



Microsoft®
SQL Server® 2012

Introducing the BI Semantic Model in Microsoft® SQL Server® 2012

SQL Server Technical Article

Writer: Peter Myers, Bitwise Solutions

Published: October 2012

Applies to: SQL Server 2012

Summary: This white paper describes the BI Semantic Model, a single model that serves all of the end-user experiences for Microsoft BI, including reporting analysis and dashboarding.

Copyright

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site 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. All rights reserved.

Contents

- Introduction 4
- Microsoft’s BI modeling strategy and vision..... 4
- Introducing the BI Semantic Model in SQL Server 2012 6
 - Data model..... 8
 - Business logic and queries 8
 - Data access..... 9
 - Scenario 1..... 9
 - Scenario 2..... 10
- Create a PowerPivot workbook in SQL Server 2012 11
- Create a BI Semantic Model with the tabular project template..... 14
- Choose the appropriate model development approach 14
 - Data model..... 15
 - Business logic 15
 - Data access and storage 16
- Conclusion..... 17
 - Flexibility 17
 - Richness 17
 - Scalability 17

Introduction

For well over two decades, the IT industry has recognized that businesses benefit from a model layer over their data sources. Model layers can deliver high-performance query responses even over extremely large volumes of data, and they can encapsulate business rules and effectively secure access to data. Data modeling capabilities were first delivered in Microsoft SQL Server 7.0, and with each release of the product, the data modeling capabilities have been improved upon. In SQL Server 2012, the BI Semantic Model—a new terminology and a set of existing and new capabilities—is available to produce data models.

This white paper has been written specifically to introduce the BI Semantic Model in SQL Server 2012. It starts by describing the evolution of the product through to the vision that has inspired the BI Semantic Model. It then explains the BI Semantic Model architecture, and describes—in broad terms—how tabular models can be developed. Finally it describes considerations to help you choose the best-fit approach for developing a BI Semantic Model.

Microsoft's BI modeling strategy and vision

Microsoft first introduced an OLAP server with SQL Server 7.0—with a product named OLAP Services—to meet this need in 1998. SQL Server became the first DBMS to include an OLAP server. It also introduced two technologies, OLE DB for OLAP and Multidimensional Expressions (MDX), which are currently the industry standards for accessing OLAP databases. SQL Server 7.0 OLAP Services also introduced innovative aggregation and storage technology that enabled customers to build highly responsive and scalable solutions at lower cost. OLAP Services also offered native integration with Microsoft Excel. With SQL Server 7.0 OLAP Services, Microsoft began a journey of democratizing access to business intelligence (BI) technology.

Just two years later, OLAP Services was enhanced to support richer dimension and cube definitions. It also became the first OLAP product to include data mining. The name was changed to Analysis Services to reflect these broadened capabilities. Over the five years that followed, SQL Server 2000 Analysis Services became a market leader due to its performance, accessibility, reduced total cost of ownership, and relative ease of solution development.

In 2005, Microsoft released the next generation of Analysis Services. In this significant update, the product delivered enterprise-class OLAP and data mining features. Specifically, advances to the OLAP feature resulted in a feature rich, scalable, and higher performance product that customers could use to solve increasingly sophisticated business problems with unprecedented levels of scale and complexity. With these richer capabilities, the term *Unified Dimensional Model (UDM)* was introduced to describe a model that could be created, consisting of multidimensional objects such as cubes and dimensions.

Analysis Services became the foundation of the Microsoft BI stack with its ability to host models that power all of the end-user experiences, including Microsoft Excel client PivotTables and PivotCharts, Excel Services, PerformancePoint Services, and SQL Server Reporting Services.

SQL Server 2008 Analysis Services continued to build on the foundation of the previous release and increased the bar on scalability and performance, while reducing the cost of building solutions.

In 2010 Microsoft released SQL Server 2008 R2. In this release, the Analysis Services product group produced two add-ins that extended the capabilities of Excel 2010 and Microsoft SharePoint Server 2010 to deliver self-service BI. These two add-ins allow business users to create and share models. The add-ins, referred to together as PowerPivot, have revolutionized how organizations define and use models:

- Business users can create and deploy their own data solutions.
- IT manages the models.

The PowerPivot for Excel add-in supports loading, exploring, relating, and enriching data. The data can come from traditional databases, like SQL Server and Oracle, or from other types of data sources, like OData feeds, Reporting Services reports, text files, and spreadsheets. The Excel add-in hosts the Microsoft SQL Server 2012 xVelocity (formerly known as VertiPac) analytics engine storage engine, which is a column-based data store that can load, manage, and query large volumes of data. Because the storage engine typically achieves 10x compression (depending on the structure of your data), it can efficiently store and query larger volumes of data than is natively supported in the Excel client. When the Excel workbook is saved, the model metadata and data is embedded into the workbook file, simplifying deployment and management.

