidate sales reporting across the entire organization, you will need to convert the sales transactions into one or more reporting currencies.

To implement currency conversions in either modeling experience, you must have access to the currency exchange rate data and include that data in your model.

In multidimensional models, you can use the Analysis Services Business Intelligence Wizard to create MDX currency conversion calculations that are optimized to support multiple source and reporting currencies. In a tabular model you can build your own currency conversion solution by creating DAX formulas.

### Named Sets

In multidimensional modeling named sets provide a way for you to return a set of dimension members that are commonly used in reporting applications. For example, you may want to create a named set to return the last 12 months. Creating this named set within your cube enables you to centrally define the set logic, to access the set from any reporting application, and to simplify the logic stored within your reporting application. To create the Last 12 Months named set you could use the following MDX expression.

Create Set CurrentCube.[Last 12 Months] As  
Max([Date].[Calendar].[Month]).Lag(11):Max([Date].[Calendar].[Month])

Named sets are not available in tabular modeling.

# Data Access and Storage

## Performance and Scalability

Performance and scalability are critical factors that must be considered for the success of any BI solution. As each of the modeling experiences leverage different underlying technologies, they have different performance characteristics and behaviors that you must understand to properly consider which modeling experience best fit your needs.

### Multidimensional Models

As stated earlier in the white paper, the multidimensional models of Analysis Services use an OLAP engine. On disk, OLAP data can be stored in MOLAP and ROLAP data architectures. In MOLAP, data is stored on disk in an optimized multidimensional format with typical 3x compression. In ROLAP, data is stored in the source relational database.

When you think about performance, it is generally useful to think about it into two buckets: query performance and processing performance.

#### Query Performance

Query performance directly impacts the quality of the end-user experience. As such, it is the primary benchmark used to evaluate the success of an OLAP implementation. Analysis Services provides a variety of mechanisms to accelerate query performance, including aggregations, caching, and indexed data retrieval. In addition, you can improve query performance by optimizing the design of your dimension attributes, cubes, and MDX queries.

One of the primary ways to optimize query performance is the use of aggregations. An aggregation is a precalculated summary of data that is used to enhance query performance for multidimensional models. When you query a multidimensional model, the Analysis Services query processor decomposes the query into requests for the OLAP storage engine. For each request, the storage engine first attempts to retrieve data from the storage engine cache in memory. If no data is available in the cache, it attempts to retrieve data from an aggregation. If no aggregation is present, it retrieves the data from a measure group’s partitions.

Designing data aggregations involves identifying the most effective aggregation scheme for your querying workload. As you design aggregations, you must consider the querying benefits that aggregations provide compared with the time it takes to create and refresh the aggregations. In fact, adding unnecessary aggregations can worsen query performance because the rare hits move the aggregation into the file cache at the cost of moving something else out.

Caching is also important for Analysis Services query performance tuning. You should have sufficient memory to store all dimension data plus room for caching query results. During querying, memory is primarily used to store cached results in the storage engine and query processor caches. To optimize the benefits of caching, you can often increase query responsiveness by preloading data into one or both of these caches. This can be done by either pre-executing one or more queries or using the create cache statement.

#### Processing Performance

Processing is the operation that refreshes data in an Analysis Services database. The faster the processing performance, the sooner users can access refreshed data. Analysis Services provides a variety of mechanisms that you can use to influence processing performance, including efficient dimension design, effective aggregations, partitions, and an economical processing strategy (for example, incremental vs. full refresh vs. proactive caching).

You can use partitions to separate measure data (typically fact table data) into physical units. Effective use of partitions can enhance query performance, improve processing performance, and facilitate data management. For each partition, you can have a separate aggregation design and a separate refresh schedule, which can greatly optimize processing performance. For each fact table you can also have a combination of MOLAP and ROLAP partitions. This type of partitioning strategy can be used to provide real-time querying or can be used to provide access to data sets too large to process into a cube.

Using query and processing optimization techniques like these can help you scale your multidimensional models to handle terabytes of data. For more information about performance tuning, see the [Analysis Services 2008 R2 Performance Guide](http://sqlcat.com/sqlcat/b/whitepapers/archive/2011/10/10/analysis-services-2008-r2-performance-guide.aspx) (http://sqlcat.com/sqlcat/b/whitepapers/archive/2011/10/10/analysis-services-2008-r2-performance-guide.aspx).

