Denodo Power BI Custom Connector - User Manual

Revision 20230320

Denodo Inc.: 525 University Avenue, Suite 31, Palo Alto, CA 94301. USA www.denodo.com

CONTENTS

1 OVERVIEW

Microsoft Power BI is a suite of business analytics tools for analyzing data and sharing insights. Microsoft Power BI Desktop is part of these tools and it lets you build advanced queries, models and reports that visualize data and can easily be shared with others. Power BI Desktop combines data from disparate databases, files, and web services with visual tools that help you understand and fix data quality and formatting issues automatically.

The Denodo Power BI Custom Connector allows users of Power BI Desktop to connect to and access data at the Denodo Platform making it easy for users to query, shape and mashup data from your Denodo Platform databases to build reports and dashboards that meet the needs of your organization. It also enables Power BI to access Denodo data in *DirectQuery* mode, which reduces the needs to preload the data.

2 INSTALLATION

2.1 INSTALLING THE DENODO ODBC DRIVER

Given that Denodo Power BI Custom Connector is an ODBC based connector, it is necessary to install the Denodo ODBC driver in the machine that will be configuring and publishing an ODBC DSN for Power BI to use. In order to install the driver, go to the `<DENODO_HOME>\tools\client-drivers\odbc` folder, where you can find two `.zip` files:

- `DenodoODBC_x86.zip` contains the ODBC driver for 32-bit clients.
- `DenodoODBC_x64.zip` contains the ODBC driver for 64-bit clients.

Unzip the appropriate driver version depending on the Microsoft Power BI Desktop version that you are going to use and then execute the `.msi` file extracted. A simple installation wizard will guide you through the process.

2.2 CREATING THE DSN

Once you have installed the Denodo ODBC driver (see the **Installing the Denodo ODBC driver section**) you will need to add a new ODBC data source (DSN). On Windows systems:

1. Go to `Control Panel > Administrative Tools > Data Sources (ODBC)`.
2. Select `Add User DSN` or `Add System DSN`. The difference is that `User DSN` can only be used by the current user and `System DSN` can be used by all the users of the system.
3. Select the `DenodoODBC Unicode` driver and click on the `Finish` button.

In the configuration dialog fill in the following information:

- **Data Source** (mandatory): name of the ODBC data source (e.g. Power BI - Denodo).
- **Database** (mandatory): database in Denodo Virtual DataPort.
- **Server** and **Port** (mandatory): host name and port of the server that runs Virtual DataPort. The default ODBC port is 9996.
- **User Name / Password**: credentials to connect to Denodo. These fields may be empty because the Denodo Power BI Custom Connector will request authentication data from Power BI.

In order to get more information regarding the setup of a DSN, check the Denodo Platform documentation.

2.3 CREATING A DENODO COMPATIBLE CONNECTION STRING

When creating a Denodo-compatible connection string, we must take into account that the Driver field must be omitted, as this will be transparently set at connection time by the connector itself.

The connection string shall have three mandatory parameters: SERVER, PORT and DATABASE:

```
Server=<Server name>;DATABASE=<Database name>;PORT=<Port number>
```

Additionally, it can contain an optional parameter: **SSLmode**:

```
Server=<Server name>;DATABASE=<Database name>;PORT=<Port number>;SSLmode=<SSL mode>
```

When writing the connection string, it must be taken into account:

1. The connection string must keep the correct order of its parameters: SERVER, PORT, DATABASE and SSLMode.
2. The name of these parameters must always be written in the same way. For example, if you choose to write them in upper case, they must always be written in upper case; if you decide to write them capitalized (writing the first letter of a word in uppercase and the rest of the letters in lowercase) they must always be written that way.

Authentication parameters must be omitted, as authentication is configured in later steps.

2.4 SELECTING A DENODO POWER BI CUSTOM CONNECTOR

2.4.1 Certified Denodo Connector

Denodo Power BI Custom Connector is certified by Microsoft since the October 2018 release, so you can use the Denodo connector included in Microsoft Power BI. The use of this certified Denodo Connector is the recommended option unless you are using a version of Microsoft Power BI prior to the October 2018 release or if you want to benefit from a patch or fix of the Denodo Power BI Custom Connector (see the **DenodoConnect component section**) that has not yet been published in Microsoft Power BI or On-premises data gateway.

2.4.2 DenodoConnect component

The Denodo Power BI Custom Connector is also available on the DenodoConnects section of the Denodo support site. You can download this component and then follow the steps in the **Configuring Microsoft Power BI Desktop** and **Installing the Denodo Power BI Custom Connector sections** in order to use this Denodo Connector when your Microsoft Power BI does not include the Denodo Power BI Custom Connector (releases prior to October 2018) or when the Denodo Power BI Custom Connector version available on the support site offers a patch or fix that has not yet been published in Microsoft Power BI or On-premises data gateway.

Note that installing the Denodo Power BI Custom Connector is a temporary solution while the required version is not published in Microsoft Power BI or On-premises data gateway. Once the Certified Denodo Connector, pre-installed in Microsoft Power BI, is updated to the version you need, you should uninstall the Denodo Power BI Custom Connector (see the **Uninstalling the Denodo Power BI Custom Connector section**).

2.4.2.1 Configuring Microsoft Power BI Desktop

If you need to use a different version of the Denodo connector than the one pre-installed in Microsoft Power BI Desktop, you will need to lower the security level for extensions in Power BI Desktop to enable loading unsigned/uncertified connectors:

1. Go to `File > Options and settings > Options`.
2. Go to the `Security` tab.
3. Under `Data Extensions`, select `Allow any extension to load without warning or validation`.

4. Restart Power BI Desktop.

2.4.2.2 Installing the Denodo Power BI Custom Connector

You should skip this step if you are going to use the certified connector for Denodo included in Power BI Desktop. Otherwise, once you have installed the correct driver version and configured Microsoft Power BI Desktop, you must install the Denodo Power BI Custom Connector.

The distribution consists of:

- A `dist` folder containing the `.mez` binary file for the connector.
- A `doc` folder containing this user manual.
- README, NOTICE and RELEASE NOTES files that contain information about the Denodo Power BI Custom Connector.

In order to use the Denodo Power BI Custom Connector in Power BI Desktop, you must copy the `.mez` file into:
`C:\Users\[userName]\Documents\Power BI Desktop\Custom Connectors`

You will probably have to create the last two folders in this path, therefore, make sure you create them exactly as spelled above.

2.4.2.3 Uninstalling the Denodo Power BI Custom Connector

When the version of the certified connector for Denodo included in Power BI contains the necessary fixes or updates, uninstall the Denodo Power BI Custom Connector. To do this, delete the `.mez` file that you had copied into:

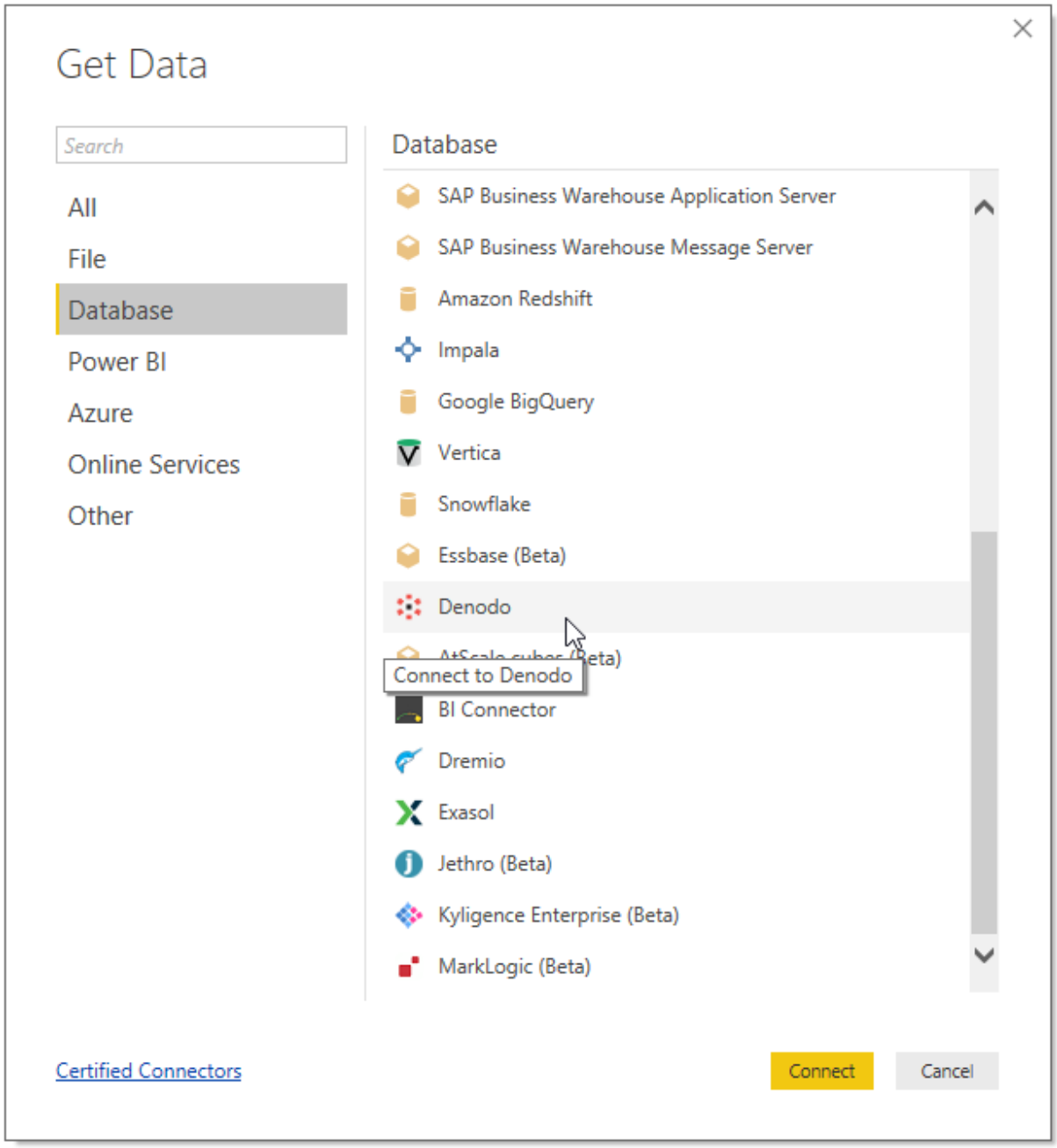
`C:\Users\[userName]\Documents\Power BI Desktop\Custom Connectors`

Using the On-premises data gateway, the `[userName]` is the user running the gateway service which is `NT SERVICE\PBIEgwService` by default.