To enrich the model, you can use a new expression language named Data Analysis Expressions (DAX). DAX is predominantly based on many existing and familiar Excel functions, but it also includes new functions that have never been relevant for use in Excel. You use DAX to define calculated columns that extend the tabular rowsets, and measures that are evaluated at query time.

The PowerPivot for SharePoint add-in provides a collection of server components that provide query processing and management control for PowerPivot workbooks that are published to SharePoint Server 2010. By using a special library template, the PowerPivot Gallery, business users can browse snapshots of Excel workbook reports, use the Excel client or Reporting Services to create new analytic experiences based on a workbook, and configure a data refresh policy for the PowerPivot workbook.

Corporate IT can monitor and manage the PowerPivot workbooks stored in SharePoint by using a Management Dashboard. This dashboard visually presents logged information about server health, quality of services, workbook activity, and data refresh history and failures.

The PowerPivot for Excel add-in is available for download from Microsoft at no cost.

For more information about PowerPivot, or to download the PowerPivot for Excel add-in, see <http://powerpivot.com>.

For more information about DAX, see the white paper titled [Data Analysis Expressions \(DAX\)](#)

[In the Tabular BI Semantic Model](http://www.microsoft.com/download/details.aspx?id=28572) (<http://www.microsoft.com/download/details.aspx?id=28572>).

Moving again from strength to strength, Analysis Services is now in its sixth version. SQL Server 2012 Analysis Services has broadened its capabilities, features, and scope of reach. To define the vision for this next generation product release, the Analysis Services product group decided on a set of key guiding principles:

- Build on the strengths and success of Analysis Services.
- Expand its reach to a much broader base.
- Continue to make use of its rich existing ecosystem of tools, developers, and partners.

The Analysis Services product group understands that key to broader adoption is to embrace the relational data model. Because the relational data model is well understood by developers and IT professionals today, it makes sense to bring it into the fold of Analysis Services.

At the same time, the product group has carried forward the existing OLAP capabilities (the UDM) to ensure that customers' existing investments and skills remain relevant. With the recognition that not all BI applications have the same requirements, users can still build solutions that have advanced needs in terms of scale, performance, and complexity. New capabilities support solutions that need to be developed quickly. This combination of old and new helps meet the diverse needs of all BI applications.

This vision for Analysis Services delivers flexibility. First, it brings together the tabular and multidimensional development approaches under a single BI platform to deliver the best of both worlds and choice to the developer. Second, there is choice in terms of scenarios, delivering capabilities for self-service and corporate BI solutions. Microsoft calls this vision the *BI Semantic Model*.

Introducing the BI Semantic Model in SQL Server 2012

The BI Semantic Model is a single model that serves all of the end-user experiences for Microsoft BI, including reporting analysis and dashboarding. The model can integrate data from a number of data sources, whether they are traditional data sources, such as databases or LOB applications, or nontraditional sources, such as OData feeds, text files, and spreadsheets.

The following tools enable users to enrich models:

- Calculations to encapsulate business rules.
- Multilevel hierarchies to support navigation and the summarization of measures.
- Key performance indicators (KPIs) to monitor performance.
- Perspectives to provide simplified views of a model.

The model can accelerate access to the data, and then it can deliver query results using a data model experience that is appropriate for a variety of client tools, including those that support interactive analysis, data visualization, reporting, scorecards, dashboards, and custom applications. The BI Semantic

Model enables users to work with the data in all of these ways regardless of how the model was developed.

The BI Semantic Model powers both the entire range of client tool experiences and the full spectrum of BI applications that business users and developers can build using the Microsoft BI stack. This spectrum can include the following scenarios:

- Personal BI applications that business users create to meet their own specific needs
- Team BI solutions that business users create and share with their colleagues within small organizations or departments
- Corporate BI applications that are developed, managed, and sanctioned by corporate IT and rolled out to a large user base

Figure 1 shows the different Microsoft BI tools that you can use in each scenario.

