

Using DirectQuery in the Tabular BI Semantic Model

SQL Server Technical Article

**Writer:** Cathy Dumas, Business Intelligence Consultant, Avantage Partners.

**Contributors:** Marius Dumitru, Akshai Mirchandani, Edward Melomed, Karen Aleksanyan, Bob Meyers, Siva Harinath

**Technical Reviewers:** TK Anand, Greg Galloway, Marius Dumitru, Edward Melomed, Willfried Färber, Alberto Ferrari

**Published:** April 2012

**Applies to:** SQL Server 2012

**Summary:** This paper introduces DirectQuery, a mechanism for directly querying a SQL Server data source for Analysis Services tabular models. Readers will learn when and how to use DirectQuery in SQL Server 2012.

# ****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.

Some examples depicted herein are provided for illustration only and are fictitious.  No real association or connection is intended or should be inferred.

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.

Table of Contents

[Copyright 2](#_Toc320705913)

[Introduction 5](#_Toc320705914)

[Sample data 5](#_Toc320705915)

[DirectQuery benefits 6](#_Toc320705916)

[DirectQuery limitations 6](#_Toc320705917)

[DirectQuery architecture 7](#_Toc320705918)

[How the DirectQuery architecture differs from ROLAP 8](#_Toc320705919)

[Creating a DirectQuery enabled model 8](#_Toc320705920)

[Creating a new DirectQuery enabled model 9](#_Toc320705921)

[Converting an existing model to a DirectQuery enabled model 12](#_Toc320705922)

[Choosing a DirectQuery mode 14](#_Toc320705923)

[Partitioning a DirectQuery enabled model 15](#_Toc320705924)

[Creating a DirectQuery only partition 16](#_Toc320705925)

[Adding partitions to a DirectQuery model 18](#_Toc320705926)

[Changing the DirectQuery partition 18](#_Toc320705927)

[Setting the DirectQuery impersonation settings 19](#_Toc320705928)

[Data warehousing and data modeling techniques for DirectQuery 21](#_Toc320705929)

[Using a columnstore index 21](#_Toc320705930)

[Loading data efficiently during ETL 22](#_Toc320705931)

[Performing calculations at the column level 22](#_Toc320705932)

[Designing date and time calculations for DirectQuery 23](#_Toc320705933)

[Performing period to date calculations using basic DAX 24](#_Toc320705934)

[Widening date tables to include previous and next periods 24](#_Toc320705935)

[Securing the data warehouse 26](#_Toc320705936)

[Using SQL Server Data Tools effectively in DirectQuery mode 26](#_Toc320705937)

[Loading data efficiently into SQL Server Data Tools 27](#_Toc320705938)

[Adding more tables to a DirectQuery enabled model 27](#_Toc320705939)

[Managing the xVelocity engine cache in SQL Server Data Tools 27](#_Toc320705940)

[Deleting the workspace database 28](#_Toc320705941)

[Connecting to a DirectQuery enabled model 29](#_Toc320705942)

[Connecting directly to a DirectQuery enabled model using an MDX client 29](#_Toc320705943)

[Connecting to a DirectQuery enabled model using a BISM file 30](#_Toc320705944)

[Connecting to a DirectQuery enabled model using a RSDS file 30](#_Toc320705945)

[Preparing the DirectQuery deployment environment 31](#_Toc320705946)

[Machine topology 31](#_Toc320705947)

[Capacity planning for Analysis Services 33](#_Toc320705948)

[Managing a DirectQuery enabled model in SQL Server Management Studio 33](#_Toc320705949)

[Changing the DirectQuery mode 34](#_Toc320705950)

[Changing the DirectQuery partition in SQL Server Management Studio 34](#_Toc320705951)

[Changing the DirectQuery partition to a DirectQuery only partition 36](#_Toc320705952)

[Browsing a DirectQuery enabled model in SQL Server Management Studio 36](#_Toc320705953)

[Changing the DirectQuery impersonation settings 37](#_Toc320705954)

[Clearing the xVelocity engine cache 38](#_Toc320705955)

[Monitoring a DirectQuery enabled model using SQL Server Profiler 39](#_Toc320705956)

[Security considerations 39](#_Toc320705957)

[Security design patterns 39](#_Toc320705958)

[Managing user access to the model 41](#_Toc320705959)

[Enabling constrained delegation 41](#_Toc320705960)

[Performance considerations 42](#_Toc320705961)

[Conclusion 44](#_Toc320705962)

[Additional resources 44](#_Toc320705963)

# Introduction

With the release of Microsoft SQL Server 2012, SQL Server Analysis Services introduces the BI Semantic Model. The BI Semantic Model includes both traditional multidimensional models developed by BI professionals and deployed to Analysis Services, and the new tabular model.

By default, data in tabular models is processed, compressed using the xVelocity in-memory analytics engine (VertiPaq), and stored in an Analysis Services database. The xVelocity engine is an in-memory columnar storage engine that is optimized for high performance analysis and exploration of data. Tabular models are queried using Multidimensional Expressions (MDX) or Data Analysis Expressions (DAX). The xVelocity engine provides fast query times for aggregation queries even on large fact tables.

Not all scenarios are suited for the xVelocity engine. Although the compression ratio for the xVelocity engine is very good (10x compression is typical, with a range between 3x and 100x), some data sets are too large to fit in memory. The xVelocity engine also requires data to be processed into the cache, which means new data is not available in real time. The data in Analysis Services must also be secured, using a different security model from the SQL Server data source.

Analysis Services offers a solution to these problems in DirectQuery. When a tabular model is running in DirectQuery mode, data is not stored in the cache of the xVelocity engine. Rather, queries are executed directly on the relational data source. The benefits of DirectQuery include access to real-time data, increased size of the data sets that can be queried, reduced memory use by Analysis Services, improved Analysis Services start times, and access to the more granular SQL Server security model.

This paper shows how and when to use DirectQuery mode for tabular models. It describes design considerations, shows some data warehouse design patterns, and demonstrates how SQL Server Data Tools, SQL Server Management Studio, and SQL Server Profiler are used to build, secure, manage, and monitor DirectQuery enabled models.

## Sample data

The examples in this white paper are based on the AdventureWorksDW2012 data file. The AdventureWorks Internet Sales Tabular Model SQL Server 2012 sample project is also used for demonstration purposes. These samples are available for download at <http://msftdbprodsamples.codeplex.com/releases/view/55330>.

# DirectQuery benefits

DirectQuery has a number of benefits:

* New data can be retrieved in real time. Loading data into the tabular model is not required.
* Larger data sets can be used. If your source data in the SQL Server data source cannot be compressed into a 1 terabyte or smaller Analysis Services database, consider DirectQuery. Also, if your Analysis Services database does not fit in half of the memory on the machine hosting the Analysis Services instance, consider DirectQuery.
* The SQL Server security model can be used. The SQL Server security model offers some features that Analysis Services does not, for example cell level security compliance with HIPPA data encryption requirements, and auditing at the SQL Server level when individuals execute queries at the data source. For more information, see “Additional resources.”
* Memory and CPU resources need not be allocated to Analysis Services for caching, processing, and querying data. Analysis Services is not CPU intensive in DirectQuery mode. Analysis Services uses some memory in DirectQuery mode when it computes intermediate results, but the memory requirements are much smaller than when it queries the cache of the xVelocity engine.
* Metadata discovery operations are faster, because data for a DirectQuery enabled model need not be loaded into memory to complete the discovery operation. This speeds operations like expanding databases in the Object Explorer in SQL Server Management Studio.

# DirectQuery limitations

There are a few very important design considerations if you are planning to use DirectQuery:

* MDX queries are not supported for a model in DirectQuery mode. This means you cannot use Microsoft Excel or Microsoft PerformancePoint as reporting clients on top of a DirectQuery-only model, because these clients all issue MDX queries.
* As of this writing, the only production-ready reporting client that has a graphical user interface that constructs DAX queries is Power View. Power View should be considered a prerequisite for any DirectQuery implementation in SQL Server 2012.
* Only SQL Server 2005 and later data sources are supported. The preferred data sources are SQL Server 2012 and SQL Server 2008 R2, because these two data sources have superior performance and quality in DirectQuery scenarios.
* Only one data connection can exist in the model. You cannot query two or more SQL Server instances from a DirectQuery enabled model.
* DirectQuery enabled models cannot be created in PowerPivot. Only models deployed to a standalone Analysis Services instance can use DirectQuery.

There are some other restrictions in DirectQuery– calculated columns are not supported, row security is not supported, and some DAX functions are not supported. For more information, see [DirectQuery Mode](http://msdn.microsoft.com/en-us/library/hh230898(v=sql.110).aspx) (http://msdn.microsoft.com/en-us/library/hh230898(v=sql.110).aspx). However, the data warehouse and the data model can be designed around these restrictions, so that equivalent (or better) functionality is provided by the SQL Server data source and in the tabular model. More information about these data warehousing and modeling techniques appears later in this paper.

Analysis Services is optimized for use with the xVelocity cache. Queries typically perform better if they are answered from the cache. Your preference when data modeling should be to use the in-memory cache for your tabular model. DirectQuery is a better choice when you are using Power View and your scenario requires one or more of the benefits offered by DirectQuery.

# DirectQuery architecture

DirectQuery works by translating DAX queries to SQL. The following diagram shows the basic DirectQuery architecture.



**Figure 1:** DirectQuery architecture

Clients issue DAX queries through Power View, SQL Server Management Studio, AMO, ADOMD.NET, or through the Analysis Services OLE DB Provider. When Analysis Services receives a DAX query, the DAX formula engine formulates a query plan based on the supplied query. The DAX query is then translated to SQL by a dedicated DirectQuery engine. The translated query is then executed by SQL Server and results returned to Analysis Services, which returns the provided query results to the end user.

## How the DirectQuery architecture differs from ROLAP

Those familiar with multidimensional modeling should be familiar with relational OLAP (ROLAP), which is the multidimensional solution for allowing real-time access to relational data sources. DirectQuery uses a fundamentally different architecture, which provides a different set of capabilities.

DirectQuery has a dedicated engine for executing queries on the relational data source. After a DAX query has been translated to SQL, SQL Server performs the computations required to return query results. The DirectQuery engine can optimize SQL queries to reduce the number of queries issued, many times translating a DAX query to a single SQL query.

ROLAP does not have a dedicated engine for real-time queries. Instead, ROLAP uses the same formula engine and storage engine as multidimensional OLAP (MOLAP) for answering queries. When an MDX query is issued on a ROLAP-enabled multidimensional model, the formula engine computes the required data set and data is processed on the fly into an in-memory data cache. The Analysis Services storage engine then computes the results from this in-memory cache and returns the results to the end users.

DirectQuery requires fewer machine resources compared with ROLAP, because data is not cached in memory. Also, DirectQuery supports impersonating the current user’s credentials when connecting to SQL Server, which enables the SQL Server security model to be used. ROLAP does not support impersonating the current user’s credentials. Also, because queries on the relational data source are optimized, simple DAX queries on DirectQuery enabled models may perform better than similar MDX queries requiring relational data source access on ROLAP enabled models.

ROLAP provides support for data sources other than SQL Server, such as Oracle and Teradata. Time intelligence functions are supported in ROLAP. Neither of those features is supported for DirectQuery. MOLAP and ROLAP results may be combined in response to a single MDX query, whereas DirectQuery does not support mixed real-time and cached data access. MDX query results may be cached, so not all MDX queries issued to a ROLAP enabled model require a query to be issued on the relational data source. ROLAP may perform better than DirectQuery in situations where query results are cached.