3 USING THE DENODO POWER BI CUSTOM CONNECTOR

Whether you are going to use a non-certified Denodo connector, after installing the Denodo Power BI Custom Connector you have to start, or restart, Power BI Desktop before beginning to work with it.

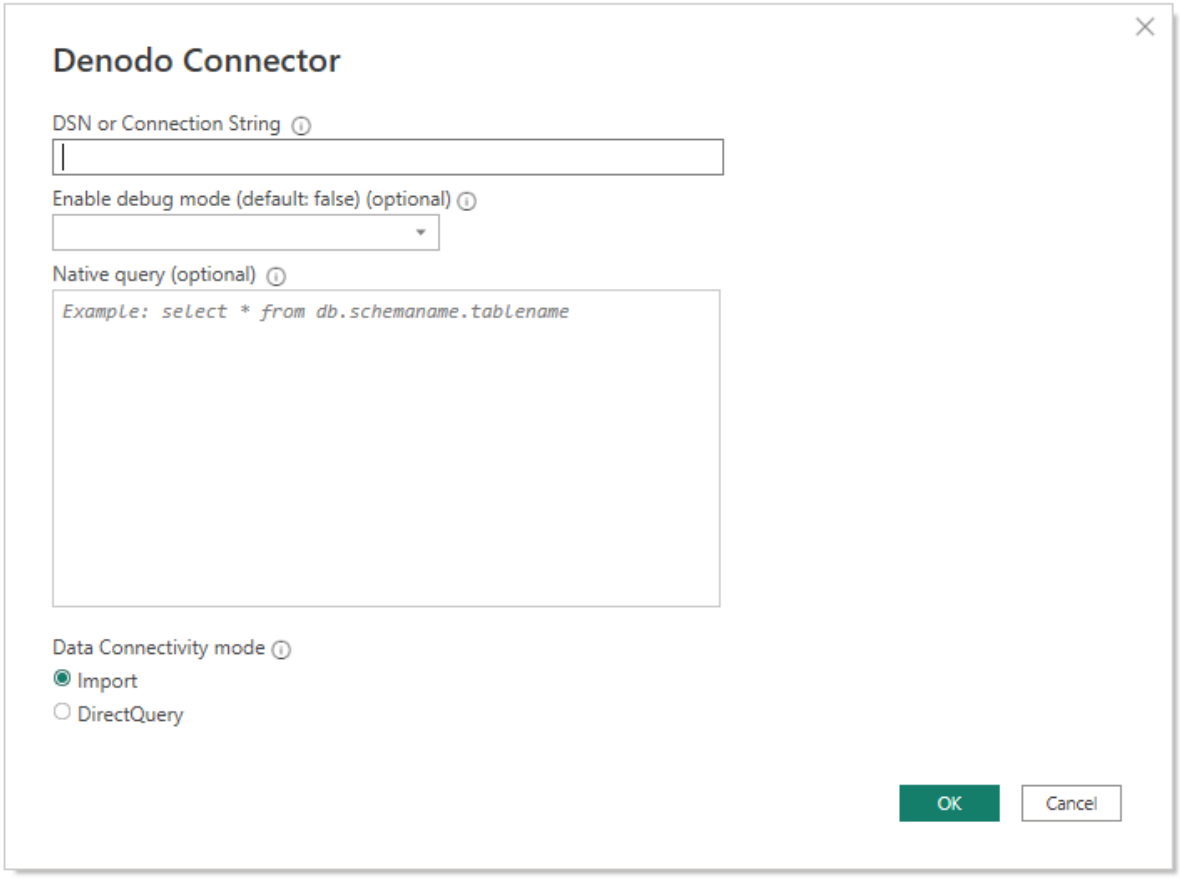
In order to connect to data, select the `Get Data` icon from the Home ribbon and select `Denodo` in the `Database` section.



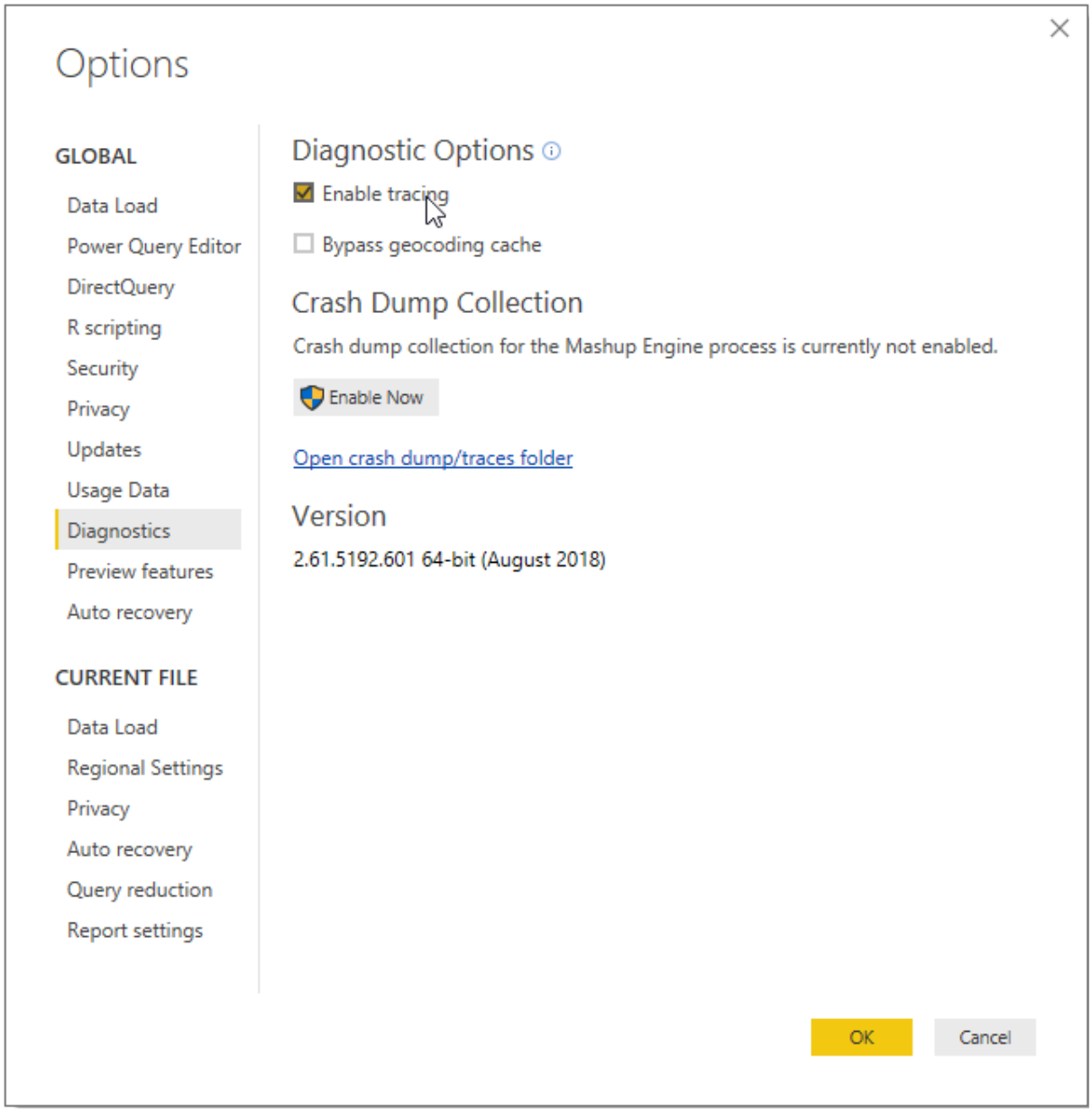
There are two ways to connect to the data source of your choice:

- Via DSN (ODBC data source name)
- Using a connection string

A new window will ask you for the DSN (see the **Creating the DSN section**) or a Connection String (see the **Creating a Denodo compatible connection string**), the option to enable the debug mode and the connectivity mode that can be `Import` or `DirectQuery`.



The `Enable debug mode` is an optional field that allows you to add trace information to log files. These files are created by Power BI Desktop when you enable tracing in the application using the `Diagnostics` tab in the `Options` menu. Note that the default value for `Enable debug mode` is false and in this scenario there will be no trace data, from Denodo Power BI Custom Connector, in the log files.



The `Native Query` is an optional field where you can enter a query. If this query field is used, the resulting dataset will be the result of the query instead of a table or a set of tables.