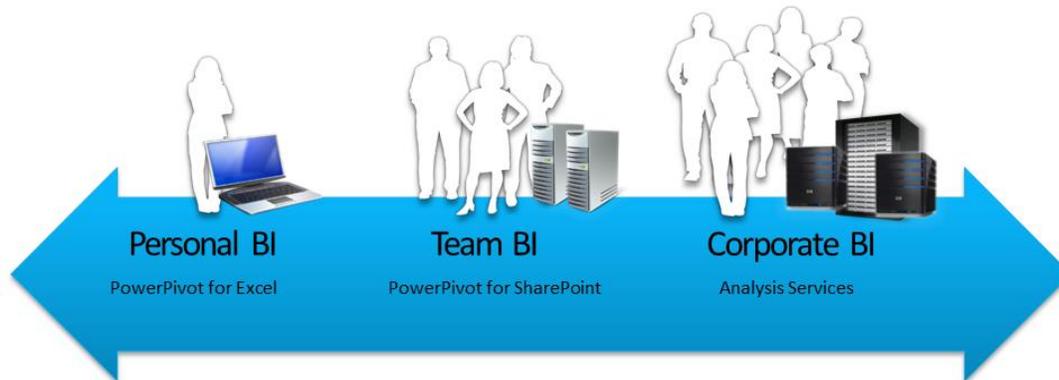


Figure 1
Analysis Services powering the entire spectrum of BI applications

Regardless of where the BI solution falls within this spectrum, the BI Semantic Model is the one model—the one technology—that powers all end-user experiences.

The BI Semantic Model can migrate from personal BI, through to team BI, to corporate BI. A single product powers all of these capabilities.

BI Semantic Model

The BI Semantic Model can consume data from traditional data sources, including SQL Server, Access, Oracle, Teradata, and DB2. In addition, tabular models can consume data from a wider variety of data sources, extending to Excel workbooks, data files, OData feeds, and cloud services, including Windows Azure SQL Database.

The model then enriches and provides accelerated access to the data, and it exposes an intuitive interface that is surfaced by all tools in the Microsoft BI stack: Reporting Services (including the new interactive report authoring tool named Power View), Excel, PowerPivot, and SharePoint.

For more information, see [Power View](http://technet.microsoft.com/library/hh213579(SQL.110).aspx) (http://technet.microsoft.com/library/hh213579(SQL.110).aspx) on the Microsoft TechNet website.

The model can be thought of conceptually in three layers: data model, business logic and queries, and data access.

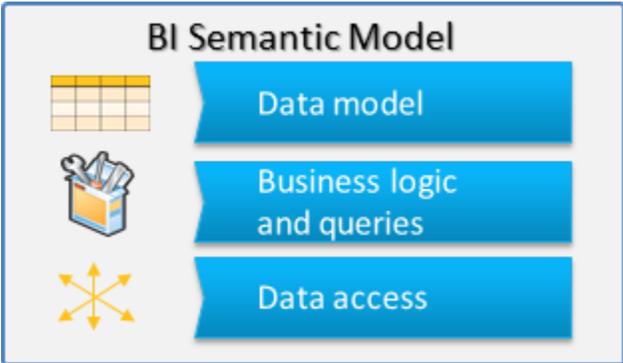


Figure 2
The conceptual layers of the BI Semantic Model

Data model

The data model can be explained from both the data modeler’s and model consumer’s perspective.

First, from the data modeler’s perspective, to develop a tabular model the model developer uses Excel 2010 or SQL Server Data Tools. The tabular model consists of tables, columns, and relationships. For the multidimensional project type, the data model is developed in SQL Server Data Tools and it consists of dimensions and measures.

Second, from the model consumer’s perspective, the model can be developed as a tabular model but consumed as a multidimensional model. This flexibility is achieved because the tabular model exposes a multidimensional interface like a cube. Existing cube client tools like Excel PivotTables function for both tabular and multidimensional models.

Business logic and queries

The BI Semantic Model offers both MDX and DAX for querying. MDX is a well-known calculation and query language that dates to the beginning of Analysis Services. MDX is based on multidimensional constructs (cubes, dimensions, hierarchies, levels, members, and so on). In contrast, DAX is a newer expression language that has evolved into a query language in the SQL Server 2012 release. DAX was

introduced to enrich PowerPivot models in SQL Server 2008 R2. It is based on a subset of Excel functions. It uses tabular constructs (tables, columns, and relationships) and includes functionality to support model development, including relationship navigation, context modification, and time intelligence. Importantly, DAX lowers the barriers to entry for defining sophisticated business logic in tabular models.

In the initial release, the tabular models support both MDX and DAX queries, while multidimensional models support only MDX queries. Microsoft plans to make it possible to query multidimensional models with DAX.

Data access

The BI Semantic Model offers developers the option to either cache the data or have it passed through to the underlying data source. Both MOLAP and ROLAP storage modes support this functionality for multidimensional models.