### Tabular Models

Tabular models use the xVelocity analytics engine, which provides in-memory data processing, or DirectQuery, which passes queries to the source database to leverage its query processing capabilities.

The benefits of columnar databases and in-memory data processing go hand in hand. Columnar databases achieve higher compression than traditional storage, typically 10x compression depending on the data cardinality. Data cardinality focuses on characterizing the data distribution within a single column. High data cardinality means that the data values within a column are highly unique (for example, customer number). Low data cardinality means that the data values within a column can repeat (for example, gender and marital status). The lower the data cardinality, the higher the compression, which means that more data can fit into memory at a given time. As data modelers, it is important to understand the cardinality of your data to determine which data sets are best suited for your tabular model as well as the associated memory requirements to support the model.

#### Querying Performance

When a user queries a tabular model, the engine performs memory scans to retrieve the data and to calculate aggregations on the fly without the need of disk I/O processing. This approach can provide very high query performance without requiring special tuning and aggregation management.

The best and easiest way to optimize query performance for tabular models is to maximize available memory. From a scalability perspective, data volume is mostly limited by physical memory. It is highly recommended that you provide sufficient memory to contain all of the data in your tabular model. In scenarios where memory is constrained, the in-memory engine also provides basic paging support according to physical memory. In addition, there are server-side configuration settings that allow IT to more finely manage the memory available to tabular models. For more information about tabular model memory configuration, see [Memory Properties](http://msdn.microsoft.com/en-us/library/ms174514.aspx) (http://msdn.microsoft.com/en-us/library/ms174514.aspx).

#### Processing Performance

Tabular models provide two primary processing performance differences from multidimensional models:

* Unlike multidimensional models, tabular models load data directly into memory and do not require writing data onto disk
* Because tabular models do not categorize data into dimensions and measure groups, processing can be much more flexible. Both of these differences mean there is less overhead with each data refresh which in turn can enable quicker turnaround times and greater agility.

Consider this example. In your organization it is common for sales reps to move from one region to another on a regular basis. Business users want to see the sales data rolled up by the latest region and sales rep assignments.

In a multidimensional model, to accomplish this task, you must first refresh your sales organization dimension. After the sales organization dimension has been refreshed, you must refresh the sales measure group partition. Refreshing the sales partition updates both the detailed data and aggregations. The final step in your data preparation (as a best practice) is to warm the Analysis Services query cache to retrieve the most useful data from disk into memory.

Depending on your data model design, the data size, and your specific choice in processing techniques (incremental vs. full processing), this could take minutes or hours. The good news is that there are a variety of proven techniques that BI professionals use every day to optimize the processing footprint of multidimensional models as they balance data processing requirements with data availability demands.

Now consider the same scenario in a tabular model. In the tabular model, there is no concept of dimensions and measure groups. Instead, data is organized into tables that have relationships to each other. Assume that sales organization data and sales data are each in their own respective tables with a relationship based on the individual sales rep. With this design, when you refresh your sales organization table, it automatically updates any impacted calculated columns, relationships, and user hierarchies. This means that the sales data automatically reflects the updated sales region rollups without the need to reprocess the sales data. This flexibility can provide significant benefits when you have rapidly changing dimensions and you need the data to reflect the latest updates.

In addition, note that with tabular models, it is also not necessary to build aggregations, write data to disk, or warm the query cache to get the data into memory. With tabular models, data moves from disk directly into memory and is ready to go.

Similar to multidimensional models, tabular models enable you to break your table data into partitions, eliminating the need for unnecessary data processing. For example, you can break down larger tables into multiple partitions, such as one monthly partition for each month in the current year and then one yearly partition for each of the prior years. This approach enables you to isolate those processing partitions that require a refresh.

Unlike multidimensional models, note that while you can process multiple tables in parallel, you cannot process an individual table’s partitions in parallel.

#### DirectQuery

As an alternate to the xVelocity in-memory mode of tabular models, BI professionals can also build tabular models using DirectQuery mode. DirectQuery is available for tabular models with SQL Server data sources. DirectQuery provides you with the ability to bypass data processing by passing DAX queries and calculations to the source database to leverage the capabilities of SQL Server. This can be especially useful with large data volumes that require frequent refreshing. With DirectQuery, however, calculated columns and some DAX functions are not supported.