You can write a query that queries only one of the databases that the datasource is associated with.

```
SELECT title, name FROM film JOIN language ON film.language_id = language.language_id WHERE film.language_id = 1
```

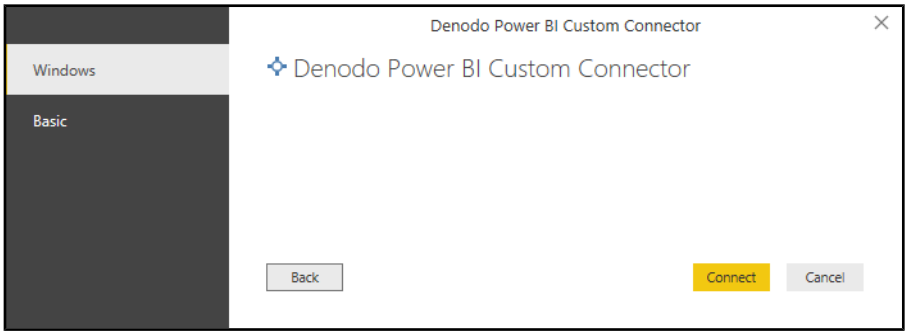
If you want to write a query that queries more than one database you have to indicate in the query the database that owns each table.

```
SELECT i_item_sk, country FROM sakila.country, ewd.item
```

The next step, before showing the Navigator window that displays a preview of the available data in Denodo Virtual DataPort, will request authentication data. The Denodo Power BI Custom Connector supports two authentication types: **Windows** and **Basic**.

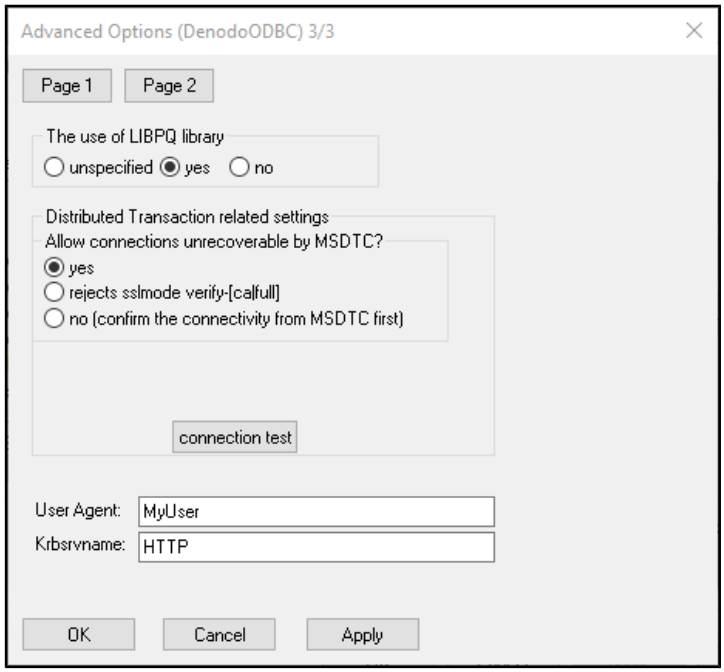
3.1 WINDOWS AUTHENTICATION

When you decide to use Windows authentication, Power BI Desktop is going to connect to Virtual DataPort using Kerberos authentication.



You will need:

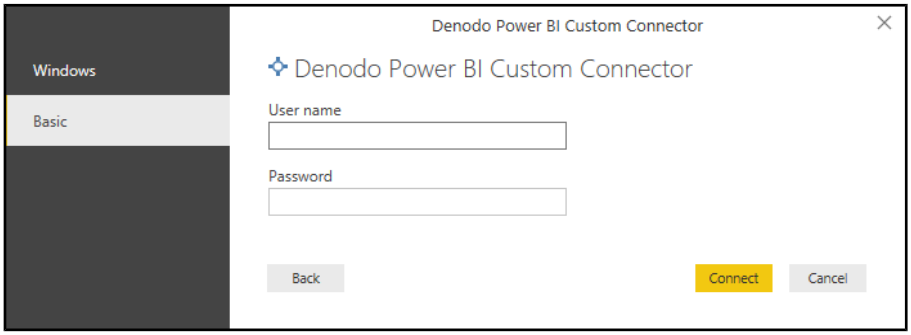
- Kerberos authentication must be enabled in the Virtual DataPort server.
- The Denodo Virtual DataPort database that the data source connects to must be configured with the option `ODBC/ADO.net authentication type` set to `Kerberos`.
- Power BI Desktop must be running in the Windows domain, because the ODBC driver requests the Kerberos ticket from the operating system's ticket cache.
- Make sure the **Advanced Options** page of the DSN configuration contains all the needed configuration for using Kerberos as an authentication method:



For more information, check the setting up process of a DSN in the Denodo Platform documentation.

3.2 BASIC AUTHENTICATION

This authentication type allows you to connect Power BI Desktop to your Virtual DataPort data using your Virtual DataPort server credentials.



3.3 USE OF DIRECTQUERY

Note that, using the DirectQuery mode for accessing data, changes to the underlying data are not necessarily reflected in the visualizations immediately. Reports or dashboards are refreshed when open (or explicitly refreshed), but otherwise Power BI does keep recently requested data for some time (usually 15 minutes per tile).

See the links below for more information:

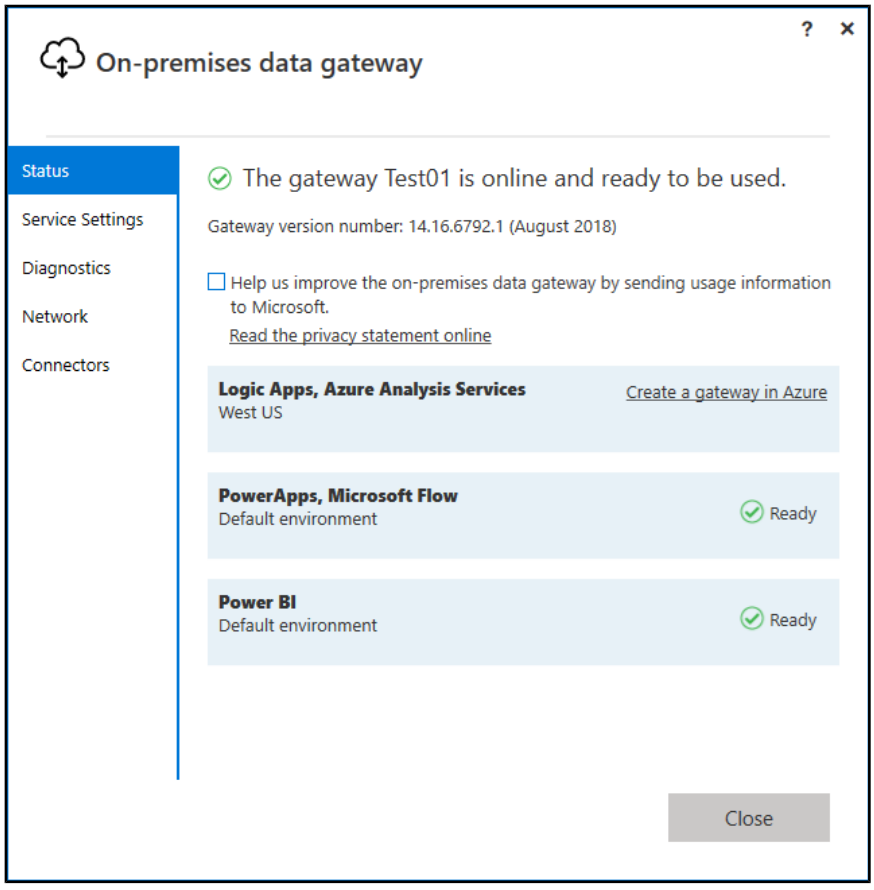
https://powerbi.microsoft.com/en-us/documentation/powerbi-desktop-use-directquery/
https://powerbi.microsoft.com/en-us/documentation/powerbi-desktop-directquery-about/

3.4 USING THE ON-PREMISES DATA GATEWAY

The On-premises data gateway (Enterprise gateway) acts as a bridge, providing quick and secure data transfer between on-premises data (data that is in your Power BI Desktop application, not in the cloud) and the Power BI Service, also known as Power BI online, app.powerbi.com or Power BI Cloud. Hence, when you build reports in Power BI Desktop that use Denodo Power BI Custom Connector, you can use the On-premises data gateway to publish and refresh those reports from the Power BI Service. Note that when you publish a report, the associated dataset is also imported, so you can also use it from Power BI Service in order to create new reports.