In SQL Server 2012, the In-Memory storage mode is used for caching the tabular model data. xVelocity is an in-memory column store engine that caches all of the data in memory, organizes it by column, and uses state-of-the-art compression and querying techniques. xVelocity provides extremely rapid query performance without the need for indexing or aggregations, because it simply performs brute-force scanning of data in memory.

The xVelocity in-memory analytics storage engine is the same storage engine that powers PowerPivot for Excel, PowerPivot for SharePoint, Analysis Services in SQL Server 2012 (for tabular models), and the new columnstore index, which is available in the SQL Server 2012 database engine. They are all based on the same technology.

Finally, for tabular models, there is the DirectQuery storage mode. DirectQuery is a pass-through mode to a backend data store. In SQL Server 2012, DirectQuery is supported only for models that are based on a single SQL Server relational data source.

To emphasize the flexibility of the BI Semantic Model architecture, consider the following two scenarios.

Scenario 1

A model developer produces a BI Semantic Model using the tabular project type in SQL Server Data Tools. This model is based on several data sources to enable sales analysis. The developer then enriches the model by using DAX to define business logic in the form of calculated columns and measures. The model data is cached in the in-memory column store. A business user then connects to this model by using Power View. By sending DAX queries to the server to request a rowset, Power View can query the model definition and data through the tabular interface.

Now the same model can be consumed by using the Excel client. In this version of the scenario, nothing has changed in terms of what the model developer produced and deployed. However, the business user is using a tool that requires a multidimensional view of the model. Excel can consume the model like a cube, and it sends MDX queries to the server to request a cellset.

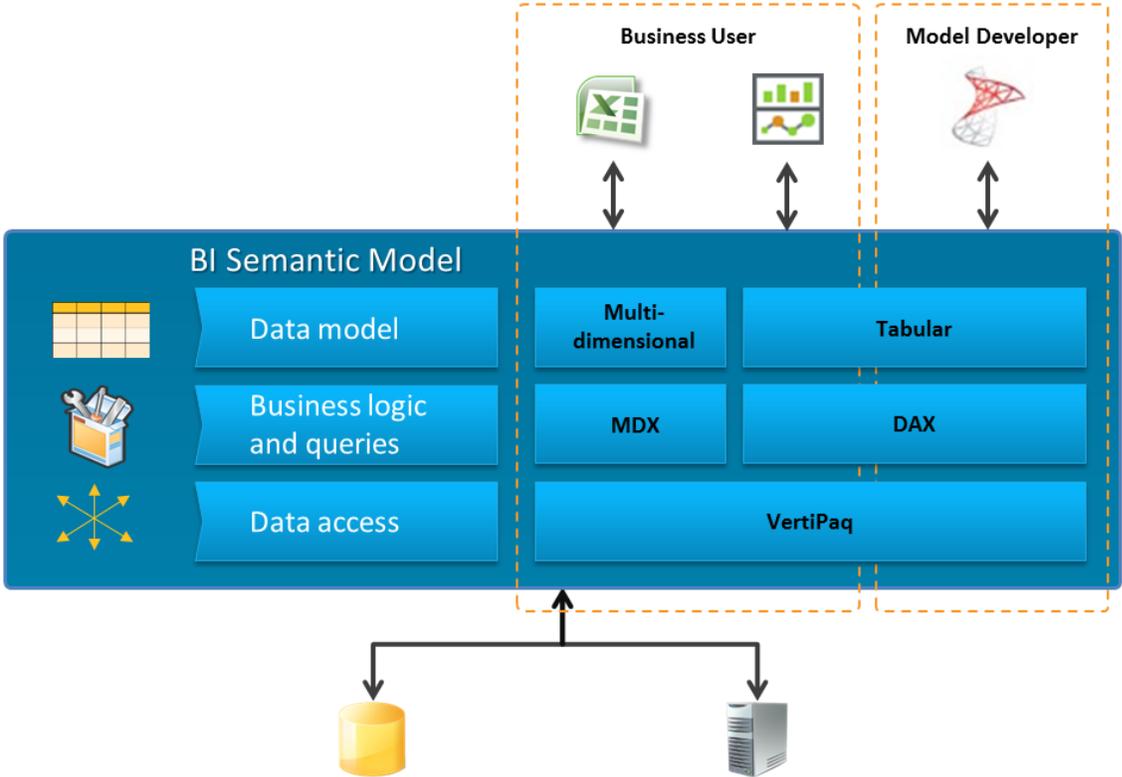


Figure 3
Scenario 1

Scenario 2

The model developer has used the classic multidimensional modeling experience, previously referred to as the UDM, to build a BI Semantic Model using the multidimensional project type. The developer has defined an MDX script to enrich the model with calculations (calculated members, scopes and assignments, and named sets). The model’s data is cached in a MOLAP store. Business users can use the Excel client to connect to the model, and the client tool consumes the multidimensional view of the model.