# Creating a DirectQuery enabled model

DirectQuery enabled models are created in SQL Server Data Tools, because PowerPivot does not support DirectQuery. There are two ways to create a DirectQuery enabled model. A brand new model can be set to DirectQuery mode. This is the preferred approach. A pre-existing model can also be changed into a DirectQuery enabled model in SQL Server Data Tools.

**Note** Although it is possible to change a model into a DirectQuery enabled model in SQL Server Management Studio, this approach is not recommended. If your model requires changes so that it is compatible with DirectQuery, those changes must be made in SQL Server Data Tools and the model redeployed.

After SQL Server Data Tools is in DirectQuery mode, you can continue importing tables, creating relationships, creating measures, and so on. The development steps that are different for DirectQuery enabled models are:

* Setting the storage mode for the model (DirectQuery only vs. hybrid).
* Setting the DirectQuery impersonation settings.
* Configuring partitions for DirectQuery.

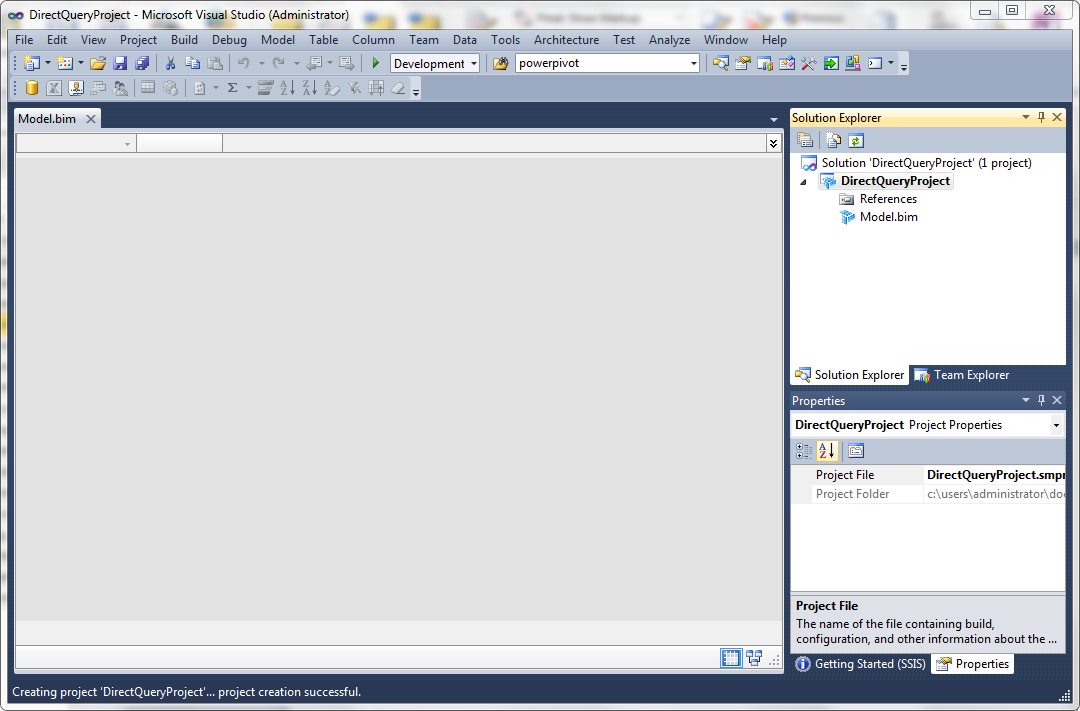
After these tasks have been completed, the model can be deployed and tested using real-time data access.

This section of the white paper shows you how to create a basic DirectQuery enabled model using the AdventureWorksDW2012 SQL Server data source.

## Creating a new DirectQuery enabled model

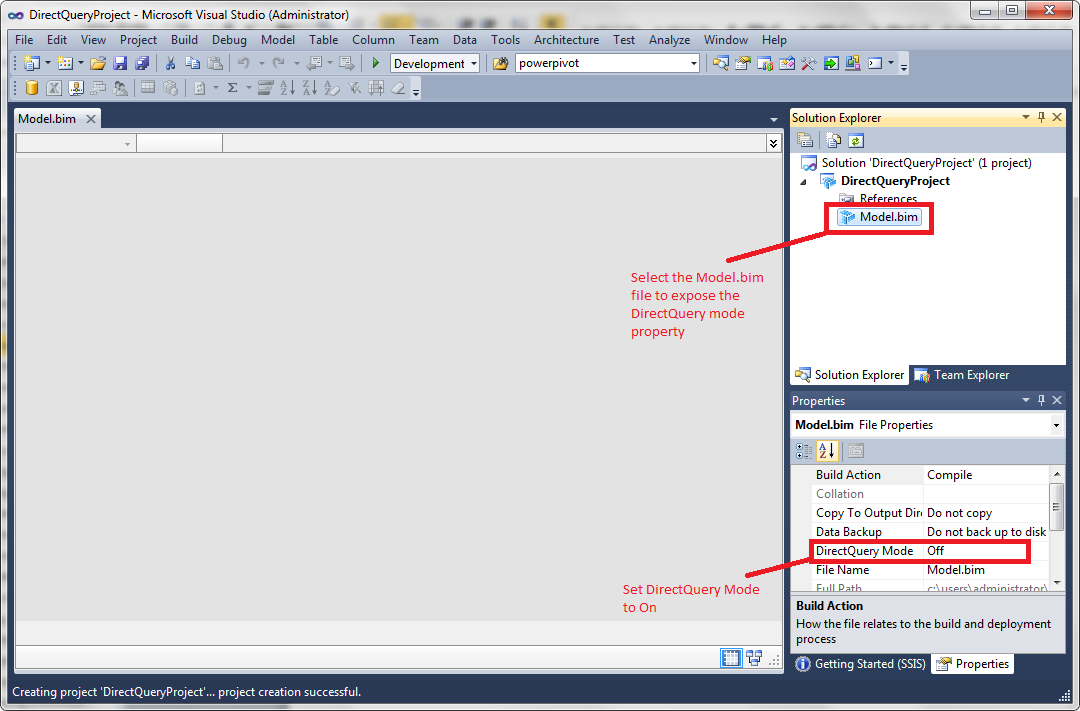
To create a new DirectQuery enabled model:

1. Launch SQL Server Data Tools.
2. On the **File** menu, point to **New**, and then click **Project**. Select **Analysis Services Tabular Model** from the Business Intelligence or Analysis Services category, set the name of the project to DirectQueryProject, and then click **OK**. An empty model appears, as shown in Figure 2.



**Figure 2:** An empty model in SQL Server Data Tools

1. Verify that both Solution Explorer and Properties window are visible. To show Solution Explorer, on the **View** menu, click **Solution Explorer**. To show the Properties window, on the **View** menu, click **Properties Window**.
2. From Solution Explorer, click to select the Model.bim file.
3. From the Properties window, change the value of the **DirectQuery Mode** property from **Off** to **On**. The following picture shows the location of the **DirectQuery Mode** property.



**Figure 3:** Changing the DirectQuery Mode property in the Properties window

After the DirectQuery mode is set to **On**, SQL Server Data Tools is in DirectQuery mode. The user interface changes as follows:

* The Import Wizard shows only SQL Server as a data source.
* The Role Manager disables row filters.
* You cannot add calculated columns.
* The **Paste**, **Paste Append**, and **Paste Replace** commands are disabled in the **Edit** menu.
* Error messages are provided when measures use unsupported DAX functions.
* The Partition Manager exposes the commands for configuring DirectQuery partitions.

For the purposes of the examples in this white paper, import some tables from the AdventureWorks database to start modeling in DirectQuery mode.

To import tables:

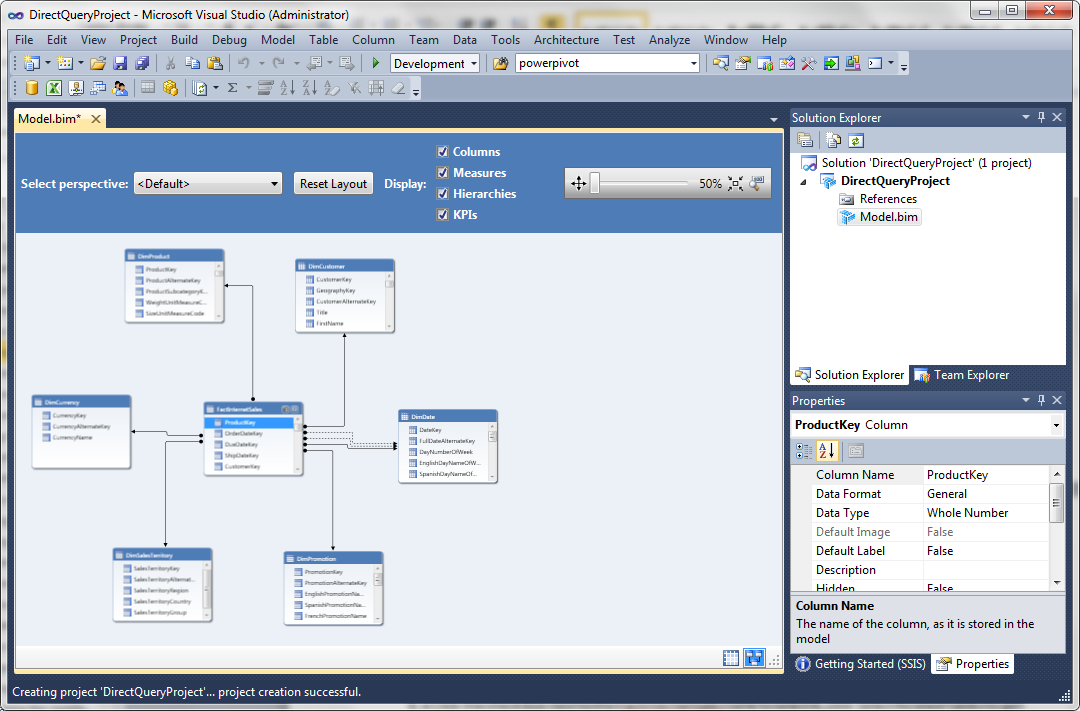
1. On the **Model** menu, click **Import from Data Source** to launch the Import Wizard.

or

On the Analysis Services toolbar, click the database icon.

1. Click **Next** to use Microsoft SQL Server as the data source.
2. Enter the server name and the database name for the AdventureWorksDW2012 data source and then click **Next**.
3. Enter the impersonation information for the DirectQuery data source. Enter the user name and password for a Windows user with at least read permission on the AdventureWorksDW2012 database and then click **Next**.
4. Click **Next** to import tables using preview mode.
5. Click the check box next to the FactInternetSales table to select it, click **Select Related Tables** to get some related dimensions, and then click **Finish**.

Seven tables and their relationships are imported. The following picture shows the model after the tables have been imported:



**Figure 4:** The diagram view for the model after the data was imported using the Import Wizard

Save this model before continuing.

## Converting an existing model to a DirectQuery enabled model

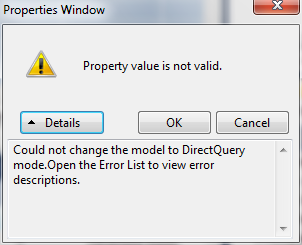
Any pre-existing model can be converted to a DirectQuery enabled model by changing the DirectQuery Mode property. If the model is compatible with DirectQuery, when you change the DirectQuery Mode property in the Properties window to **On**, SQL Server Data Tools switches to DirectQuery mode and you can edit the model. However, if the model is not compatible with DirectQuery, the conversion operation fails with an error.

The following example, which uses the AdventureWorks tabular project sample, illustrates the process of converting a model to a DirectQuery enabled model.