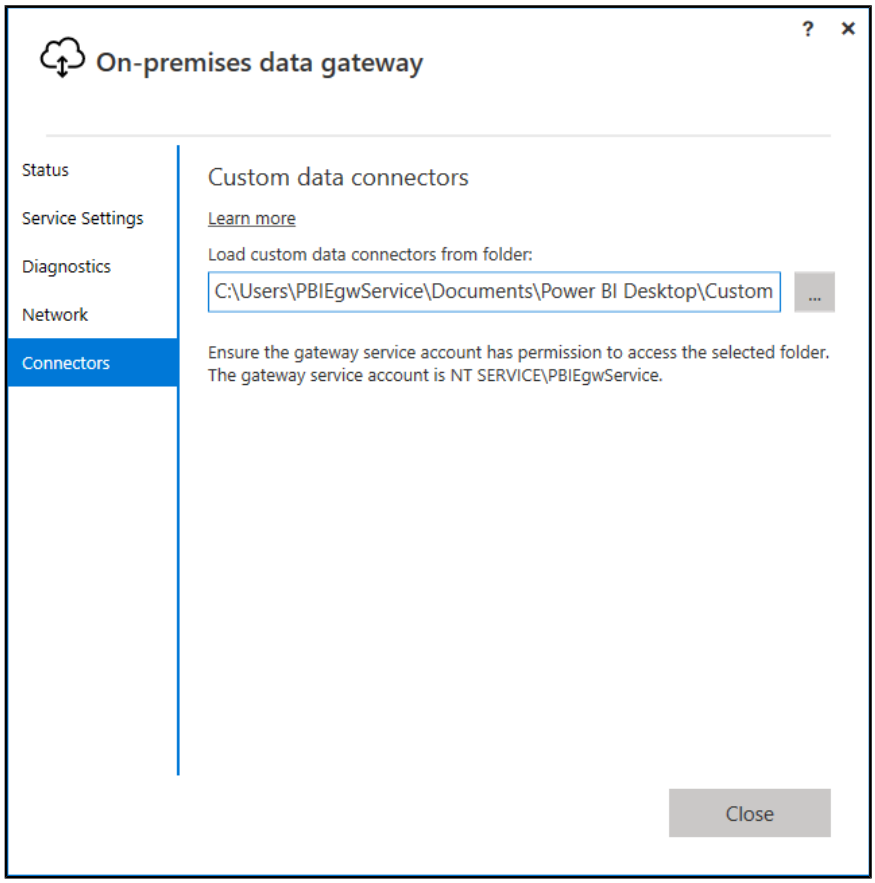
To enable and use this capability you have to install the August 2018 version of the On-premises data gateway or a later version.

After the installation, you have to sign in and register your gateway. In the On-premises data gateway configurator, selecting the `Status` tab will allow you to verify that your gateway is online and ready to be used.

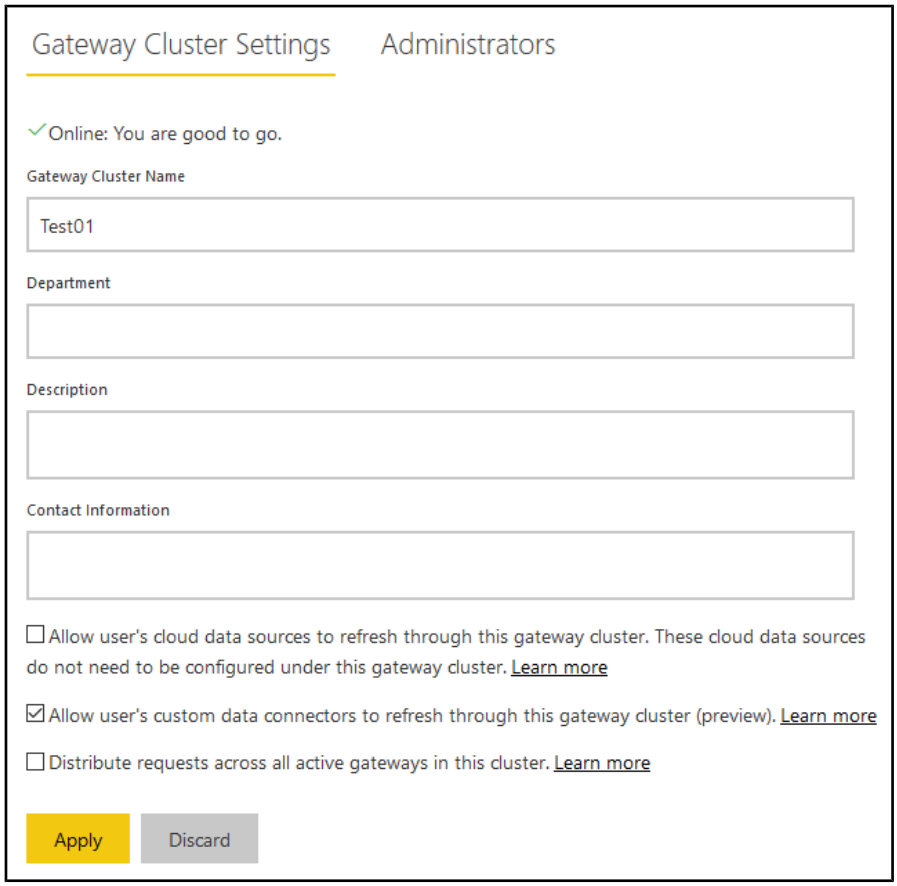


Additionally, you can see a `Connectors` tab with an option to choose a folder to load the custom connectors from in case you want to use a version of the Denodo connector other than the one included in Microsoft Power BI. Make sure you pick a folder that can be accessed by the user running the gateway service which is `NT SERVICE\PBIEgwService` by default. The gateway will automatically load the custom connector files located in that folder and you should see them in the list of the data connectors. Therefore, in order to allow connectivity via the Microsoft On-

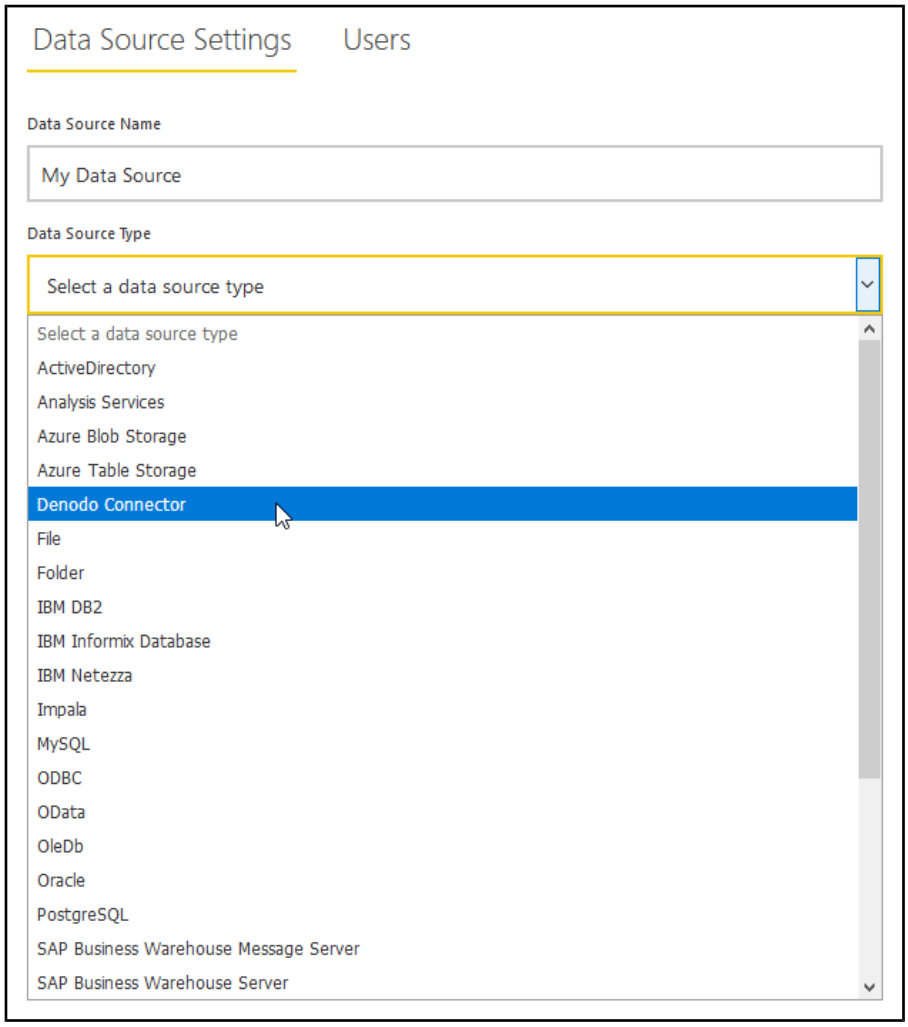
Premises Data Gateway using a Denodo Power BI Custom Connector, different to the one incorporated in Microsoft Power BI, you have to copy the `.mez` file into the folder specified in this step (`C:\Users\PBIEgwService\Documents\Power BI Desktop\Custom Connectors` by default). Note that when adding or changing a connector you must restart the On-Premises Data Gateway.



After configuring the On-premises data gateway, you need to create a data source for the Denodo Power BI Custom Connector in the Power BI Service using the gateway settings page. Since the ability to connect Power BI Service via the Gateway was introduced in August 2018 as a "Preview" feature, depending on the version of your On-premise data gateway or Power BI Service, you may need to enable the use of custom connectors checking the option `Allow user's custom data connectors to refresh through this gateway cluster`.