Now business users can consume the same model by using Power View. Even though the model developer has developed the model using MDX, Power View can consume it through the tabular interface by sending DAX queries. Microsoft plans to make this capability available in an update of SQL Server 2012.

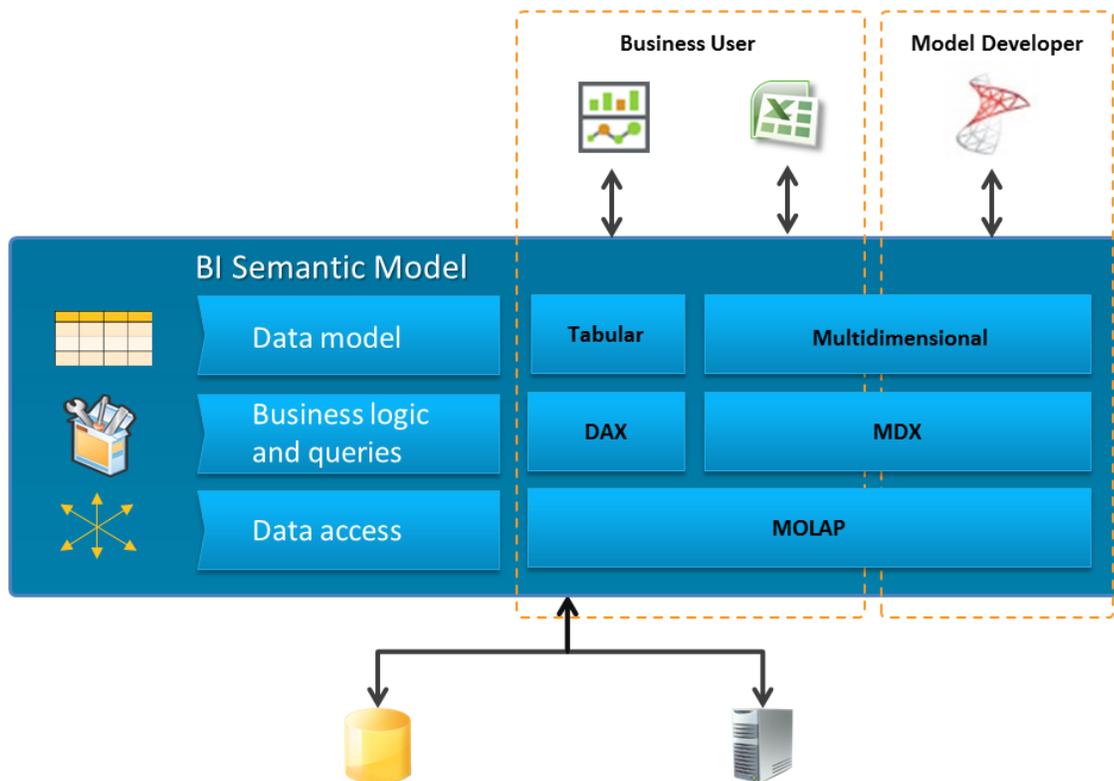


Figure 4
Scenario 2

Both scenarios emphasize the flexible nature of the BI Semantic Model. Model developers can choose to develop models in the way that works best for them, and client tools consume them in the way that works best for the consumption experience.

Create a PowerPivot workbook in SQL Server 2012

PowerPivot for Excel has been upgraded in the SQL Server 2012 release to include an improved design experience and additional features and functionality.

The model developer can work in diagram view or grid view. The diagram view represents a metadata view of the model, and the grid view represents a data view of the model. Some tasks can be achieved in both views. Model developers can choose the view they prefer. However, data enrichment operations that use DAX, such as adding calculated columns and defining measures, are available in data view only. Metadata-driven operations such as the creation of multilevel hierarchies are available in diagram view only. Figures 5 and show a model and a table in diagram view.