To change the AdventureWorks sample project to a DirectQuery enabled model:

1. Open the Adventure Works DW Tabular SQL Server 2012 project.
2. Verify that both Solution Explorer and Properties window are visible. To show Solution Explorer, on the **View** menu, click **Solution Explorer**. To show the Properties window, on the **View** menu, click **Properties Window**.
3. From Solution Explorer, click the Model.bim file.
4. From the Properties window, change the value of the **DirectQuery Mode** property from **Off** to **On**.

Because the Adventure Works sample was not designed as a DirectQuery enabled model, this operation fails with the following error message.



**Figure 5:** DirectQuery error message

All errors must be corrected before the model can be converted to DirectQuery mode.

Many errors that occur when converting to DirectQuery mode are shown in the Error List. To view the Error List, on the **View** menu, click **Error List**. Double-click any errors in calculated columns, measures, or key performance indicators (KPIs) in the Error List to navigate to the source of the error. You cannot double-click to navigate to errors in roles. After all errors in the Error List are corrected, you can repeat step 4 to try to convert the model to a DirectQuery enabled model again.

Some errors appear in a modal dialog box instead of the Error List. If the model uses data sources other than SQL Server or if the model uses functions in measures that are unsupported in DirectQuery mode, you are notified of the error when you try to change the model to DirectQuery mode. In these cases, correct the error and try to convert the model to a DirectQuery enabled model again.

**Tip** You can use the DAX Editor for SQL Server to view all DAX formulas for all measures in the model at the same time. This editor is available for download at <http://daxeditor.codeplex.com/>.

This exercise is time-consuming if you use the Adventure Works sample model. In this case, it is easier to create a new model in DirectQuery mode and then manually re-create the functionality in the Adventure Works sample using the data modeling techniques shown later in this paper. The examples in this paper take the approach of working from a new model.

## Choosing a DirectQuery mode

There are two DirectQuery modes: DirectQuery only mode or hybrid query mode. Models in DirectQuery only mode always answer queries from the relational data source. Models in a hybrid query mode can answer queries directly from the relational data source using DirectQuery or from data stored in the xVelocity engine cache. The DirectQueryMode connection string parameter specifies the source to use (relational data or xVelocity engine cache) when fetching query results.

There are two hybrid modes: DirectQuery with In-Memory and In-Memory with DirectQuery. In both of these modes, data is retained in the xVelocity engine cache. The only difference is the mode that is used to answer queries by default. In DirectQuery with In-Memory mode, queries are answered using DirectQuery by default. Queries are answered from the xVelocity engine cache by default when a model is running in In-Memory with DirectQuery mode.

When designing a DirectQuery enabled model, decide up front whether a cache will be retained. You can use this information when determining the supported reporting clients, planning capacity, and designing security.

You must always explicitly specify a DirectQuery mode before deploying your model. Deployment fails if you attempt to deploy a DirectQuery enabled model without specifying the query mode.

The following table summarizes the best practices for using each mode.

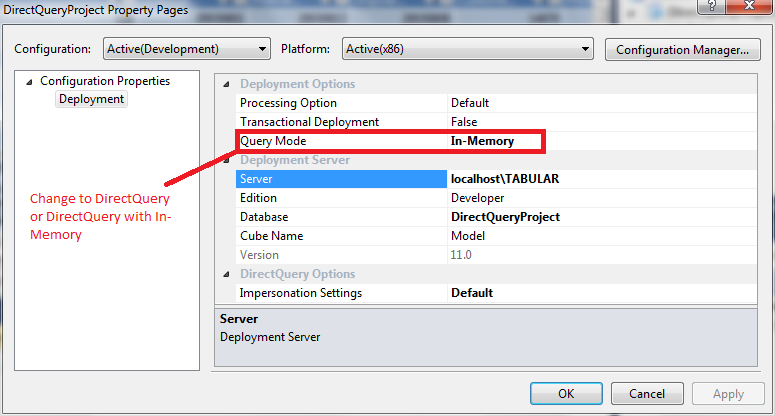
|  |  |
| --- | --- |
| Query mode | When to use |
| DirectQuery | Use when Power View is the only reporting client for a DirectQuery enabled model. |
| DirectQuery with In-Memory | Use when MDX-issuing clients, such as Excel, must connect to a model running in DirectQuery mode. Power View may connect in DirectQuery mode using a Business Intelligence Semantic Model (BISM) or a Report Server Data Source (RSDS) file.  Use when Power View is the default client for querying the model. |
| In-Memory with DirectQuery | Use when MDX-issuing clients, such as Excel, must connect to a model running in DirectQuery mode. Power View can only connect in DirectQuery mode by using an RSDS file if the DirectQuery mode parameter is embedded in the connection string.  Use when the default client for querying the model is an MDX-issuing client. Do not use when BISM files are used to connect to Power View. |

**Table 1:** Best practices for using each DirectQuery mode.

To select a query mode for the DirectQueryProject sample:

1. View Solution Explorer.
2. Right-click the DirectQueryProject project file and then click **Properties**.
3. Change the **Query Mode** property to **DirectQuery**, **DirectQuery with In-Memory**, or **In-Memory with DirectQuery** and then click **OK**.

The following picture shows the location of the Query Mode property.



**Figure 6:** The project properties for the tabular project with the Query Mode property highlighted

After you have set the query mode, you can deploy the model successfully. If you do not change the query mode, the build fails with the error “Could not deploy the model because DirectQuery mode is On but the Query Mode deployment property is set to an incompatible value. Change the Query Mode deployment property to one of the DirectQuery modes and then deploy the model.” To correct this build error, change the **DirectQuery Mode** property as described earlier and then rebuild the tabular model project.

## Partitioning a DirectQuery enabled model

Each table in a DirectQuery table has exactly one partition designated as the DirectQuery partition. When the model is queried in DirectQuery mode, the query that defines the DirectQuery partition for a table is used for building the query to the relational data source, and all other partition definition queries are disregarded. By default, the first partition added to the table serves as the DirectQuery partition. The DirectQuery partition may be an empty partition used exclusively for pass-through queries, or it may contain data that is used when answering queries from the xVelocity engine cache.

It can be useful to create additional partitions when the model is running in a hybrid mode. Partitions can be used to manage processing of data into the cache. Partitioning a DirectQuery only model is not recommended.

This section of the white paper shows some partitioning techniques for DirectQuery enabled models.

### Creating a DirectQuery only partition

By default, the DirectQuery partition is allowed to have data in the xVelocity engine cache. However, there are certain scenarios where it is not desirable to have data in the cache. In those cases, the partition should be converted to a DirectQuery only partition before deployment. When a partition is designated as a DirectQuery only partition, data is never loaded into the cache for the partition, even if the partition is included as part of a processing operation.

The following list describes the scenarios where a table should have a DirectQuery only partition defined:

* The model is in DirectQuery only mode
* The model is in a hybrid mode, and tables have two or more partitions defined to manage processing.

In these cases, the DirectQuery partition must be modified in the Partition Manager to change it to a DirectQuery only partition.

To change a partition to a DirectQuery partition in SQL Server Data Tools:

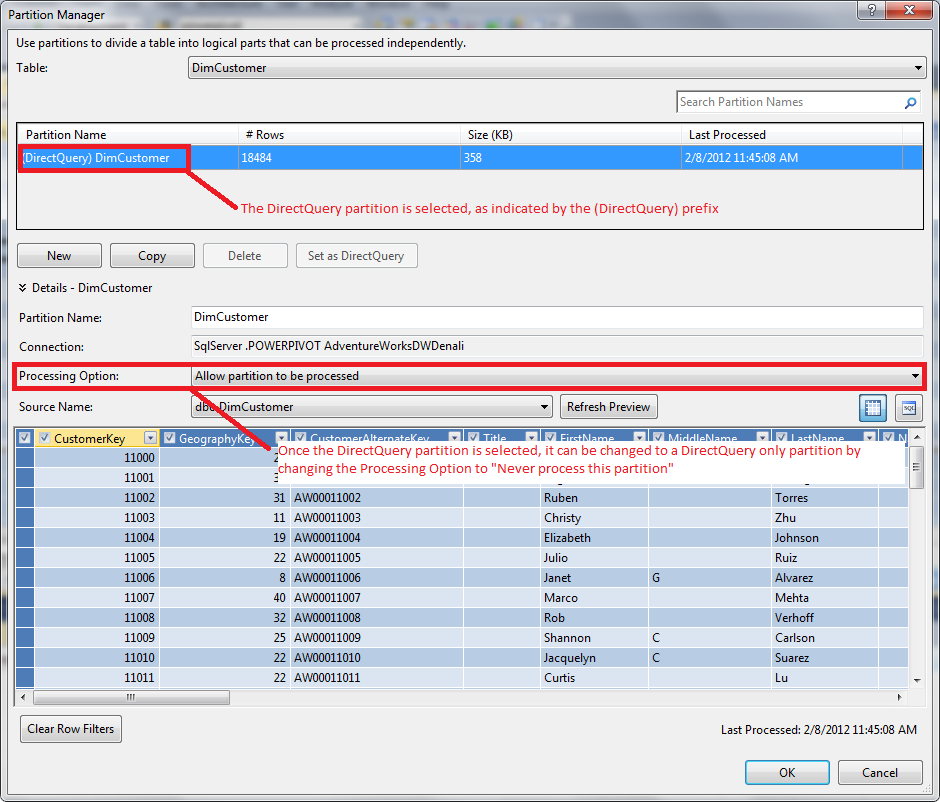
1. Select the table that contains the DirectQuery partition that you want to change.
2. On the **Table** menu, click **Partitions** to launch the Partition Manager.

or

On the Analysis Services toolbar, click the partition icon.

1. Select the DirectQuery partition. This partition is identified by the (DirectQuery) prefix in front of the partition name.
2. In the **Processing Option** box, select **Never process this partition**.
3. Click **OK**. The DirectQuery partition is now a DirectQuery only partition, and data will never be processed into its cache.

The following picture shows the location of the Processing Option property in the Partition Manager.



**Figure 7:** The Partition Manager in SQL Server Data Tools with the DirectQuery properties highlighted

When you change a partition to a DirectQuery only partition, all of the data is cleared from the workspace database and from the designer in SQL Server Data Tools. This illustrates one of the properties of DirectQuery only partitions – the transaction that commits the change to a DirectQuery only partition automatically drops all the data for that partition.

It may be helpful to postpone setting a partition to DirectQuery only until after the model is deployed, so that data can remain in the workspace database for testing purposes during development. Note that if you take this approach, any time you redeploy the model you must reapply the partition settings unless the partitions are preserved using the Deployment Wizard. For information about changing a partition to a DirectQuery only partition after deployment, see “Managing a DirectQuery model in SQL Server Management Studio.” However, it may be useful to set the partition to a DirectQuery only partition during development if security of the xVelocity engine cache is important. If there is no data in the cache, then there is no need to worry about securing the workspace database.

### Adding partitions to a DirectQuery model

Models running in hybrid mode may be partitioned like any other tabular model. Partitioning a table in a model makes it easier to load data for large tables. For more information about creating and using partitions, see [Partitions (SSAS-Tabular Model)](http://msdn.microsoft.com/en-us/library/hh230976(v=sql.110).aspx) (http://msdn.microsoft.com/en-us/library/hh230976(v=sql.110).aspx).

One partitioning pattern for a hybrid model is to create a single DirectQuery only partition that covers the entire data set, and multiple in-memory partitions that partition a table by date or by an attribute like geography. These smaller in-memory partitions can then be processed, merged, or deleted just as they would in any other tabular model. In this scenario, the partition definitions for the in-memory partitions may overlap with that of the DirectQuery partition, because the DirectQuery partition is never processed.