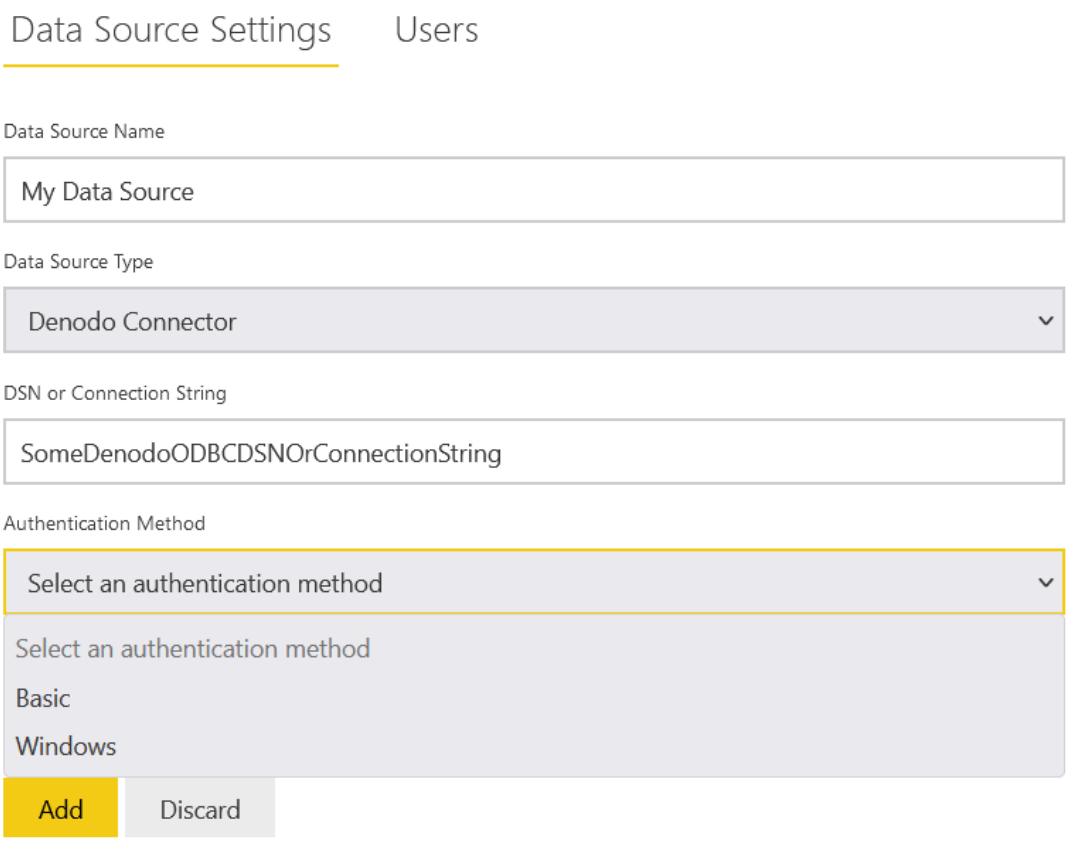
Now, you will see the `Denodo Connector` as an available data source that you can create under this gateway cluster.



In order to create the data source, you have to specify the way to connect to the data source of your choice:

- Via DSN (see the **Creating the DSN section**)
- Using a connection string (see the **Creating a Denodo compatible connection string**)

And also you have to specify the authentication mode too.



The authentication methods available are **Basic** and **Windows**, just as when using the connector in Power BI Desktop.

Basic authentication

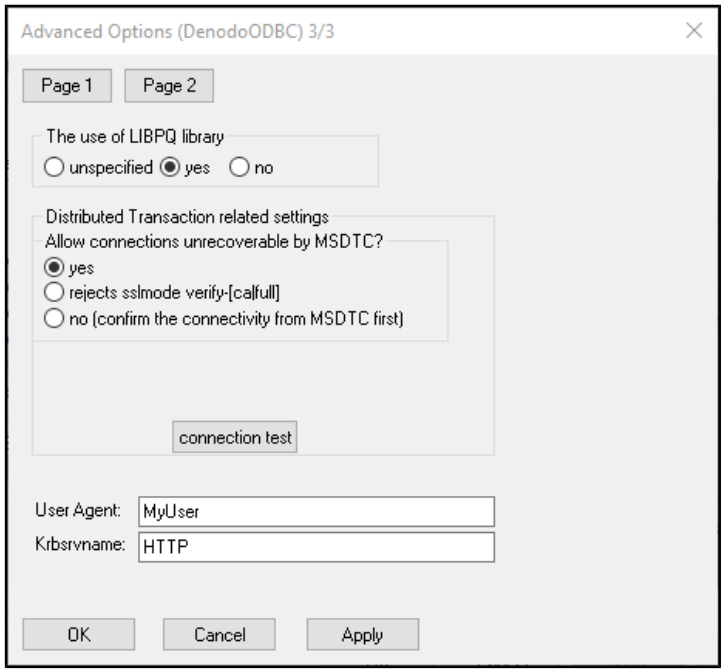
This authentication type allows you to create a Data Source in Power BI Service to connect to your Virtual DataPort data using your Virtual DataPort server credentials.

Windows authentication

When you decide to use Windows authentication, Power BI Service is going to connect to Virtual DataPort using Kerberos authentication.

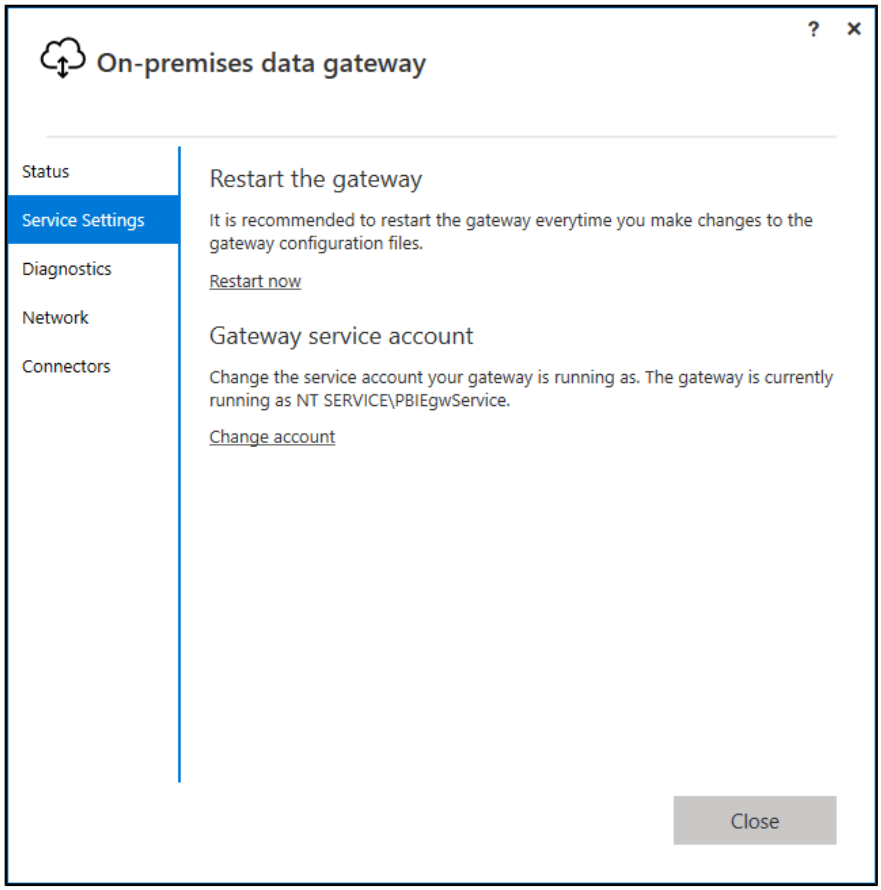
You need:

- In the Data Source Settings, enter the username and password to create the Kerberos ticket.
- Kerberos authentication enabled in the Virtual DataPort server.
- The Denodo Virtual DataPort database that the DSN connects to must be configured with the option `ODBC/ADO.net authentication type` set to `Kerberos`.
- In the advanced options of the DSN configuration, consider the type of authentication you want to use:



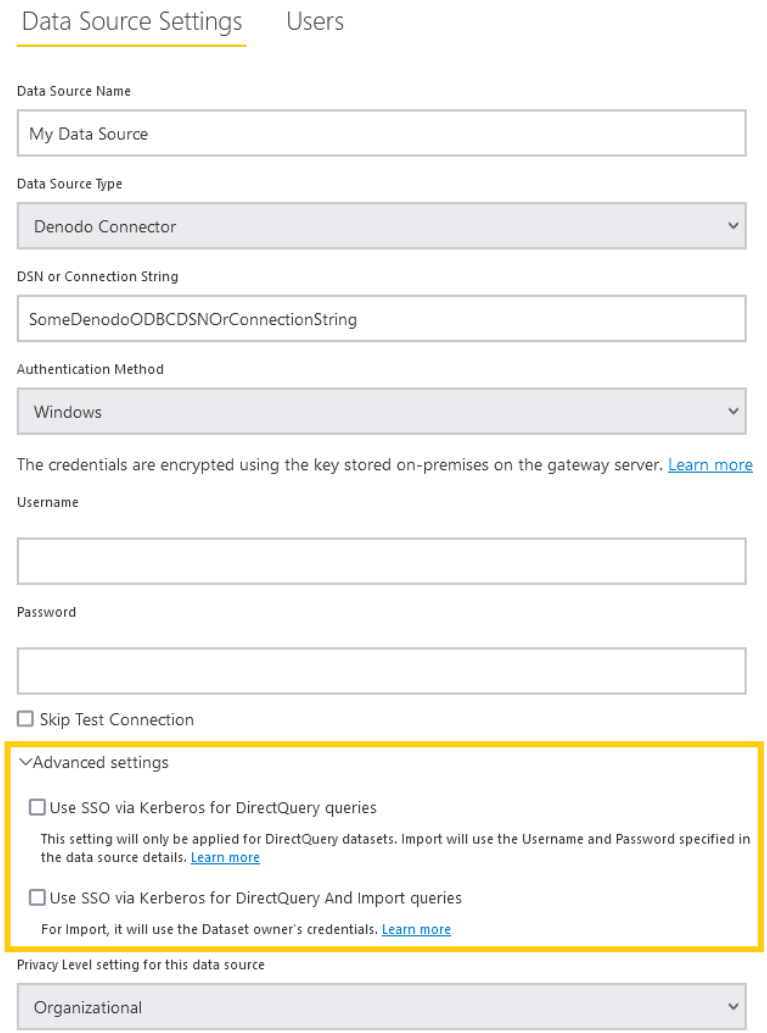
For more information, check the setting up process of a DSN in the Denodo Platform documentation.

Note that the gateway runs in a service account, `NT SERVICE\PBIEgwService` by default, so the DSN must be a System DSN or must be defined by the user who runs the gateway.