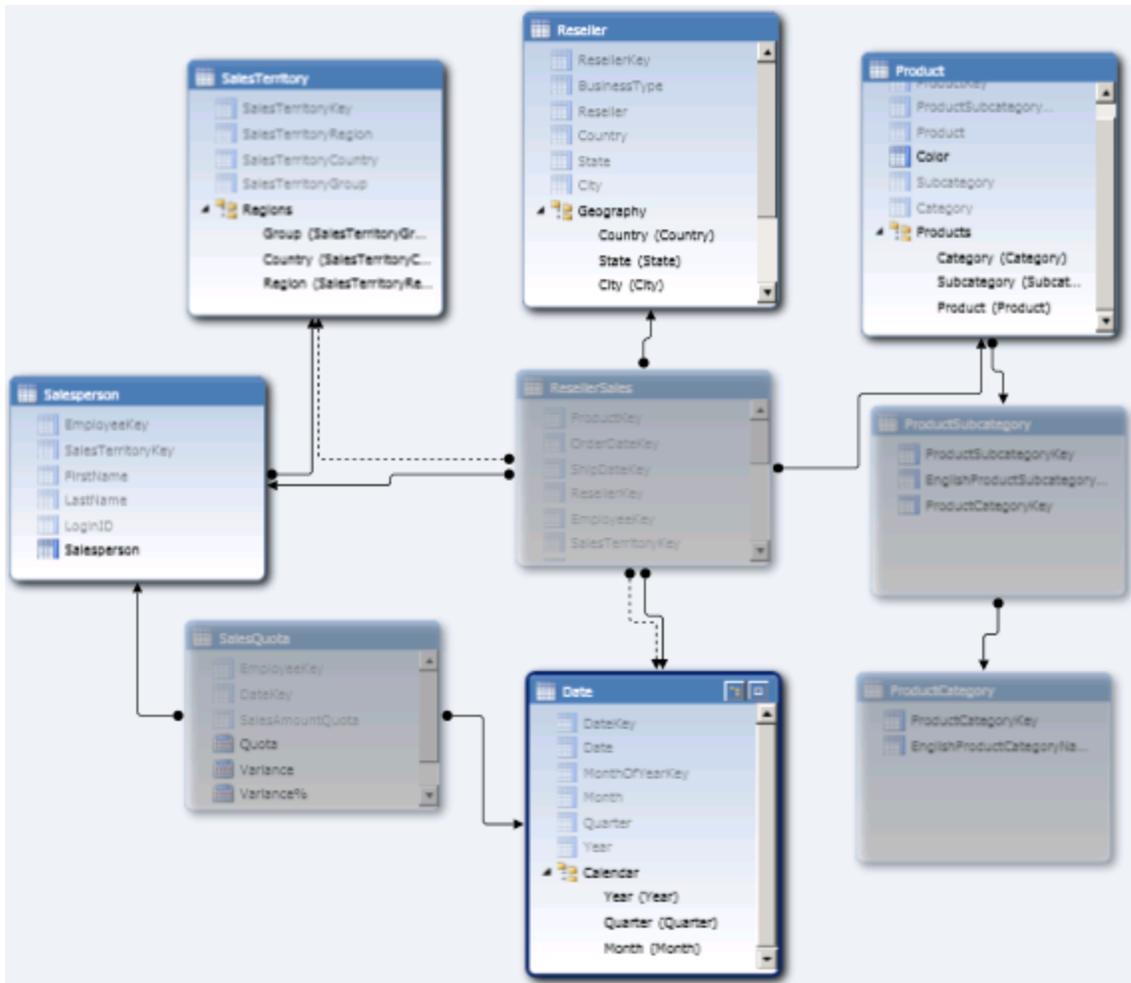


Figure 5
A model expressed in diagram view

Column Name	Icon
EmployeeKey	Key icon
DateKey	Key icon
SalesAmountQuota	Table icon
Quota	Measure icon
Variance	Measure icon
Variance%	Measure icon
Sales Performance	KPI icon

Figure 6
A table in diagram view consisting of three hidden columns, three measures, and a KPI

The tabular view also includes a new calculation area where you define and preview model measures and KPIs.

EmployeeKey	DateKey	SalesAmountQuota	Add Column
272	20050701	\$10,000.00	
272	20051001	\$15,000.00	
272	20060101	\$50,000.00	
272	20060401	\$60,000.00	
		Quota: 138,395,000	
		Variance: -138,395,000	
		Variance%: -100.00 %	
		Sales Performance: 	

Labels in image: Data Grid (points to the first four rows), Calculation Area (points to the summary rows).