**Note** When you create partitions using this pattern, the DirectQuery partition definition must span the range of data across all of the in-memory partitions. Failure to define partitions in this way can cause query results to differ based on the query mode (DirectQuery or in-memory), which is not desirable.

Another partitioning pattern is to leave a single partition on a table that can be processed. This approach makes sense for smaller tables.

Avoid adding additional partitions to a table that has a DirectQuery partition that can be processed. If the added partitions overlap with those of the DirectQuery partition, processing will fail (if the table has a unique identifier) or duplicate data will be loaded into the model. If the added partitions do not overlap with the DirectQuery partitions, queries will return different results based on the query mode (DirectQuery or in-memory), which is not desirable.

It is not useful to create a small partition that returns real-time data and larger partitions with historical data with the intention that these results be combined in a single query. This scenario is not supported in SQL Server 2012. Models are queried either in DirectQuery mode or in in-memory mode, never in both in response to a single query.

Partitioning a model in SQL Server Data Tools is optional. Models can be partitioned later in SQL Server Management Studio or through automation. For an example of partitioning a model in code, see http://cathydumas.com/2011/12/27/creating-a-tabular-partition-in-code/.

Never add partitions to a DirectQuery only model. Although the user interface and the engine permit you to create and process these in-memory partitions, the partitions are never used to return query results. DirectQuery only models should have a single DirectQuery only partition to minimize resources used by Analysis Services.

### Changing the DirectQuery partition

Sometimes it is useful to change the DirectQuery partition from the only created automatically by SQL Server Data Tools to a different partition. For example, when you delete the DirectQuery partition in a model with two or more partitions, SQL Server Data Tools automatically changes the DirectQuery partition to the next partition defined in the table, which is not necessarily the partition you want to use as the DirectQuery partition. Also, you may simply decide that another partition that you have created is more appropriate to use as the DirectQuery partition.

**Note** Ensure that the DirectQuery partition definition spans the range of data in the table. If the DirectQuery partition does not include all data, query results may differ based on the query mode (DirectQuery or in-memory), which is not desirable.

To change the DirectQuery partition for a table in SQL Server Data Tools:

1. Select the table that contains the DirectQuery partition that you want to change.
2. On the **Table** menu, click **Partitions** to launch the Partition Manager.

or

On the Analysis Services toolbar, click the partition icon.

1. Select the partition that you want to use as the DirectQuery partition.
2. Click **Set as DirectQuery**. The (DirectQuery) prefix is added to the display name of the selected partition.

**Note** The **Set as DirectQuery** button is only enabled when there are two or more partitions in the table.

1. [optional] Set the new partition as a DirectQuery only partition, as described in the section “Creating a DirectQuery only partition.”
2. Click **OK**. The DirectQuery only partition is changed.

If the previous DirectQuery partition was a DirectQuery only partition, the processing option for the partition is automatically changed so that data can be loaded into the xVelocity engine cache using that partition definition query.

## Setting the DirectQuery impersonation settings

Impersonation in Analysis Services refers to the credentials used to connect to the data source. For more information about impersonation in tabular models, see [Impersonation (SSAS – Tabular Models)](http://msdn.microsoft.com/en-us/library/gg492180(v=sql.110).aspx) (http://msdn.microsoft.com/en-us/library/gg492180(v=sql.110).aspx).

Modelers specify the credentials used to connect to the data source for processing in the Import Wizard or in the **Existing Connections** dialog box. Either a specific Windows user’s credentials or the Analysis Services service account can be used. For in-memory models, these credentials are the only credentials used after the model is deployed because Analysis Services only connects to the data source when processing. DirectQuery enabled models, by contrast, connect to the data source in two scenarios – when querying the model and when processing the model (for models in hybrid mode). Because there is one additional scenario for connections, there is one additional set of impersonation settings, called the DirectQuery impersonation setting.

The DirectQuery impersonation setting specifies the credentials to use when querying the data source in DirectQuery mode. There are two possible values for this setting: Default and ImpersonateCurrentUser.

The following table describes the DirectQuery impersonation settings.

|  |  |
| --- | --- |
| Setting | Meaning |
| Default | Always use the credentials specified in the Import Wizard to connect to the data source, both when querying and processing the model. |
| ImpersonateCurrentUser | When querying the model from a client application using DirectQuery mode, use the current user’s credentials to connect to the relational data source.  When processing, use the credentials specified in the Import Wizard to connect to the relational data source. |

**Table 2:** DirectQuery impersonation settings.

The following list shows some security considerations to remember when using default impersonation settings:

* Power View users should not be granted access to the underlying SQL Server data source.
* Users can read all data in the model, and additional end-user security cannot be defined.

The following list shows some security considerations to remember when using the ImpersonateCurrentUser setting:

* Power View users must be granted read access on the underlying SQL Server data source.
* More granular row security can be added to the SQL Server data source, restricting the query results on a per-user basis when querying in DirectQuery mode.
* Users querying the xVelocity engine cache can view all data in the cache, regardless of their privileges on the SQL Server data source.
* Analysis Services must delegate the user’s credentials to SQL Server. If Analysis Services and SQL Server reside on two separate machines, Kerberos must be configured to enable delegation. If Analysis Services and SQL Server reside on the same machine and Kerberos is not configured, the user running the Analysis Services service account must have sufficient permissions in Windows to delegate credentials. For more information, see “Enabling constrained delegation.”

The security model is simplified when the model is using default impersonation settings. Users are granted access only to the Analysis Services model, so it is not necessary to manage permissions twice (once on Analysis Services and once on SQL Server). Also, when using hybrid mode, it is guaranteed that per-user security is never enforced. Instead, read access to the model is managed using roles on the Analysis Services database.

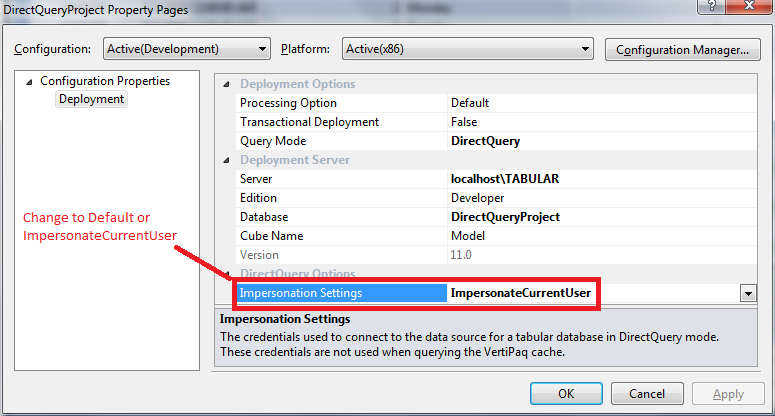
The security model is different when the model is using the ImpersonateCurrentUser setting. Users must be granted access to both Analysis Services and SQL Server. SQL Server security applies if queries are answered via DirectQuery, and Analysis Services security applies if queries are answered from the xVelocity cache.

Consider using ImpersonateCurrentUser for DirectQuery only models when you need to use row security on a DirectQuery model. Avoid using ImpersonateCurrentUser for hybrid models, as it can be confusing to manage two different security contexts for the same underlying model.

To change the DirectQuery impersonation information:

1. View Solution Explorer.
2. Right-click the DirectQueryProject project file and then click **Properties**.
3. Change the value of the **Impersonation Settings** property to the value you want (Default or ImpersonateCurrentUser).

The following picture shows the location of the **Impersonation Settings** property.



**Figure 8:** The project properties for the tabular project with the **Impersonation Settings** property highlighted

# Data warehousing and data modeling techniques for DirectQuery

DirectQuery models have some restrictions on supported functions and some different performance and data loading characteristics. It is important to design your data warehouse and your data model with these characteristics and limitations in mind. This section of the white paper describes how to optimize your data warehouse and your data model for DirectQuery.

## Using a columnstore index

Using a columnstore index improves the performance of a DirectQuery model. Columnstore indexes are a new feature in SQL Server 2012. These indexes use similar xVelocity in-memory technologies as Analysis Services. For basic information about the columnstore index, see [Columnstore Indexes](http://msdn.microsoft.com/en-us/library/gg492088(v=SQL.110).aspx) (http://msdn.microsoft.com/en-us/library/gg492088(v=SQL.110).aspx). For an in-depth introduction to the columnstore, see <http://channel9.msdn.com/Events/TechEd/NorthAmerica/2011/DBI312>.

If you are using a SQL Server 2012 data source, consider using a columnstore index to improve performance. However, not all scenarios are suited for using the columnstore index. Tables with a columnstore index are read-only. If you are using DirectQuery to enable real-time updates to data sources, using a columnstore index may not be appropriate.

## Loading data efficiently during ETL

Using DirectQuery does not eliminate the need for ETL. Updates to the source table still need to be managed.

SQL Server locks databases both while queries are answered and when data is inserted. If the necessary locks for read or insert cannot be acquired, performance suffers.

Consider batching updates to the SQL Server data source to reduce the amount of time that the data source is locked. Alternatively, consider creating a snapshot on the SQL Server database and have the DirectQuery enabled model query the database snapshot. This ensures consistent data is returned, even while data is being loaded and columnstore indexes are being rebuilt.

If you are using columnstore indexes, data should be loaded into fact tables in accordance with the best practices described at <http://msdn.microsoft.com/en-us/library/gg492088(v=SQL.110).aspx#Update>. If staging tables are used for loading data, additional memory and CPU resources are required on the machine hosting the SQL Server data source. These requirements should be considered when the hardware for DirectQuery is sized.

## Performing calculations at the column level

DirectQuery does not support the use of calculated columns in SQL Server 2012, although measures are supported. If you need to perform calculations at the column level, you must change your data warehousing or data modeling approach to calculate the required values.

There are many ways to perform calculations in the data warehouse itself. You can create a view that defines the required calculated columns. The values for the columns can be computed in the view definition, via a trigger when the data is inserted, or as part of the ETL process.

Other approaches include creating a computed column in your SQL Server fact table, optionally persisting the column. These computed columns can directly refer to columns in the same table or could indirectly refer to columns in related tables through user-defined functions. Alternatively, you could write a stored procedure that generates the table that you import into the data model.

Also, calculations can be performed outside of the data warehouse through changes to the source definition for the table, so that the value is computed when the relational data source is queried. For example, say you want to include a Product Subcategory column in the DimProduct table. The AdventureWorks tabular model example has a calculated column named “Product Subcategory” on the Product table, which uses the RELATED() function in DAX to fetch the related subcategory from the Subcategory table. In DirectQuery mode, you would modify the source table definition to fetch the related column value using an SQL JOIN statement.

To modify the source definition of a table to compute a column in SQL Server Data Tools:

1. Select the DimProduct table.
2. Press F4 to show the property grid.
3. Click the **Source Data** property to edit it.
4. Using the drop-down box in the top right corner of the dialog box, switch to Query Editor.
5. Paste the following query:

SELECT

DimProduct.ProductKey

,DimProduct.ProductAlternateKey

,DimProduct.ProductSubcategoryKey

,DimProduct.WeightUnitMeasureCode

,DimProduct.SizeUnitMeasureCode

,DimProduct.EnglishProductName

,DimProduct.StandardCost

,DimProduct.FinishedGoodsFlag

,DimProduct.Color

,DimProduct.SafetyStockLevel

,DimProduct.ReorderPoint

,DimProduct.ListPrice

,DimProduct.[Size]

,DimProduct.SizeRange

,DimProduct.Weight

,DimProduct.DaysToManufacture

,DimProduct.ProductLine

,DimProduct.DealerPrice

,DimProduct.[Class]

,DimProduct.Style

,DimProduct.ModelName

,DimProduct.LargePhoto

,DimProduct.EnglishDescription

,DimProduct.StartDate

,DimProduct.EndDate

,DimProduct.Status

,DimProductSubcategory.EnglishProductSubcategoryName

FROM

DimProductSubcategory

INNER JOIN DimProduct

ON DimProductSubcategory.ProductSubcategoryKey = DimProduct.ProductSubcategoryKey

1. Click **OK**. The new column is generated.

Although this paper demonstrates the use of columns that are computed outside of the data warehouse, doing the calculations inside of the data warehouse may increase performance.