3.4.1.1 Single Sign-On (SSO)

If you use Windows authentication, you can enable (under the `Advanced settings` of the data source) the Single Sign-On authentication schema (SSO) in order to use the same credentials of the user accessing your reports at Power BI for accessing the required data in Denodo.



There are two options for enabling SSO: `Use SSO via Kerberos for DirectQuery queries` and `Use SSO via Kerberos for DirectQuery And Import queries`. If you are working with DirectQuery based reports, both use the SSO credentials of the user that signs in to the Power BI service. The difference comes when you work with Import based reports: in this scenario, the former option uses the credentials entered in the data source page (`Username` and `Password` fields), while the latter uses the credentials of the dataset owner.

It is important to note that there are certain prerequisites and considerations that you must take into account in order to use the Kerberos-based SSO. Some of these important requirements are:

- Kerberos constrained delegation must be enabled for the Windows user running the Microsoft Power BI Gateway, and configuration of both the local Active Directory and Azure Active Directory environments should be performed according to the instructions offered by Microsoft for this purpose.
- By default, the Microsoft Power BI Gateway sends the `user principal name` (UPN) when it performs an SSO authentication operation. Therefore, you will need to review the attribute that you will use as a login identifier in Denodo Kerberos Authentication and, if it is different from `userPrincipalName`, adjust the Gateway settings according to this value.

The Microsoft Power BI Gateway configuration file called `Microsoft.PowerBI.DataMovement.Pipeline.GatewayCore.dll.config`, stored at `\Program Files\On-premises data gateway` has two properties called `ADUserNameLookupProperty` and `ADUserNameReplacementProperty` that allow the Gateway to perform local Azure AD lookups at runtime. The `ADUserNameLookupProperty` must specify against which attribute of the local AD it must map the `user principal name` that comes from Azure AD so, in this scenario, `ADUserNameLookupProperty` should be `userPrincipalName`. Then, once the user is found, the `ADUserNameReplacementProperty` value indicates the attribute that should be used to authenticate the impersonated user (the attribute that you will use as login identifier in Denodo).

It should be taken into account that changes in this configuration file are at the Gateway level and therefore will affect any source with which SSO authentication is done through the Microsoft Power BI Gateway.

Please, check the Configure Kerberos-based SSO from Power BI service to on-premises data sources documentation for any further information.

Once a data source is created for the Denodo Connector, you can refresh Power BI reports. To publish a report on powerbi.com you will need to:

1. Select from the menu File: Publish > Publish to Power BI
2. Save the report on the computer.
3. Select the workspace where you wish to publish.

3.5 USE OF NATIVE QUERY

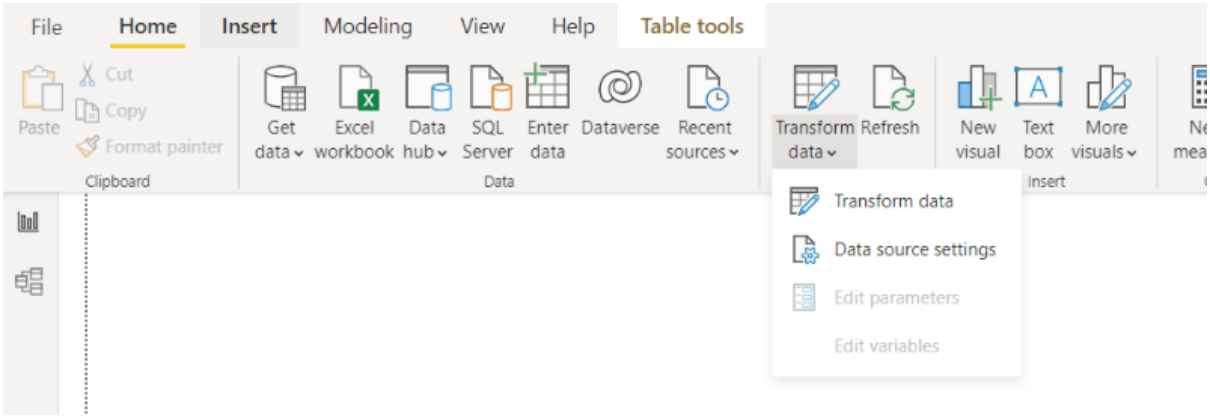
The Native Queries offer the possibility to import data into Power BI based on the use of Queries.

There are two ways to create the native query:

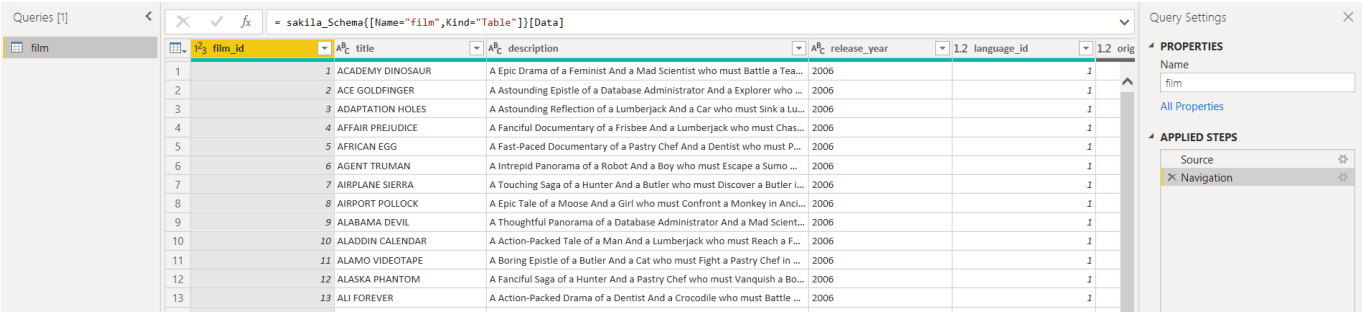
1. You can modify a table imported in PowerBI by adding the Native Query to its definition.

   First, you import the table or tables from which you want to sift the information.

   Once they are loaded in PowerBI, access the data transformation menu, `Home > Transform data`.



In this menu we can see the list of queries (each imported table corresponds to a query), the data corresponding to each one of them and the steps that are applied to the datasource to obtain these data.



In order to introduce the native query we must select the table we want to modify in the queries list and access the advanced editor, `View > Advanced Editor`. In the definition we can see how to obtain the source, the database, the schema and finally the table from which the data is obtained.

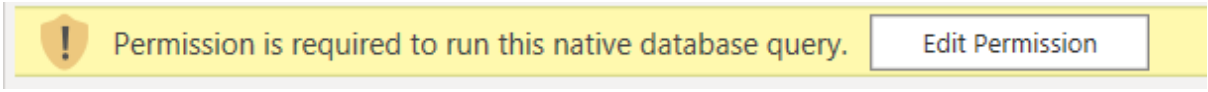
To create the native query the following function will be used: `Value.NativeQuery(target as any, query as text, optional parameters as any, optional options as nullable record) as any`

- `Target`: provides the context for the operation described by query
- `Query`: describes the query to be executed against target
- `Optional parameters`: may contain either a list or record as appropriate to supply the parameter values expected by query.
- `Optional options`: may contain options that affect the evaluation behavior of query against target.

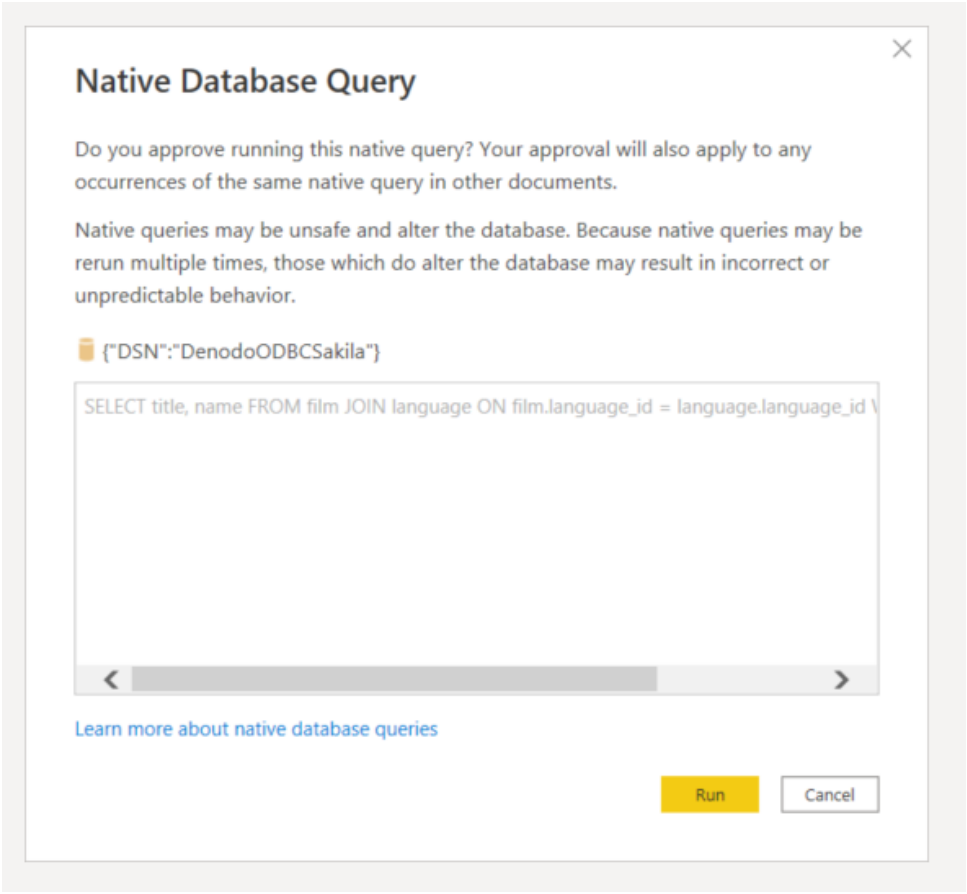
In the options section you can define in which mode you want to retrieve the query information: import or direct query. For this purpose, the EnableFolding option is used, whose true value will activate the direct query mode. If this option has false value or does not appear in the list of options, the data will be retrieved in import mode.