Columns below table: SalesQuota, Product, ProductCategory, Salesperson, ProductSubcategory, SalesTerritory

Figure 7
The SalesQuota table in grid view

There are numerous new capabilities and DAX functions. It is now possible to:

- Define distinct count measures.
- Define multilevel hierarchies.
- Define and format measures.
- Define KPIs.
- Add descriptions to tables, columns, measures, and KPIs.
- Define a time table with a date column (which must be based on a Date column type) so that integer key values can be used in fact tables.
- Sort column values by another column in the same table.
- Define multiple relationship paths (direct or indirect) between two tables.
- Define perspectives to support subject-specific subsets of the entire model structure.
- Define binary columns (useful for storing images).
- Define reporting properties to optimize the presentation of data in reporting applications.

As was the case in the first release of PowerPivot, the PowerPivot workbook can be deployed to a SharePoint Server 2010 library to secure, schedule data refresh for, and share access to the model. An additional function in the PowerPivot Gallery enables you to create a Power View report.

SQL Server 2012 also includes a new function that restores a PowerPivot workbook to an instance of Analysis Services in tabular mode. In this scenario, the server can host the model in environments without SharePoint Server 2010.

Another reason to import the workbook to a server instance is to overcome the 2 GB size limit imposed by an Excel workbook. Further, you can use the import process to configure features that are not supported in PowerPivot, such as roles for row-level security and table partitions for more efficient table data refresh. The next section discusses these additional features.

Create a BI Semantic Model with the tabular project template

While SQL Server Data Tools and PowerPivot share model designers, these environments are optimized for their audiences. SQL Server Data Tools, for example, provides a project development approach designed for IT professionals, whereas PowerPivot provides deep integration with Excel 2010 for business users.

Generally the solutions that IT professionals produce have greater needs in terms of size, scalability, and security. SQL Server Data Tools project templates support another important requirement, source control integration.

SQL Server 2012 offers four additional capabilities for IT professionals who are developing tabular models.

First, the data volume sizes can exceed the 2 GB limit imposed by Excel 2010. For tabular projects, size is limited by the server capacity in terms of disk space and memory. It is important to understand that tabular project development depends on an instance of Analysis Services. Note that this is not a requirement for PowerPivot because the xVelocity in-memory analytics engine is installed with the PowerPivot for Excel Add-in.

Second, the DirectQuery storage mode for query pass-through can only be configured in a tabular project.

Third, you can define table partitions to support data refresh on a subset of the table's data. After the database is deployed to a server, database administrators can process the database tables or partitions; create, configure, and delete partitions; and merge partitions.

And fourth, you can define roles. Roles define permission sets for users and groups, and they can be used to limit access to specific data. You can write dynamic and static expressions in DAX to define row filters for the model tables.

You can also import a PowerPivot workbook to create a new tabular project. For example, a business user might create a solution using PowerPivot. By restoring the workbook, IT can add features that are not supported in PowerPivot, such as roles to grant read permission to different groups of users. Additionally, recognition by IT that the model is important to the organization may prompt them to import the workbook into a tabular project and take ongoing responsibility for developing the model.

Choose the appropriate model development approach

For business users, the appropriate design experience is likely to be PowerPivot for Excel.

Multidimensional projects or tabular projects are usually the best options for IT professionals. The choice depends on the needs of each of the three conceptual layers (data model, business logic, and

data access and storage). In some cases, you may need to make tradeoffs to determine what will work best for both the model developer and the required solution.

SQL Server Data Tools includes a tabular project template, which enables you to develop new models using DAX and the In-Memory or DirectQuery storage modes. You can use the multidimensional project template to develop a new model by defining multidimensional objects (dimensions and cubes). In this template, the cube's MDX script is used to define calculations, and the cube's partitions are defined to configure the MOLAP or ROLAP storage modes.

For more information about the features available for each development approach, see [Feature by Server Mode or Solution Type](http://technet.microsoft.com/library/hh212940(SQL.110).aspx) ([http://technet.microsoft.com/library/hh212940\(SQL.110\).aspx](http://technet.microsoft.com/library/hh212940(SQL.110).aspx)) on TechNet.

Data model

If you want to work with familiar relational concepts to produce the model, you can choose the tabular project template. This almost always results in a simplified development effort and faster time to solution. While many advanced concepts available in multidimensional projects are not natively supported, it is still possible to implement some of them. For example, parent-child hierarchies and many-to-many relationships can be implemented by using DAX calculations.

A further consideration for choosing the tabular project template is that it is simple and fast to wrap a model over a raw database to support reporting and analytics. By importing the required tables, ensuring all relationships are defined, defining key measures, and perhaps renaming or hiding certain columns, you can develop an intuitive, interactive, and highly responsive model.

If your skillset already includes developing multidimensional models, this may be the natural way to continue producing models. However, it can be more difficult to design, develop, enrich, and optimize multidimensional models than to build a tabular model. Also, it usually takes longer to produce and deploy a multidimensional model.