## Designing date and time calculations for DirectQuery

Most time intelligence functions are not supported in DirectQuery mode. For more information, including a complete list of unsupported functions, see [Formula Compatibility in DirectQuery Mode](http://msdn.microsoft.com/en-us/library/hh213006(v=sql.110).aspx) (http://msdn.microsoft.com/en-us/library/hh213006(v=sql.110).aspx). To overcome this limitation, change your approach for time intelligence in DirectQuery. There are two major techniques to use– rewriting time intelligence functions for period to date calculations using basic DAX, and widening the date table to add earlier and later periods to the row context.

### Performing period to date calculations using basic DAX

Period to date calculations, such as TOTALYTD(), TOTALMTD(), and TOTALQTD() must be computed manually in DAX using the basic DAX functions. For example, in the AdventureWorks tabular model sample, there is a measure that calculates the rolling sum of the margin for the current quarter, as shown here.

Internet Current Quarter Margin:=

TOTALQTD([Internet Total Margin],'Date'[Date])

The following DAX measure is functionally equivalent to the previous measure, but rewritten to use basic DAX.

Internet Current Quarter Margin:=

IF(

HASONEVALUE(DimDate[CalendarQuarter]) && HASONEVALUE(DimDate[CalendarYear]),

CALCULATE(

[Internet Total Margin],

FILTER(

ALLEXCEPT( DimDate, DimDate[CalendarYear], DimDate[CalendarQuarter]),

DimDate[DateKey] <= MAX( DimDate[DateKey])

)

)

)

The calculation does the following:

* The outer IF() does basic error checking. The rolling sum calculation is performed only if there is one value for Calendar Year and one value for Calendar Quarter, as indicated by the HASONEVALUE() functions. Otherwise, this measure returns a blank.
* The inner CALCULATE() calculates the margin for the quarter to date. To do this, the filter context must be set on the DimDate table so that all dates in the current quarter, up to and including the selected date, are included in the calculation. The FILTER() expression sets this context, by first removing all filters on the DimDate table except those on the calendar year and quarter using the ALLEXCEPT() function, then adding back a filter on the date key, selecting all dates that are less than or equal to the currently selected date (DimDate[DateKey]<=MAX(DimDate[DateKey]).

### Widening date tables to include previous and next periods

You can replace DAX time intelligence functions that find the previous and next periods, such as PREVIOUSYEAR(), PREVIOUSQUARTER(), NEXTYEAR(), and NEXTQUARTER() by widening the date table and adding these calculations as columns to the date table. These new columns can then be used to alter the filter context so time intelligence calculations can be performed using basic DAX. The DATEADD() function in DAX can be replaced using the same technique, using the built-in Transact-SQL DATEADD function when previous or next periods are added in calculations.

Here is an example. The AdventureWorks tabular model example defines a measure that calculates the total margin on Internet sales in the previous quarter, like this.

Internet Margin Previous Quarter:=

CALCULATE(

[Internet Total Margin],

PREVIOUSQUARTER(

'Date'[Date])

)

You can replicate this functionality using a widened date table as follows. First, using the data modeling technique illustrated in the preceding example, change the source definition for the DimDate table to add two new columns, PreviousCalendarQuarterNumber and PreviousCalendarQuarterYear.

The new table definition is as follows.

SELECT

DimDate.DateKey

,DimDate.FullDateAlternateKey

,DimDate.DayNumberOfWeek

,DimDate.EnglishDayNameOfWeek

,DimDate.DayNumberOfMonth

,DimDate.DayNumberOfYear

,DimDate.WeekNumberOfYear

,DimDate.EnglishMonthName

,DimDate.MonthNumberOfYear

,DimDate.CalendarQuarter

,DimDate.CalendarYear

,DimDate.CalendarSemester

,DimDate.FiscalQuarter

,DimDate.FiscalYear

,DimDate.FiscalSemester

,PreviousCalendarQuarterNumber=IIF(DimDate.CalendarQuarter=1, 4, DimDate.CalendarQuarter-1)

,PreviousCalendarQuarterYear=IIF(DimDate.CalendarQuarter=1, DimDate.CalendarYear-1, DimDate.CalendarYear)

FROM

DimDate

Now you can use these columns to get information about the previous quarter. The following DAX measure calculates the margin for the previous quarter.

Internet Margin Previous Quarter:=

IF(

HASONEVALUE(DimDate[CalendarQuarter]) && HASONEVALUE(DimDate[CalendarYear]),

CALCULATE(

[Internet Total Margin],

All(DimDate),

DimDate[CalendarYear] = VALUES(DimDate[PreviousCalendarQuarterYear]),

DimDate[CalendarQuarter] = VALUES(DimDate[PreviousCalendarQuarterNumber])

)

)

The calculation does the following:

* The outer IF() does basic error checking. The rolling sum calculation is performed only if there is one value for Calendar Year and one value for Calendar Quarter, as indicated by the HASONEVALUE() functions. Otherwise, this measure returns a blank.
* The CALCULATE() function calculates the total margin for sales, with three filters applied.
* The ALL() function removes all filters from the DimDate table for the calculation.
* The next Boolean expression restricts the CalendarYear used for the calculation. This filter specifies that the year associated with the previous calendar quarter, and not the current year, should be used for the calculation.
* The final Boolean expression restricts the CalendarQuarter used for the calculation, specifying that the previous quarter number should be used instead of the current quarter number.

## Securing the data warehouse

One of the benefits of using DirectQuery is that you can use the SQL Server security model to secure the data source. A discussion of this security model is outside the scope of this document. After you define the roles for the SQL Server data source, you must add some Analysis Services users to these roles.

The SQL Server data warehouse must be configured to grant read access to two types of Analysis Services users:

* The user account (either a named Windows user or the Analysis Services service account) that is used to process data
* End users, either directly or indirectly, that query the data using DirectQuery

Hybrid models require you to specify a user who has read access to the relational data source for loading data. You do this in the Import Wizard.

End users must be directly granted access to the relational data source when the DirectQuery impersonation settings are set to impersonate the current user. However, when they use the default impersonation settings, end users should not be granted access to the data source for querying purposes. Granting users read permission to the data source is unnecessary, since their credentials are never used when querying via DirectQuery. Instead, the credentials for the user that can process the model are used to query the model.

Minimize the number of users with read access to the relational data source. If you are using default impersonation settings, only the user specified in the Import Wizard needs access to SQL Server. All other users are granted read access to the DirectQuery enabled model using roles on the Analysis Services database, not by granting access to SQL Server.

# Using SQL Server Data Tools effectively in DirectQuery mode

When a model is converted to a DirectQuery enabled model in SQL Server Data Tools, the query mode for the workspace database is set to the hybrid In-Memory with DirectQuery mode. Therefore, you must manage the xVelocity engine cache for the workspace database, even if you plan to deploy your model as a DirectQuery only model when you deploy to production. This section of the white paper describes some techniques for managing the cache in the workspace database and in SQL Server Data Tools.

## Loading data efficiently into SQL Server Data Tools

By default, when a table is imported, data is loaded into the xVelocity engine cache and queries in the designer are answered from this cache. There is no way to force the designer to create a DirectQuery only partition at import time. Therefore, if you use the Import Wizard, you must plan to use only a subset of the production data when you are building your model to avoid importing excessive amounts of data into the cache.

The recommended approach is to create a custom database that contains a subset of your data for use in the development environment. When you deploy to production, change the connection strings to point to the production data set. These connection strings can be modified in the Deployment Wizard or in SQL Server Management Studio.

Another approach is to define a view on your data source, restrict the number of rows returned from the view, and then import from the data source in SQL Server Data Tools. This row restriction can then be removed from the view in production without the need for a change to the model metadata.

## Adding more tables to a DirectQuery enabled model

After you add tables to a DirectQuery enabled model, you cannot launch the Import Wizard from the **Model** menu or from the Analysis Services toolbar to add more tables to the model. If you try to do this, import fails with the error “DirectQuery error: All tables used when querying in DirectQuery Mode must be from a single relational Data Source.” Instead, use the **Existing Connections** dialog box to add more tables.

To add more tables to a model using the **Existing Connections** dialog box, follow these steps:

1. On the **Model** menu, click **Existing Connections**.

or

On the Analysis Services toolbar, click the connection icon.

The **Existing Connections** dialog box appears with the connection to the SQL Server data source for the model automatically selected.

1. Click **Open**. This launches the Import Wizard.
2. Follow the instructions in the Import Wizard to add more tables and then click **Finish**. The new tables are added to the model.

## Managing the xVelocity engine cache in SQL Server Data Tools

By default, any workspace database with tables contains data. Any administrator on the Analysis Services instance has access to the data in this database. Anybody with access to the file location where the data is stored on disk can attach the database to another Analysis Services instance and read the data in the cache. Also, any member of a read role can read data in the workspace database. The workspace database needs to be secured against unintended data access.

One way to avoid unintended data access from the workspace database is to construct the model SQL Server Data Tools based on the schema of an empty SQL Server database.

Another way to prevent unintended access is do configure the model so that any imported data is automatically cleared. You can do this by configuring all partitions in the model to be DirectQuery only partitions. For more information, see “Creating a DirectQuery only partition.”

You can also manually remove data from the workspace database by issuing a Process Clear command in SQL Server Data Tools.

To remove data from partitions using Process Clear:

1. On the **Model** menu, point to **Process**, and then click **Process Partitions**.

or

On the Analysis Services toolbar, click the **Process** menu and then click **Process Partitions**.

1. Select the check boxes of the partitions that you want to clear.
2. Change the setting for **Mode** to **Process Clear + Process Recalc** and then click **OK**.

Repeat this procedure for all tables that you want to clear.

Also, you can postpone the addition of members to read roles until after deployment to avoid read access to the cache. Note that if you redeploy the model from SQL Server Data Tools, the roles (and therefore the set of members added to the roles) are overwritten. If you do not want to overwrite the list of role members, do not redeploy from SQL Server Data Tools. Instead, use the Deployment Wizard to redeploy the model and select **Deploy roles and retain members** or **Retain roles and members** to preserve the list of role members. For more information about the Deployment Wizard, see [Deploy Model Solutions Using the Deployment Wizard](http://msdn.microsoft.com/en-us/library/ms176121(v=sql.110).aspx) (http://msdn.microsoft.com/en-us/library/ms176121(v=sql.110).aspx).

Finally, you can configure SQL Server Data Tools to automatically delete the workspace database when the tabular project is closed. This option ensures that it is impossible to view data after the designer is closed. However, if you elect to do this, opening the project is slower.

To configure SQL Server Data Tools to delete the workspace database when you close the DirectQuery project:

1. Verify that both Solution Explorer and Properties window are visible. To show Solution Explorer, on the **View** menu, click **Solution Explorer**. To show the Properties window, on the **View** menu, click **Properties Window**.
2. From Solution Explorer, click the Model.bim file.
3. From the Properties window, change the value of the **Workspace Retention** property to **Delete Workspace**.

## Deleting the workspace database

The workspace database can also be manually deleted after the tabular project is closed. The workspace database is located in the data directory of the Analysis Services instance hosting the workspace database. This location varies by server configuration.