In order to create our native query we must first obtain a variable with the value of the target and then use the `Value.NativeQuery` function.

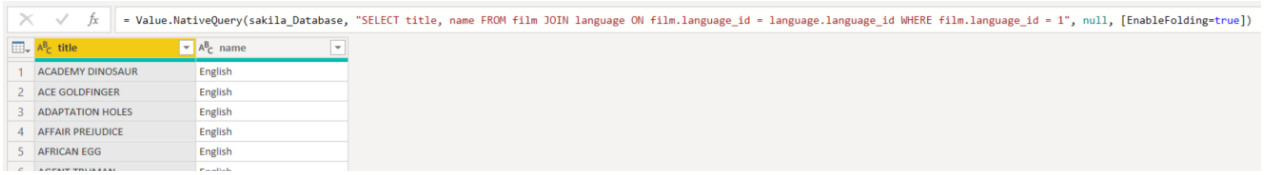
When finished, select `Done`, the advanced editor window will close and a message will appear asking for permission to execute the native query.



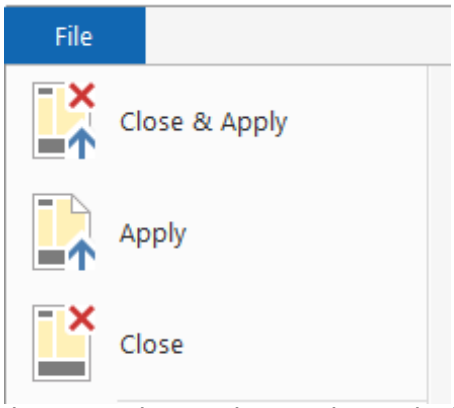
If you select `Edit Permission` you could see a window with the query to be executed and you must choose if you want to execute the query or cancel the process.



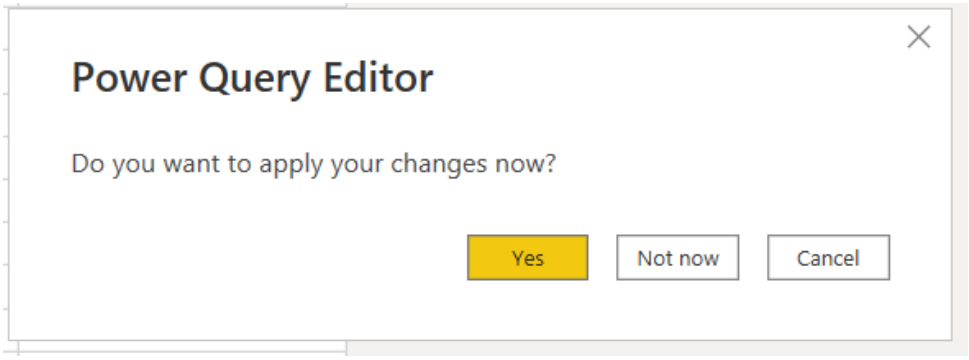
If the query has been executed, the returned data will be shown.



To apply the modifications made in the transform data window, go to the menu `File > Apply`.

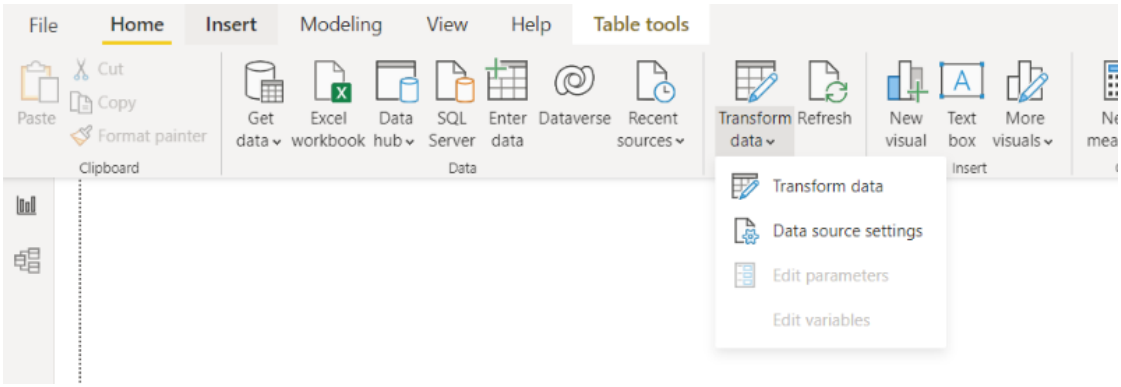


If this has not been done when closing the transform data window, a dialogue box will appear asking if you want to apply the changes introduced at this moment or leave it for later.

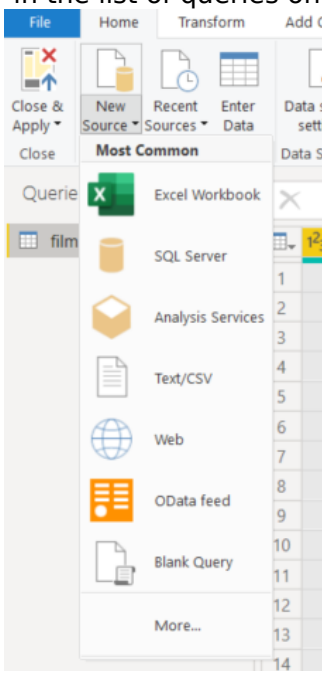


If you accept the changes when you return to the main Power BI window, they will be reflected in the query in the fields menu on the right side of the screen.

2. You can create the query from scratch, using the `Blank Query` option:



Once we are in the transform data menu, `Home > Transform data`, we must select `Home > New Source > Blank Query`. As we can see, a new query will appear in the list of queries on the left side.



To enter the query, go to the advanced editor, `View > Advanced Editor`. You will see the basis for generating a let expression in M language.

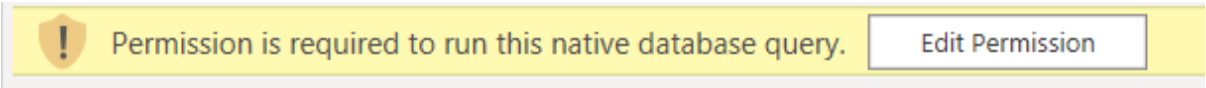
In this case we need to generate the target, which provides the context for the operation described by query and the native query.

To create the native query the following function will be used: Value.NativeQuery(target as any, query as text, optional parameters as any, optional options as nullable record) as any

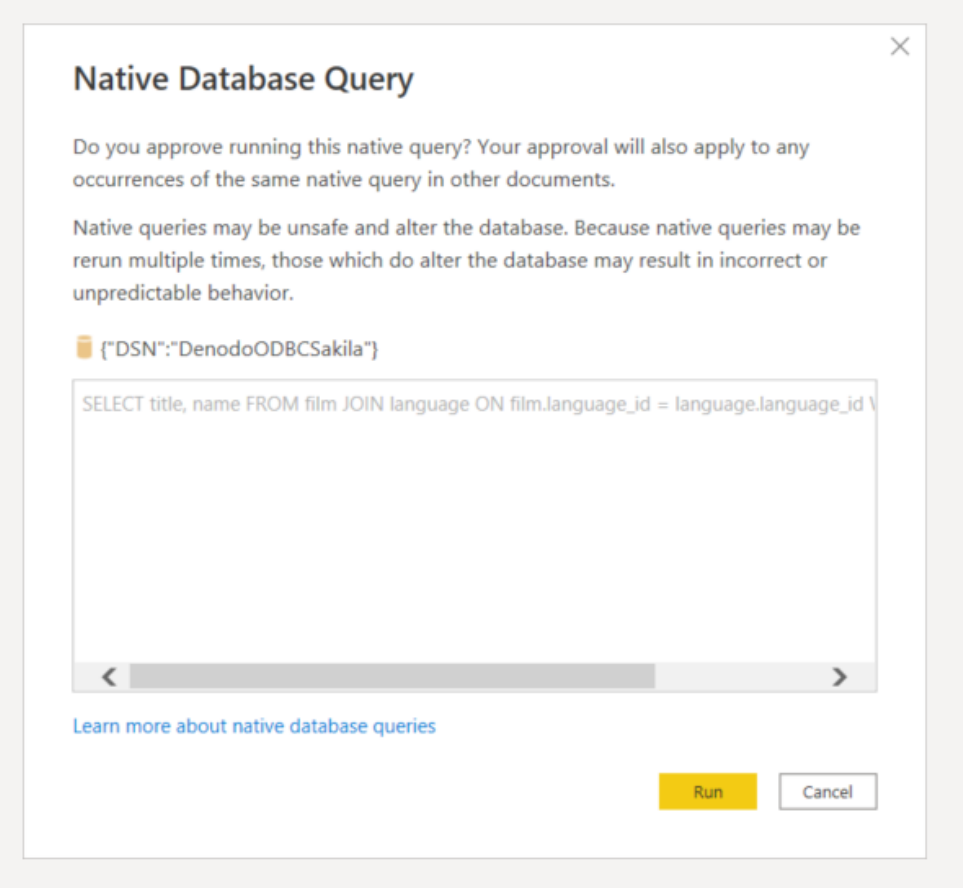
- Target: provides the context for the operation described by query
- Query: describes the query to be executed against target
- Optional parameters: may contain either a list or record as appropriate to supply the parameter values expected by query.
- Optional options: may contain options that affect the evaluation behavior of query against target.