In contrast, the multidimensional model allows you to work with advanced concepts, including native parent-child hierarchies, native many-to-many relationships, dimension attribute relationships, extremely sophisticated calculations, cube and dimension write-back, actions, translations, and more. If your model requires any of these advanced structures, your choice of development approach is simple: a multidimensional model is the only model type that supports them.

Business logic

It is important to understand that MDX is sophisticated than DAX. Although DAX keeps simple problems simple, it cannot always address complex problems.

DAX is based on Excel formulas and relational concepts, and it has a lower initial learning curve than MDX. However, complex DAX calculations can be challenging for model authors, and they require an understanding of advanced DAX concepts such as filter context.

In contrast, MDX is based on multidimensional concepts, so it involves a higher initial learning curve. It can be used to define complex solutions and is ideally suited for models that need the power of multidimensional expressions. Common examples include planning, budgeting, and forecasting models. MDX can define calculated members (including measures), KPIs, scopes and assignments, and named sets. With the one notable exception of measures, DAX cannot define or simulate any of these types of constructs.

Data access and storage

Tabular models support two storage modes: In-Memory and DirectQuery.

The In-Memory storage mode is an in-memory column store that typically produces 10x compression or greater. It achieves high performance by default without the need for tuning because the server performs brute-force memory scans. The xVelocity in-memory analytics engine uses the server's resources to perform scans across the data to isolate what is needed, and it exploits not just the memory available on the machine but also all of the cores. Although there is basic paging support, to ensure best performance, you should ensure that server resources can cache the entire model in memory.

The DirectQuery storage mode essentially translates DAX queries to the underlying data source, and it can exploit the backend database capabilities. However, there are a number of caveats. In SQL Server 2012, the model must be based on a single SQL Server relational data source, and the model must not contain copy-and-paste data or calculated columns. Roles with row filters cannot be defined, so row-level security must be implemented at the data source. Additionally, the model can only be queried with DAX.

The benefits of the DirectQuery storage mode are that the model can query the latest data and leverage SQL Server security.

Note: If the SQL Server relational tables have columnstore indexes built, the xVelocity in-memory analytics engine is employed at the data source.

Multidimensional models support two basic choices of storage modes: MOLAP and ROLAP.

The MOLAP storage mode is a disk-based store that typically achieves 3x compression. This storage mode performs disk scans with in-memory subcube caching, and aggregation designs and tuning are typically required. Because there is extensive paging support, MOLAP supports models with extremely high data volumes that can scale to multiple terabytes in size.

In contrast, the ROLAP storage mode passes through query requests to the underlying data source. Although it supports most relational data sources, it does not support aggregations except those made available by SQL Server relational indexed views. DirectQuery always results in a single query sent to the underlying data source. With ROLAP, Analysis Services can split query processing over multiple requests to the data source and perform post processing.

Conclusion

Analysis Services in SQL Server 2012 supports one data model to deliver rich, interactive, scalable, and rapid access to data. This model, the BI Semantic Model, provides flexibility, richness, and scalability.

Flexibility

To define business logic, you can choose either tabular modeling with DAX or multidimensional modeling with MDX. Both experiences support data caching and pass-through storage modes. Whatever design approach you choose, the business user and client tool experience remain the same.

Richness

Both modeling experiences support rich capabilities and advanced constructs, including perspectives. They enable you to encapsulate sophisticated business logic by using DAX or MDX. Both models can be secured using fine-grained permissions sets to the row level or the cell level.

Scalability

The tabular model can deliver high performance based on an in-memory data store, and the multidimensional model can achieve similar results with a MOLAP data store. For access to the latest data in the data sources, DirectQuery and ROLAP support for pass-through modes for tabular and multidimensional models, respectively. Both model types can scale to exploit the resources of large enterprise servers.

In summary, the development approach taken to produce a BI Semantic Model is the choice of the model developer. Model developers can continue to develop sophisticated, high-performing multidimensional models, as they have been able to since the SQL Server 2005 release, or they can take the new, more simplified and rapid approach by developing tabular models.

For more information:

<http://www.microsoft.com/sqlserver/>: SQL Server Web site

<http://technet.microsoft.com/en-us/sqlserver/>: SQL Server TechCenter

<http://msdn.microsoft.com/en-us/sqlserver/>: SQL Server DevCenter

Did this paper help you? Please give us your feedback. 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 due to having good examples, excellent screen shots, clear writing, or another reason?
- Are you rating it low due to poor examples, fuzzy screen shots, or unclear writing?

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

[Send feedback.](#)