To find the location of the data directory:

1. Start SQL Server Management Studio and connect to the Analysis Services instance that hosts the workspace database.
2. In the Object Explorer, right click the Analysis Services instance and then click **Properties**. The **DataDir** property specifies the location on disk of the workspace database.

For an Analysis Services instance named TABULAR running on a 64-bit operating system, the default location for the workspace database is C:\Program Files\Microsoft SQL Server\MSAS11.TABULAR\OLAP\Data.

# Connecting to a DirectQuery enabled model

Some reporting clients can connect directly to a DirectQuery enabled model using just a connection string. Others, such as Power View, must have an intermediate data source like a BISM or RSDS file defined before the connection can occur.

The right method for connecting to the data source depends on the answers to a number of questions about the deployment environment, such as:

* Is Power View the only client connecting to the model, or are other clients also connecting to the model in hybrid mode?
* Are all users that connect to the model Windows users, or are there any users that do not have Windows credentials that must connect to reports based on the model?
* Are connections to the model crossing machine, Microsoft SharePoint farm, or domain boundaries?
* Is user-level security applied on the SQL Server data source?

The answers to these questions will determine the approach for connecting to a DirectQuery enabled model. The following sections describe three connection methods – connecting using an MDX client that connects via a connection string, connecting using a BISM file, and connecting using a RSDS file.

## Connecting directly to a DirectQuery enabled model using an MDX client

Any MDX client can be configured to connect to a DirectQuery enabled model running in hybrid mode. Connections can be made directly from the client applications, or connections can be saved in an ODC file for reuse.

If the model is running in In-Memory with DirectQuery mode, connection attempts from authenticated users work if they use the default connection string, and query results are returned from the xVelocity engine cache.

If the model is running in DirectQuery with In-Memory mode, you must specify that you want to connect using In-Memory mode by setting the DirectQueryMode connection string parameter to the value InMemory. If this connection string parameter is not specified, any connection attempt fails with the error “DirectQuery error: MDX/SQL operations are not supported in DirectQuery mode.”

To connect to a DirectQuery model using In-Memory mode from Excel:

1. On the **Data** tab, in the **Get External Data** group, click **From Other Sources** and then click **Analysis Services**.
2. Type the name of the Analysis Services instance that hosts the DirectQuery model and then click **Next**.
3. Select the DirectQuery database and then click **Finish**. The **Import Data** box appears.
4. In the **Import Data** box, click **Properties**.
5. Click the **Definition** tab.
6. In the **Connection String** box, append the text ;DirectQueryMode=InMemory to the pre-existing text.

If the model is running in DirectQuery only mode, any connection attempt from an MDX client fails with the error “DirectQuery error: The database is in DirectQuery mode, therefore, cannot support InMemory mode queries.”

## Connecting to a DirectQuery enabled model using a BISM file

The BI Semantic Model connection file is a new content type for SharePoint that enables you to connect to Analysis Services databases or PowerPivot for SharePoint models from reporting clients. This content type is available when PowerPivot for SharePoint is configured on the farm. For more information about BISM files, see [PowerPivot BI Semantic Model Connnection (.bism)](http://msdn.microsoft.com/en-us/library/gg471575(v=sql.110).aspx) (http://msdn.microsoft.com/en-us/library/gg471575(v=sql.110).aspx).

When using a BISM file to connect to a DirectQuery model, ensure that the model is running in DirectQuery or DirectQuery with In-Memory mode. This is because there is no way to override the default query mode for the database either in Power View or in the BISM file.

You should only use Power View to connect to DirectQuery models using BISM files. Although Excel and other MDX issuing clients can normally use a BISM file to connect to a tabular model, they cannot use the BISM file to connect to a DirectQuery model running in a hybrid mode because you cannot edit the connection string properties on a BISM file to switch between DirectQuery and In-Memory mode.

If your Analysis Services and SQL Server instances do not reside in the SharePoint farm that hosts Power View, some additional configuration is required before you can use a BISM file to connect to your DirectQuery model. You must either configure Kerberos or add the account that runs the SQL Server Reporting Services application in SharePoint to the list of administrators on the Analysis Services instance that hosts the DirectQuery model. Either of these configurations enables the report user’s credentials to flow through to Analysis Services and SQL Server. A complete discussion of these configuration options is out of the scope of this document. For more information about configuring Kerberos and configuring Power View and BISM files to enable DirectQuery connectivity when Analysis Services is outside of the SharePoint farm, see “Additional resources.”

In general, using the BISM file is the preferred approach for connecting from Power View to a tabular model. However, there are some scenarios in which using a RSDS file to connect to the tabular model is preferable. The next section discusses these scenarios.

## Connecting to a DirectQuery enabled model using a RSDS file

Reporting Services uses RSDS files to connect to Analysis Services models.

Use a RSDS file when:

* The model is running in In-Memory with DirectQuery mode, and a DirectQuery-specific RSDS file is used to connect to the tabular model from Power View. Connection strings in the RSDS file are user configurable.
* Users must connect to reports based on the tabular model using credentials other than Windows credentials.
* PowerPivot for SharePoint is not configured on the SharePoint farm, so the BISM content type is not available.
* Analysis Services resides outside of the SharePoint farm, and configuring Kerberos or elevating the privileges of the account running the Reporting Services application are not acceptable solutions.

Additional configuration is required when you use RSDS files to connect to Analysis Services. You must either set stored credentials for the RSDS file or configure the RSDS file to impersonate a specified user when connecting to Analysis Services. Also, if you are using stored credentials, note that user access to the model must be managed on the RSDS file, not on Analysis Services, which adds complexity to the security model. A complete discussion of these configuration options is out of the scope of this document. For more information, see <http://www.sqlpass.org/summit/2011/Speakers/CallForSpeakers/SessionDetail.aspx?sid=2027>.

Note that Power View never prompts for credentials when connecting to a data source. This means that you cannot configure dynamic or row security using SQL authentication for users who request access to reports using credentials other than Windows credentials, because the SQL authentication credentials must always be stored in the Analysis Services model itself.

# Preparing the DirectQuery deployment environment

When building a deployment environment for DirectQuery, you must plan the deployment of Power View, Analysis Services, and SQL Server. Factors to consider are machine topology and machine capacity. These factors influence other configuration requirements for your application, such as the implementation of Kerberos.

## Machine topology

The following diagram shows a typical topology for a DirectQuery workload, where all users connecting to the model are Windows users. This topology includes Microsoft Silverlight 5, SharePoint, Power View, Analysis Services, and SQL Server.



**Figure 9:** The machine topology for a typical DirectQuery deployment

The SharePoint farm typically hosts multiple machines, including the web front end for handling incoming requests and one or more application servers. At least one application server will host the Power View application. If you are using a BISM file to connect to Analysis Services, you must also configure PowerPivot for SharePoint in the farm. PowerPivot can be configured on the same server as the Power View application, or it can be configured on a different server if you anticipate a large PowerPivot workload.

This SharePoint farm must be running SharePoint 2010 SP1, Enterprise edition. The application server hosting Power View and the application server for PowerPivot must have SQL Server 2012 Enterprise or BI edition licenses.

Always configure the SharePoint farm using the configuration tools provided with the SQL Server installer. This ensures all required permissions for connectivity are configured correctly. Additional configuration to pass credentials between machines within the farm is not required.

Typically, the Analysis Services and SQL Server instances will reside outside of the SharePoint farm that hosts Power View. Care must be taken to configure the environment so credentials pass from the farm to Analysis Services. For more information, see “Connecting to a DirectQuery model using a BISM file” and “Connecting to a DirectQuery model using a RSDS file.”

We recommend that both the Analysis Services instance and the SQL Server instance be hosted on the same machine for a DirectQuery only workload. Because there is not much CPU contention between Analysis Services and SQL Server, this configuration should perform well out of the box. Hosting Analysis Services and SQL Server on separate machines introduces the requirement to configure Kerberos for authentication and also introduces the risk of performance degradation due to network latency.

## Capacity planning for Analysis Services

This section discusses capacity planning for Analysis Services in DirectQuery mode. SQL Server and Reporting Services have their own capacity requirements. Guidance for sizing these installations is outside of the scope of this document.

In DirectQuery scenarios, Analysis Services is not CPU intensive. The computational work is performed by the SQL Server engine, not by the xVelocity engine, so Analysis Services does not have high CPU requirements. Analysis Services running in tabular mode works optimally with up to 4 sockets with 8 cores per socket, which should be sufficient to handle most DirectQuery workloads. Always select processors with larger L2 caches, which are optimal for Analysis Services.

Disk speed is not an important consideration when capacity planning for tabular models, because any data for the tabular model is retained in memory.

The memory requirements for Analysis Services in DirectQuery mode are much lower than for in-memory Analysis Services databases. Typical DAX queries on a DirectQuery model fetch a small fraction of rows in the database, leading to low memory usage for simple calculations. However, the memory required for Analysis Services depends on the DAX queries that are issued. Queries that fetch a large number of results (for example, queries that generate a large number of intermediate results for computation or table queries that return a large number of rows as output) place higher memory requirements for Analysis Services.

# Managing a DirectQuery enabled model in SQL Server Management Studio

Many administration tasks for DirectQuery enabled models can be performed in SQL Server Management Studio. You can attach, detach, back up, restore, synchronize, rename, or delete a DirectQuery enabled model just as you can for any other tabular model. Also, you can create roles, delete roles, manage role membership, and alter connection string information just as for other tabular models. DirectQuery enabled models in hybrid mode can be processed just as any other tabular model, and you can add partitions to the hybrid model in the same way you partition other tabular models.

**Note** Although it is possible to restore a PowerPivot model and change it to a DirectQuery enabled model in SQL Server Management Studio, this approach is not recommended. If your model requires changes so that it is compatible with DirectQuery, those changes must be made in SQL Server Data Tools and the model redeployed. The preferred approach is to create a new project in SQL Server Data Tools using the Import from PowerPivot template, convert the model to a DirectQuery enabled model, and then redeploy the model.

Some management operations in SQL Server Management Studio are different for DirectQuery enabled models. You can change a model from a DirectQuery only model to a hybrid model and vice versa. The Partition Manager exposes additional commands for changing the DirectQuery partition and its properties. Browsing a DirectQuery enabled model is different because the model does not accept MDX queries. Also, you can modify the DirectQuery impersonation settings and clear the xVelocity engine cache from SQL Server Management Studio.

## Changing the DirectQuery mode

You can change a model from a DirectQuery only to a hybrid model in SQL Server Management Studio and vice versa. Also, you can change the default query mode for the model from In-Memory to DirectQuery.

**Note** Although it is possible to change a model from In-Memory to DirectQuery mode in SQL Server Management Studio, this approach is not recommended. If your model requires changes so that it is compatible with DirectQuery, those changes must be made in SQL Server Data Tools and the model redeployed.

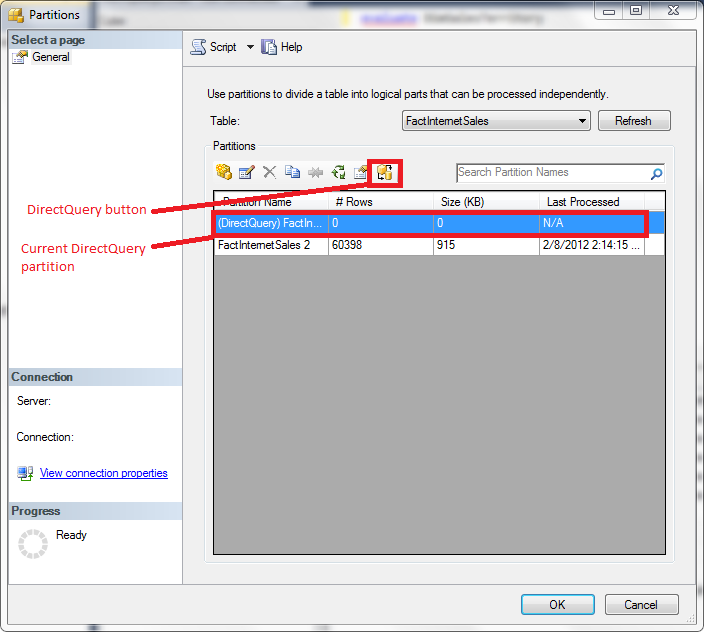
To change the DirectQuery mode:

1. Launch SQL Server Management Studio and connect to the Analysis Services instance that hosts your model.
2. Select the DirectQuery database in the Object Explorer.
3. Right-click the database and then click **Properties**.
4. Change the **DirectQueryMode** property to **DirectQuery**, **DirectQueryWithInMemory**, or **InMemoryWithDirectQuery**.