## Programmability

Analysis Management Objects (AMO) is the API for developing and managing Analysis Services objects. This API was created before tabular modeling was added to Analysis Services and so it only contains classes for objects traditionally associated with multidimensional modeling: cubes, dimensions, measures groups, MDX scripts, and so on. However, this API can also be used for developing and managing tabular models. This is a benefit of multidimensional and tabular modeling being encapsulated by the BI Semantic Model. While internally tabular and dimensional models are distinct, the BISM presents the same external interface. Although you can use AMO for programming both tabular and multidimensional models, it is a less intuitive interface for tabular models. For more information, including a tabular model AMO code sample, see [Analysis Services Tutorials](http://msdn.microsoft.com/en-us/library/hh231701.aspx) (http://msdn.microsoft.com/en-us/library/hh231701.aspx)

# Security

Having an appropriate data security strategy is important to make sure that the right people have access to the right data. Organizations must control data access in order to keep their data assets secure and to comply with privacy regulations. Both multidimensional models and tabular models offer a set of robust capabilities that satisfy a broad range of security requirements. There are subtle differences in capabilities which are important to understand before choosing the modeling experience that will best meet your security needs.

In Analysis Services, you manage multidimensional and tabular project security by creating a role and granting permissions to the role. Next, you add Windows user names and groups to the role, thereby granting the corresponding users access based on the role’s permissions.

## Row/Attribute-Level Security

In a multidimensional project, you use the concept of *dimension data security* to manage row-level access. To implement dimension data security for a role, you grant or deny access to dimension data by selecting or deselecting dimension members. You can also implement a more complex security configuration by defining a set of members using an MDX expression. You also specify whether the role should be granted or denied access to new dimension members. The access you grant or deny to a dimension member impacts the access a role has to related dimension members. For example, if you limit a role so that it can only access the Mountain Bikes product subcategory, members of the role can only view the Bikes product category and the products and sales that belong to the Mountain Bikes subcategory.

In a tabular project, you implement row-level security by granting access to rows in a table. In a SQL Server Data Tools tabular project, you grant permission by entering a DAX expression that filters the rows in a table. The role has access to new table rows if they satisfy the DAX filter. The access you grant to a row in one table impacts the access a role has to rows in related tables. If two tables have a one-to-many relationship, row filters on the table on the one side of the relationship filter rows in the table on the many side of the relationship, but not the other way around. For example, if you limit a role so that it can only view the Mountain Bikes row in the product subcategory table, members of the role can only view rows in the products and sales tables that are related to the Bikes subcategory. However, members of the role can still view all the rows in the product category table (Bikes, Clothing, and so on.).

## Dynamic Security

Your organization may need to limit access to data based on a user’s ID or other dynamic criteria. For example, associates are only allowed to see their own performance and HR data. However, creating a security role for each individual in an organization may be impractical. Instead, you can implement dynamic security, which provides the capability to drive security logic based on a user ID or some other dynamic criteria. Both tabular and multidimensional projects support dynamic security. You can configure dynamic, user-based security if your data contains a relationship between user IDs and the data users have permission to access by including the relationship in the MDX or DAX expression that you are using to manage permissions.

## Cell-Level and Advanced Security

For many applications, there is a need to restrict data access using more complex criteria than simply a row in a table. Take, for example, an employee satisfaction survey that shares aggregate results from a feedback survey. These models often contain highly sensitive data, and individual survey responses must be kept protected. Although the model may not contain people’s names, with a small enough sample size the identity of individual respondents might be derived. In these cases, you may want to implement more complex logic that looks at the sample size and only allows access to the resulting measure if the number of responses are greater than a certain response count. Furthermore, there may be specific questions and metric combinations you would like to specifically restrict for only HR to see. Multidimensional projects natively allow you to implement advanced security capabilities not available in a tabular project. In a multidimensional project you can implement cell-level security to restrict access to a particular cell or group of cells in your model. Cell-level security is not provided in a tabular model.

In addition, multidimensional projects also enable you to control the use of visual totals, grant or deny permission to drill through to detail data, and create default members for each role.

In a multidimensional project, preaggregated summary values are calculated when data is processed into a model in order to improve query response times. For example, the Sales of All Products is a precalculated value. Dimension data security is applied after the data is processed, so even if a user is only granted permission to access the Bikes category, by default the value of Sales for All Products will be the sum of sales for Accessories, Bikes, Clothing, and so on. This may or may not be the value that you want members of the role to see. In this case, if you want the value of Sales for All Products to be limited to the value of Sales for Bicycles, you must enable visual totals. Enabling visual totals restricts the summary values so that they must be equal to the sum of the detail values that a role has permission to access. This change impacts query response time, because summary values must be calculated at query time. Tabular projects do not precalculate summary values, so summary values are always equal to the sum of detail values, that is, visual totals are always enabled in a tabular model.

In a multidimensional model you can enable permission to drill through to detail data on a role-by-role basis. In a tabular model, roles are not used to control access to drillthrough capability. Instead all roles are able to drill through to detail.

In a multidimensional model you can specify a default member for each attribute in a dimension. A default member behaves like a filter that is automatically applied. For example, if the default member of Year is 2012, then by default, only data for 2012 is displayed. However, a user can choose to see data from a different year or to see data for all years. In a multidimensional model, you can configure a default member for each attribute that applies to all roles or you can specify a different default member on a role-by-role basis. In a tabular model, you cannot specify a default value. Instead, if you want a default filter, you will need to configure that capability in your reporting and analysis tool.

# Summary

In SQL Server 2012 Microsoft introduced the Business Intelligence Semantic Model (BISM) to support a broad range of reporting and analysis needs. The two modeling experiences encapsulated in the BISM, multidimensional and tabular modeling, provide complementary features that enable you take advantage of capabilities that best meet your needs.

Tabular modeling provides a readily accessible modeling experience with capabilities that will satisfy most reporting and analysis needs. Most users are familiar with working with tables and relationships and quickly learn to implement business logic using the Excel-like DAX language. The ease of use and simplified and flexible modeling provided by the tabular experience means that solutions can be developed quickly. The in-memory, column-oriented xVelocity engine provides extremely fast query response for data sets that may contain billions of records. Tabular models support all of the reporting and analysis tools that generate MDX queries, such as Excel and Reporting Services, and they also support Reporting Services Power View, which generates DAX queries.

Multidimensional modeling provides extensive capabilities to help you manage your most complex and largest-scale BI challenges. The multidimensional data model combined with MDX provides out-of-the-box functionality so that you can create sophisticated models and implement complex business logic. On-disk data storage, precalculated aggregates, and in-memory caching enable multidimensional models to grow to multi-terabyte scale and provide fast query response. With cell-level security you can satisfy rigorous security requirements.

In summary, tabular modeling provides a simplified modeling experience with capabilities that should meet most of your reporting and analysis needs. If you require complex modeling, business logic, or security, or if you need a very large scale solution, multidimensional modeling may be the best solution for your needs.

The following table provides a summary comparison of the characteristics of multidimensional and tabular models.