## Changing the DirectQuery partition in SQL Server Management Studio

The Partition Manager in SQL Server Management Studio is context sensitive, similar to the Partition Manager in SQL Server Data Tools. When a model is running in DirectQuery mode, a DirectQuery button is added to the toolbar in the Partition Manager. This button launches the **Set DirectQuery Partition** dialog box, which exposes the configuration settings for the DirectQuery partition.

The following picture shows the Partition Manager when the model is running in DirectQuery mode.



**Figure 10:** The Partition Manager in SQL Server Management Studio with the DirectQuery UI elements highlighted

You can change the DirectQuery partition in SQL Server Management Studio. For more information about the benefits of changing the DirectQuery partition, see “Changing the DirectQuery partition.”

To change the DirectQuery partition for a hybrid model in SQL Server Management Studio:

1. Launch SQL Server Management Studio and connect to the Analysis Services instance that hosts your model.
2. In the Object Explorer, click to expand the Analysis Services instance, the Databases folder, the DirectQuery database, and the Tables folder.
3. Right-click the table that you want to modify and then click **Partitions**.
4. Click the partition that you want to use as the DirectQuery partition. If this partition does not exist, create it.
5. On the toolbar, click the DirectQuery icon.
6. Click **OK** to set the selected partition to the DirectQuery partition.

If you did not select the correct partition in step 4, you can change the DirectQuery partition by changing the **Partition Name** property in the **Set DirectQuery Partition** box.

## Changing the DirectQuery partition to a DirectQuery only partition

You can change the processing options for a DirectQuery partition in SQL Server Management Studio. You can configure the DirectQuery partition to be a DirectQuery only partition, which never contains data, or to be an InMemoryWithDirectQuery partition, which allows data to be processed into a partition. For more information about the benefits of using a DirectQuery only partition, see “Creating a DirectQuery only partition.”

To change the DirectQuery partition to be a DirectQuery only partition:

1. Launch SQL Server Management Studio and connect to the Analysis Services instance that hosts your model.
2. In the Object Explorer, click to expand the Analysis Services instance, the Databases folder, the DirectQuery database, and the Tables folder.
3. Right-click the table that you want to modify and then click **Partitions**.
4. Select the DirectQuery partition.
5. On the toolbar, click the DirectQuery icon.
6. Change the **Processing Option** to **DirectQueryOnly** and then click **OK**.

## Browsing a DirectQuery enabled model in SQL Server Management Studio

Because models running in DirectQuery mode do not accept MDX queries, you cannot browse a model running in DirectQuery only or DirectQuery with In-Memory mode using the cube browser built in to SQL Server Management Studio. Also, you cannot launch Excel from SQL Server Management Studio to browse the model. If you try to perform these operations, you will receive the error message “DirectQuery error: MDX/SQL operations are not supported in DirectQuery mode.”

Instead, if you want to view data in a model running in DirectQuery mode from SQL Server Management Studio, you must issue a DAX query. For more information about constructing DAX queries, see “Additional resources.”

To execute a DAX query in SQL Server Management Studio:

1. Launch SQL Server Management Studio and connect to the Analysis Services instance that hosts your model.
2. Select the DirectQuery database in the Object Explorer.
3. Right-click the database**,** point to **New Query**, and then click **MDX** to start a query window.
4. Type your DAX query and then press F5. The DAX query is executed.

**Note** Red squiggles appear under DAX query keywords and Microsoft IntelliSense does not appear in the editing window. This is a limitation of the query window in SQL Server 2012. The DAX Editor, available at <http://daxeditor.codeplex.com/>, provides this functionality. Note that the DAX Editor runs in the Microsoft Visual Studio 2010 shell, not in SQL Server Management Studio.

You can also use SQL Server Management Studio to test security on a DirectQuery enabled model. To test security, modify the connection string for Analysis Services and pass the user name or role name that you want to test. If you are connecting to Power View using a BISM file, this is the only way to test security for DirectQuery enabled models. If you are using RSDS files, you can edit the connection string for the RSDS file to include the **EffectiveUserName** property for testing purposes.

**Note** You must be an administrator of the DirectQuery enabled model to perform this task.

To test security in SQL Server Management Studio:

1. If you are already connected to the Analysis Services tabular instance that you want to test, right-click the instance in the Object Explorer and then click **Disconnect**.
2. Connect to the Analysis Services instance you want to test. Click **Options** in the connection dialog box to expose more connection properties.
3. Click the **Additional Connection Parameters** tab. In the box, specify one of two things:

* To test a role, type **Roles=**<*role name*>. To test multiple roles, type **Roles=**<*role name 1*>**,** <*role name 2*>
* To test a user, type **EffectiveUserName=**<*Windows user name*> where <*Windows user name*> is in the form DOMAIN\username

1. Click **Connect**. Note that the Object Explorer shows all the objects on the server that the user connected to SQL Server Management Studio has permission to view. However, queries use the credentials specified in step 3.
2. Right-click the database you want to test, point to **New Query**, and then click **MDX** to start a query window.
3. Type your DAX query and then press F5. The DAX query is executed using the credentials supplied in step 3.

The following output shows the results when a user that does not have access to the underlying SQL Server data source executes the query evaluate DimSalesTerritory in DirectQuery Mode.

Executing the query ...

A connection could not be made to the data source with the DataSourceID of '36a84315-c076-486c-aa32-46beef1f6d24', Name of 'SqlServer .POWERPIVOT AdventureWorksDW2012'.

OLE DB or ODBC error: Login timeout expired; HYT00; A network-related or instance-specific error has occurred while establishing a connection to SQL Server. Server is not found or not accessible. Check if instance name is correct and if SQL Server is configured to allow remote connections. For more information see SQL Server Books Online.; 08001; SQL Server Network Interfaces: Error Locating Server/Instance Specified [xFFFFFFFF]. ; 08001.

Execution complete

## Changing the DirectQuery impersonation settings

The DirectQuery impersonation settings are not directly exposed in the SQL Server Management Studio user interface. To change the DirectQuery impersonation settings after the model has been deployed, you must modify the settings in an XML for Analysis (XMLA) script.

To change the DirectQuery impersonation information:

1. Launch SQL Server Management Studio and connect to the Analysis Services instance that hosts your model.
2. In the Object Explorer, click to expand the Analysis Services instance, the Databases folder, the DirectQuery database, and the Connections folder. The connection to the SQL Server data source for the model is now visible.
3. Right click the connection, point to **Script connection as**, point to **ALTER To**, and then click **New Query Editor Window**.
4. In the Connect to Analysis Services box that appears, click **Connect**. The definition of the connection appears in a new window.
5. Look for the following XMLA script element:

<ddl300:QueryImpersonationInfo>

<ImpersonationMode>ImpersonateCurrentUser</ImpersonationMode>

</ddl300:QueryImpersonationInfo>

1. Change the ImpersonationMode to the desired value, either ImpersonateCurrentUser or Default. For more information about the impersonation options, see “Setting the DirectQuery impersonation settings.”
2. Press F5 to execute the script. The impersonation settings for the model are changed.

## Clearing the xVelocity engine cache

You can clear the cache for a DirectQuery enabled model in SQL Server Management Studio by executing an XMLA script. It is useful to clear the cache when you have converted a DirectQuery enabled model running in a hybrid mode to a DirectQuery only model.

**Note** Clearing the cache removes objects in memory. It does not remove data from the model. To remove the data from the model, issue a Process Clear command.

To clear the xVelocity engine cache:

1. Launch SQL Server Management Studio and connect to the Analysis Services instance that hosts your model.
2. Select the DirectQuery database in the Object Explorer.
3. Right-click the database**,** point to **New Query**, and then click **XMLA** to start a query window.
4. Paste in the following code, changing the DatabaseID as necessary.

<ClearCache xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">

<Object>

<DatabaseID>DirectQueryProject</DatabaseID>

</Object>

</ClearCache>

1. Press F5 to execute the script. The cache is cleared.

# Monitoring a DirectQuery enabled model using SQL Server Profiler

SQL Server Profiler is a tool for monitoring and troubleshooting SQL instances, including Analysis Services instances. For general information about using SQL Server Profiler with Analysis Services, see [Introduction to Monitoring Analysis Services with SQL Server Profiler](http://technet.microsoft.com/en-us/library/ms174779(v=sql.110).aspx) (http://technet.microsoft.com/en-us/library/ms174779(v=sql.110).aspx).

Analysis Services exposes two events, DirectQuery Begin and DirectQuery End, specifically for monitoring DirectQuery enabled models. These events are triggered when a DAX query that has been translated to SQL is executed on the SQL Server instance. The DirectQuery Begin and DirectQuery End events contain the SQL statement that is executed on the data source for computation purposes. These events are especially useful if you do not have administrative access to SQL Server and you cannot capture these queries on the relational data source itself.

To capture the DirectQuery Begin and DirectQuery End events in SQL Server Profiler:

1. On the **File** menu, click **New Trace**.
2. Enter the name of the Analysis Services instance hosting the model and then click **Connect**.
3. Click the **Events Selection** tab of the **Trace Properties** box.
4. Select the **Show all events** check box.
5. Click to expand the **Query Processing** event category. Select the **DirectQuery Begin** and **DirectQuery End** check boxes and then click **Run**.

DirectQuery Begin and DirectQuery End events are not generated when a model running in hybrid mode is queried using the In-Memory query mode. Instead, the VertiPaq SE Begin and VertiPaq SE End events are generated, indicating that query results are computed using the xVelocity engine cache. For hybrid models, it may be helpful to capture the VertiPaq SE Begin and End events in addition to the DirectQuery Begin and End events to ensure query results are returned from the intended data source.

# Security considerations

Because security is a critically important part of data warehouse and data model design, information about security has already been presented in many sections of this white paper. This section summarizes the design patterns and provides information about user accounts and privileges necessary to use a DirectQuery enabled model.