| **Feature group** | **Decision criteria** | **Multidimensional/ Tabular** | **Multidimensional modeling** | **Tabular modeling** |
| --- | --- | --- | --- | --- |
|  | Time to solution | 🞉/● | Longer time to solution. | Shorter time to solution. |
|  | Learning curve | 🞉/● | Dimensional modeling and MDX language create a steeper learning curve but natively provide more complex capabilities. | Relational modeling and Excel-like DAX language create a less steep learning curve but complex capabilities may require sophisticated DAX expressions. |
| Data model | Data relationships | ●/🞉 | One-to-many.  Many-to-many.  Reference relationships must be explicitly modeled. | One-to-many.  Many-to-many requires DAX expressions.  Modeling table relationships creates reference relationships. |
| Data model | Hierarchies | ●/🞉 | Native support for standard, ragged, and parent-child hierarchies | Native support for standard hierarchies. Parent-child hierarchies require DAX expressions. |
| Data model | Additional data modeling features | ●/🞉 | Perspectives, translations, actions, drillthrough, stored procedures, and write-back. | Perspectives and drillthrough. |
| Business logic | Calculation language | ●/● | MDX | DAX |
| Business logic | Calculations | ●/🞉 | Native support for common and complex calculations. | Native support for common and many complex calculations. |
| Business logic | Aggregation functions | ●/🞉 | Sum, Count, Min, Max, Distinct Count, None, ByAccount, AverageOfChildren, FirstChild, LastChild, FirstNonEmpty, and LastNonEmpty. | Sum, Count, Min, Max, Average, DistinctCount, and various time intelligence functions like FirstDate, LastDate, OpeningBalanceMonth, and ClosingBalanceMonth. |
| Business logic | Hierarchy logic | ●/🞉 | Functions to navigate standard and parent-child hierarchies. | DAX functions to navigate parent-child hierarchies, DAX expressions to implement logic in standard dimensions. Hierarchy logic generally more difficult using DAX. |
| Business logic | KPIs | ●/● | Actual, goal, status, and trend with graphical indicators | Actual, goal, and status with graphical indicators. |
| Business logic | Currency conversion | ●/🞉 | Supports multi-currency conversion using the Business Intelligence Wizard. | Implement using DAX expressions. |
| Data access and storage | Scale | ●/🞉 | Extremely large scale (multi-terabyte) | Large Scale (Billions of records) |
| Data access and storage | Performance | 🞉/● | Indexes and preaggregated measure values stored on disk. Dimension data and query results cached in memory. Approximately 3x data compression. | In memory column-based data storage. Approximately 10x data compression. |
| Data access and storage | Data sources | 🞉/● | Relational databases. | Relational databases, Excel, Text, OData feeds, Azure Data Market, Analysis Services. |
| Data access and storage | Query language | ●/● | MDX | DAX  MDX (In-Memory mode only) |
| Data access and storage | Data storage | ●/● | MOLAP - Dimension, fact, and aggregated data stored on disk. Dimension data and query results cached in memory.  ROLAP – Dimension, fact, and aggregated data stored in a relational database. | In-Memory - All data cached in memory utilizing column-oriented xVelocity analytics engine  DirectQuery – Data stored in SQL Server 2012. |
| Data access and storage | Data compression | 🞉/● | Typically 3x. | Typically 10x. |
| Data access and storage | Client tools | ●/● | Excel, Reporting Services, Microsoft PerformancePoint, and other third-party client tools.  Reporting Services Power View supported in future SQL Server versions. | Reporting Services Power View, Excel, Reporting Services, PerformancePoint, and other third-party client tools. |
| Data access and storage | Programmability | ●/🞉 | XMLA, ASSL, ADOMD.NET, MSOLAP, AMO, Windows PowerShell for AMO. Developed for use with multidimensional models. | XMLA, ASSL, ADOMD.NET, MSOLAP, AMO, PowerShell for AMO. Available but less intuitive for use with tabular models. |
| Security | Security | ●/🞉 | Dimension member and cell-level security.  Dynamic Security. | Row-level security.  Dynamic Security. |

🞉 - Fewer capabilities.

● - More capabilities.

# For More Information

[SQL Server Website](http://www.microsoft.com/sqlserver/)

http://www.microsoft.com/sqlserver/

[SQL Server TechCenter](http://technet.microsoft.com/en-us/sqlserver/)

http://technet.microsoft.com/en-us/sqlserver/

[SQL Server DevCenter](http://msdn.microsoft.com/en-us/sqlserver/)

http://msdn.microsoft.com/en-us/sqlserver/

[Analysis Services and PowerPivot Team Blog](http://blogs.msdn.com/b/analysisservices/)

<http://blogs.msdn.com/b/analysisservices/>

[Analysis Services](http://msdn.microsoft.com/en-us/library/bb522607.aspx)

http://msdn.microsoft.com/en-us/library/bb522607.aspx

[Multidimensional Modeling (SSAS)](http://msdn.microsoft.com/en-us/library/hh230904.aspx)

http://msdn.microsoft.com/en-us/library/hh230904.aspx

[Tabular Modeling (SSAS Tabular)](http://msdn.microsoft.com/en-us/library/hh212945.aspx)

http://msdn.microsoft.com/en-us/library/hh212945.aspx

[Feature by Server Mode or Solution Type (SSAS)](http://msdn.microsoft.com/en-us/library/hh212940(v=sql.110).aspx)

http://msdn.microsoft.com/en-us/library/hh212940(v=sql.110).aspx

Did this paper help you? Please give us your [feedback](mailto:sqlfback@microsoft.com?subject=White%20Paper%20Feedback:%20Choosing%20a%20Multidimensional%20or%20Tabular%20Modeling%20Experience%20in%20SQL%20Server%202012%20Analysis%20Services). Tell us on a scale of 1 (poor) to 5 (excellent), how would you rate this paper and why have you given it this rating? For example:

* Are you rating it high because of having good examples, excellent screenshots,   
  clear writing, or another reason?
* Are you rating it low because of poor examples, fuzzy screenshots, or unclear writing?

This feedback will help us improve the quality of white papers we release.

Send [feedback](mailto:sqlfback@microsoft.com?subject=White%20Paper%20Feedback:%20Choosing%20a%20Multidimensional%20or%20Tabular%20Modeling%20Experience%20in%20SQL%20Server%202012%20Analysis%20Services).