## Security design patterns

The following table summarizes the major security considerations for a DirectQuery enabled model and the corresponding security designs.

|  |  |  |
| --- | --- | --- |
| Scenario | Design | References |
| Users are connecting to reports based on the model using credentials other than Windows credentials | Use RSDS files with stored credentials to connect to the model.  Security must be managed on the RSDS file, not on Analysis Services.  Default impersonation settings used.  Row security is not supported.  Excel or clients other than Reporting Services cannot be used to query the model. | Connecting to a DirectQuery enabled model using a RSDS file |
| Dynamic security or row security is required | All users must be Windows users.  The data warehouse must be configured for read access for each Power View user, with security defined on the data warehouse.  Analysis Services must impersonate the current user when it connects to the data source.  The model must be a DirectQuery enabled model, with all partitions configured to be DirectQuery only partition, to avoid storing data in the xVelocity engine cache.  Use BISM files to connect to the model, not RSDS files.  The workspace database cache must be emptied.  Excel or other MDX clients cannot be used to query the model. | Choosing a DirectQuery mode  Creating a DirectQuery only partition  Setting the DirectQuery impersonation settings  Securing the data warehouse  Managing the xVelocity engine cache in SQL Server Data Tools  Connecting to a DirectQuery enabled model using a BISM file |
| Connections from Power View to Analysis Services must cross SharePoint farm boundaries | If using BISM files for connections, configure Kerberos or make the Reporting Services application execution account an administrator on Analysis Services.  If using RSDS files, configure the RSDS files to impersonate a specific user or to use stored credentials. | Connecting to a DirectQuery enabled model using a BISM file  Connecting to a DirectQuery enabled model using a RSDS file |
| Connections from Analysis Services to SQL Server must cross machine boundaries | Configure Kerberos to enable delegation of credentials. | Additional resources |
| Data must be processed into and queried from the xVelocity engine cache | Use default DirectQuery impersonation settings.  All users with read access to the model can query all data in the cache.  In the Import Wizard, specify a Windows user with the minimum possible privileges to connect to the SQL Server data source for processing, because all users can see the data processed by this Windows user. | Setting the DirectQuery impersonation settings |

**Table 3:** Design patterns for security.

## Managing user access to the model

Read access must be managed in Power View, Analysis Services, SQL Server, and optionally on RSDS files.

The simplest way to manage access for Windows users is to define security groups in Active Directory Domain Services (AD DS) and then grant read access to those groups. The same group of users can be used to grant read access to the Power View reports, the BISM file, the RSDS file, and the Analysis Services model. If you are using the current Windows user’s credentials to connect to the SQL Server data source, this same group can be reused when granting access to SQL Server.

A Windows user with the minimum permissions level should be granted access to the SQL Server data source if the default DirectQuery impersonation settings are used. This Windows user should be specified in the impersonation settings in the Import Wizard, and this user’s credentials are used to connect to the model in both DirectQuery and In-Memory modes. In this scenario, an Active Directory group that contains the set of users with read access to the model is granted access only to Reporting Services and Analysis Services components, not to SQL Server.

## Enabling constrained delegation

As mentioned in the section “Setting the DirectQuery impersonation settings,” when the current user is impersonated, the current user’s credentials must be passed from Analysis Services to SQL Server. This is a multi-hop scenario, with credentials going from Power View to Analysis Services to SQL Server. One way to get credentials to flow through to SQL Server in this scenario is to enable Kerberos. Another way is to give the Analysis Services service account the Trusted Computer Base (TCB) privilege. The TCB privilege enables constrained delegation, which is used to pass the credentials from Analysis Services to SQL Server without enabling Kerberos. Constrained delegation works only if Analysis Services and SQL Server are on the same machine.

To grant the TCB privilege to the Analysis Services service account:

1. On the **Administrators** menu, click **Local Security Policy**.
2. Expand **Local Policies**, and then click **User Rights Assignments**.
3. Double-click the **Act as part of the operating system** policy.
4. Add the Analysis Services service account to the list of users allowed to act as part of the operating system.

Because the TCB privilege is very broad, it is not appropriate to grant this privilege in all security environments nor is it appropriate to grant to high privileged users. Use the privilege cautiously, and when in doubt use Kerberos to delegate credentials instead.

# Performance considerations

Analysis Services is optimized for use with the xVelocity engine cache, so query performance is better for models where the data set can fit in memory in Analysis Services.

Consider the following query that tests the Internet Current Quarter Margin measure created in the section “Calculating rolling sums using basic DAX.”

evaluate

SUMMARIZE(

DimDate,

DimDate[CalendarYear],

DimDate[CalendarQuarter],

"Margin QTD",

FactInternetSales[Internet Current Quarter Margin])

ORDER BY DimDate[CalendarYear], DimDate[CalendarQuarter]

On one test machine, with both Analysis Services and SQL Server on the same machine, executing this query in DirectQuery mode took eight seconds. Executing this same query in In-Memory mode took one second. In this case, the performance gains for using the xVelocity engine cache may outweigh the benefits of DirectQuery mode.

Usually, adding a columnstore index improves DirectQuery performance. In this case, adding a columnstore index to the DimDate and FactInternetSales tables reduced the query time to seven seconds.

Another way to improve DirectQuery performance is to optimize DAX queries and expressions. Some DAX queries and expressions that perform well in In-Memory mode do not perform well in DirectQuery mode.

For example, the following query does not perform well in DirectQuery mode.

EVALUATE

CALCULATETABLE(

SUMMARIZE(

FILTER(

ORDERS,

ORDERS[O\_ORDERDATE] >= DATE(1993,10,01) &&

ORDERS[O\_ORDERDATE] < EDATE(DATE(1993,10,01), 3)

),

ORDERS[O\_ORDERPRIORITY],

"ORDER\_COUNT",COUNTROWS(ORDERS)

),

FILTER(LINEITEM, [L\_COMMITDATE] < [L\_RECEIPTDATE])

)

ORDER BY ORDERS[O\_ORDERPRIORITY]

This query calculates the number of orders over a table with multiple filters applied. A better approach when a table has many filters applied is to add measures explicitly using ADDCOLUMNS instead of summarizing the measure.

The following functionally equivalent query performs much better in DirectQuery mode.

evaluate addcolumns(values(ORDERS[O\_ORDERPRIORITY]), "x",

calculate(

countrows(

summarize(

filter(LINEITEM,

[L\_COMMITDATE] < [L\_RECEIPTDATE]

&& related(ORDERS[O\_ORDERDATE]) >= DATE(1993, 10,01)

&& related(ORDERS[O\_ORDERDATE]) < DATE(1994, 1, 1)

)

, orders[o\_orderkey]

)

)

)

)

ORDER BY ORDERS[O\_ORDERPRIORITY]

Here is another example of a query that does not perform well in DirectQuery mode.

EVALUATE SUMMARIZE(

FILTER(

LINEITEM,

LINEITEM[L\_SHIPDATE] >= DATE(1993,01,01) &&

LINEITEM[L\_SHIPDATE] < DATE(1994,01,01) &&

LINEITEM[L\_DISCOUNT] >= 0.04 - 0.01 &&

LINEITEM[L\_DISCOUNT] <= 0.04 + 0.01 &&

LINEITEM[L\_QUANTITY] < 25

),

"REVENUE",SUMX(LINEITEM, LINEITEM[L\_EXTENDEDPRICE]\*(LINEITEM[L\_DISCOUNT]))

)

This query summarizes a measure named “REVENUE” over a table with several filters applied. In this case, the performance problem happens because there is no explicit relationship between the REVENUE measure and the filtered table used for computing results. The SQL query generated when there is no explicit relationship defined between the measure and the table is inefficient.

A better approach is to use explicit aggregations over filtered tables. In this case, you can move the SUMX function to the outside of the query and then add in the filtered table explicitly as the first argument to SUMX to create a direct relationship between the filtered table and the aggregation.

The following functionally equivalent query performs much better in DirectQuery mode.

evaluate row("x",

sumx(

filter(lineitem,

LINEITEM[L\_SHIPDATE] >= DATE(1993,01,01) && LINEITEM[L\_SHIPDATE] < DATE(1994,01,01)

&& LINEITEM[L\_DISCOUNT] >= 0.04 - 0.01 && LINEITEM[L\_DISCOUNT] <= 0.04 + 0.01

&& LINEITEM[L\_QUANTITY] < 25

)

,[L\_EXTENDEDPRICE] \* [L\_DISCOUNT]

)

)

We recommend that you create a prototype using your specific workload to ensure satisfactory performance. The performance characteristics differ between DirectQuery and in-memory models. Not all in-memory models are suited for conversion to DirectQuery enabled models.

# Conclusion

DirectQuery provides pass through access to SQL Server data sources. The primary benefit of DirectQuery is that data need not be loaded into the Analysis Services database and incremental updates need not be managed in Analysis Services. Other benefits of DirectQuery include real time data access, support for potentially very large data sets, and support for the SQL Server security model. DirectQuery is not suitable for all scenarios, because it does have some limitations that do not apply to in-memory models. Each model must be evaluated on a case-by-case basis to see whether DirectQuery is the right solution.

DirectQuery enabled models must be designed using different design patterns than in-memory tabular models. These differences begin in the data warehouse and extend to the data model, affecting table definitions, partitions, security settings, and DAX calculations. There are special considerations as well when you manage and connect to DirectQuery enabled models. Processing and cache management strategies differ. Connection strings may need to be modified, and credentials may need to be handled differently. The security model for DirectQuery enabled models is different.

This paper showed some techniques for designing, securing, and managing DirectQuery enabled models. Adopting these techniques will speed development, improve design, and improve security.

# Additional resources

DirectQuery Mode (SSAS) <http://msdn.microsoft.com/en-us/library/hh230898(v=sql.110).aspx>

SQL Server 2012 Security Best Practice <http://download.microsoft.com/download/8/F/A/8FABACD7-803E-40FC-ADF8-355E7D218F4C/SQL_Server_2012_Security_Best_Practice_Whitepaper_Feb2012.docx>

Implementing Row- and Cell-Level Security in Classified SQL Server Databases <http://download.microsoft.com/download/8/8/0/880F282A-AA4E-4351-83C0-DFFA3B56A19E/SQL_Server_2012_RLS_and_CLS_White_Paper_January2012.docx>

DAX Query Syntax Reference <http://msdn.microsoft.com/en-us/library/gg492156(v=sql.110).aspx>

DAX Query posts, Chris Webb

* DAX Queries, Part 1 <http://cwebbbi.wordpress.com/2011/07/13/dax-queries-part-1/>
* DAX Queries, Part 2 <http://cwebbbi.wordpress.com/2011/07/15/dax-queries-part-2/>
* DAX Queries, Part 3 <http://cwebbbi.wordpress.com/2011/07/18/dax-queries-part-3/>
* DAX Queries, Part 4 <http://cwebbbi.wordpress.com/2011/07/27/dax-queries-part-4/>
* DAX Queries, Part 5 <http://cwebbbi.wordpress.com/2011/08/06/dax-queries-part-5/>
* DAX Queries, Part 6 <http://cwebbbi.wordpress.com/2011/08/10/dax-queries-part-6/>

Using parameters in your DAX queries, Marco Russo <http://sqlblog.com/blogs/marco_russo/archive/2012/01/05/use-parameters-in-your-dax-queries.aspx>

DAX Editor <http://daxeditor.codeplex.com/>

Project Crescent Solution Configuration, Bob Meyers and Dave Wickert, SQL PASS Summit 2011. Session information and PowerPoint download at <http://www.sqlpass.org/summit/2011/Speakers/CallForSpeakers/SessionDetail.aspx?sid=2027>.

Deployment checklist: Reporting Services, Power View, and PowerPivot for SharePoint <http://msdn.microsoft.com/en-us/library/hh231687(v=sql.110).aspx>

How to configure SQL Server 2008 Analysis Services and SQL Server 2005 Analysis Services to use Kerberos authentication <http://support.microsoft.com/kb/917409>

Plan for Kerberos Configuration (SharePoint Server 2010) <http://technet.microsoft.com/en-us/library/ee806870.aspx>

Kerberos configuration checklist <http://blogs.msdn.com/b/psssql/archive/2010/06/23/my-kerberos-checklist.aspx>

Power View, Tabular mode databases, SharePoint, and Kerberos by Kasper de Jonge <http://www.powerpivotblog.nl/power-view-tabular-mode-databases-sharepoint-and-kerberos>

**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](mailto:sqlfback@microsoft.com?subject=White%20Paper%20Feedback:%20[DirectQuery%20in%20the%20Tabular%20BI%20Semantic%20Model]).