In the options section you can define in which mode you want to retrieve the query information: import or direct query. For this purpose, the EnableFolding option is used, whose true value will activate the direct query mode. If this option has false value or does not appear in the list of options, the data will be retrieved in import mode.

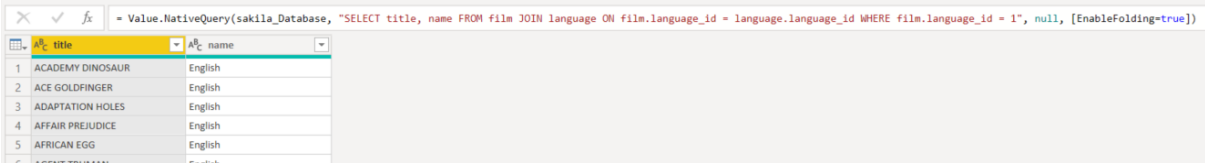
When finished, select `Done`, the advanced editor window will close and a message will appear asking for permission to execute the native query.



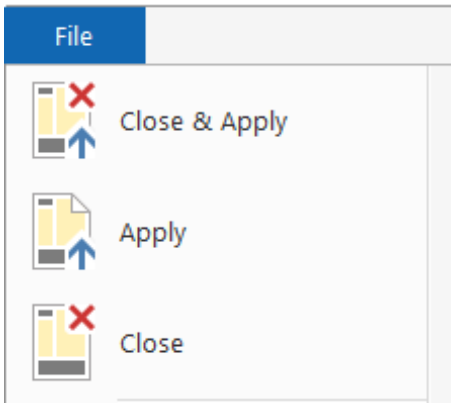
If you select `Edit Permission` you could see a window with the query to be executed and you must choose if you want to execute the query or cancel the process.



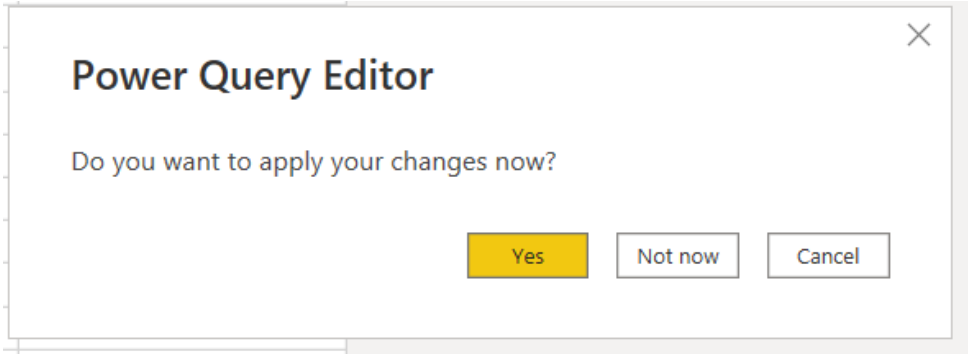
If the query has been executed, the returned data will be shown.



To apply the modifications made in the transform data window, go to the menu `File > Apply`.



If this has not been done when closing the transform data window, a dialogue box will appear asking if you want to apply the changes introduced at this moment or leave it for later.



If you accept the changes when you return to the main Power BI window, you will see the new query in the list of fields.

3.5.1.1 Mixed Informs

When creating native queries you can add the `EnableFolding` option with value `True` as shown in the previous section. In this case, the query that will be made in Direct Query mode.
If we create a query in Direct Query mode and the report we are working on is created in Import mode, we will have a mixed report where each of the queries will behave in the mode associated with it.

3.5.1.2 Use of Query field in Get Data window and Native Query in Advanced Editor.

Native Query cannot be used if the **Query** field has been used in the **Get Data** window to query. As you can see in the following image, when using the Query field, the `Value.NativeQuery` function has already been applied, returning a dataset instead of a data source object to which the `Value.NativeQuery` function can be applied.

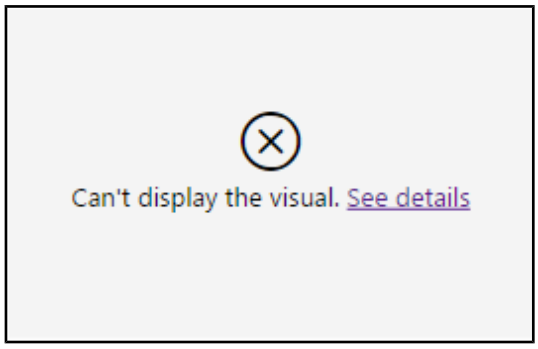
Query1

```
let
    Source = Value.NativeQuery(Denodo.Contents("DenodoODBCSakila", null, []){[Kind="Database"]}[Data],
                               "select title, name from film join language ON film.language_id = language.language_id where film.language_id = 1",
                               null, [EnableFolding=true])

in
    Source
```

4 LIMITATIONS

Most of the limitations explained in this section apply only to the **DirectQuery access mode**, and make Power BI visualizations show an error like the following:



Filtering when there is only one column in the visualization
Microsoft Power BI Desktop allows you to create a visualization with only one column. In this scenario, whether you try to apply a filter where the result should be a single row, the query will fail.

Filtering using "Top N" filter type when N = 1
When the value of N is 1 using "Top N" filter type, the query will fail.

Scenario on date data types
The Report View in Power BI uses the engine from **Microsoft Analysis Services' Tabular Models** which, at the time of this writing, does not have support for time zones, nor for `DATE` or `TIME` SQL data types. It means that when date types are loaded in Power BI, via Import or DirectQuery mode, they will be represented as Date/Time types. However, the query engine **that defines data sets for their use in reports** in Power BI is **Power Query** and this engine supports Date, Time, DateTime and DateTimeZone date types. Both engines, Microsoft Analysis Services' Tabular Models and Power Query, work in an integrated way in Power BI but due to the characteristics and limitations of each one of them, there may be operations that you will carry out with a specific one. Accordingly, consider using the Query Editor to apply filters on date data types that do not have specific support in the Microsoft Analysis Services' Tabular Models engine used by the Report View. After applying the filter you can visualize and handle the data as usual in the Report View.

Furthermore, note that ODBC does not support some SQL-92 data types, including the `TIME_WITH_TIMEZONE` and `TIMESTAMP_WITH_TIMEZONE` types. Therefore, values with date types that have time zone information will be loaded in Power BI through the Denodo ODBC driver as `TIMESTAMP` data types, without time zone.

In addition, from Denodo Platform 7.0 in versions prior to update 20190312, you will get an error in the Report View if you try to use a column with `TIMESTAMPTZ` data type in a report because it is processed as a TIMESTAMP, due to the limitation of the ODBC driver explained above, and the fact that Virtual DataPort in versions previous to the mentioned update did not allow an implicit cast between `TIMESTAMP/TIMESPTAMPTZ` data types. Nevertheless, bearing in mind that Microsoft Analysis Services' Tabular Models does not support DateTimeZone types either, you can use the Power Query Editor to apply transformations in your data in order to get its information in types understandable by Microsoft Analysis Services' Tabular Models. For instance, you can transform your DateTimeZone column into several columns where each of them will contain a component of the date: year, month, day, hour, minute and second.

From Denodo Platform 7.0 update 20190312 onwards implicit casts between types `LOCALDATE/TIME/TIMESTAMP` and `TIMESTAMPTZ` are supported. This means that if you try to use a column with `TIMESTAMPTZ` data type in a report, the error is no longer thrown. However, keeping in mind the ODBC limitation and the date handling by the two Power BI engines explained above, columns with time zone information in Denodo are always represented as `TIMESTAMP` types in Power BI and this may cause unexpected effects due to the need to perform implicit cast when dealing with this type of columns.

Loading TEXT types from Microsoft SQL Server

`TEXT` types from Microsoft SQL Server allow a very limited number of operators, they cannot be compared or sorted, except when using the `IS NULL` or `LIKE` operator. Moreover, Power BI Desktop, through the custom connector, does not allow changing the searchable property with the aim of trying to take into account this limitation. Therefore, this scenario causes an error when you try to add a value of this type in a visualization.

Loading data when a field in a view has more than 42 relations with other views.

In the case of a field belonging to a view that has more than 42 relations with other views, Power BI will generate an error that can be seen when accessing the Transform Data menu. The message is as follows:

```
Preview.Error: The type of the current preview value is too complex to display.
```

In order to process the data and solve this error, the view that generates the problem must be selected in the Transform Data screen, in the list of queries on the left. After selecting it, access the Advanced Editor and edit the M expression adding in the variable options of Denodo.Contents the following option: CreateNavigationProperties = false.

This property, with its value set to false, establishes that the navigation properties in the returned tables will not be generated, so Power BI will not try to generate the more than 42 relations between the field of the view that fails and the other views, avoiding the error.

Two examples of the use of this property are shown below:

1. In a standard expression to retrieve data from a view:

```
let
    Source = Denodo.Contents("DenodoODBC", null, [CreateNavigationProperties=false]),
    admin_Database = Source{[Name="admin",Kind="Database"]}[Data],
    powerbiMoreThan42Associations_Schema = admin_Database{[Name="powerbiMoreThan42Associations",Kind="Schema"]}[Data],
    view_View = powerbiMoreThan42Associations_Schema{[Name="view",Kind="View"]}[Data]
in
    view_View
```

2. Expression that uses the Options variable of Denodo.Contents to retrieve the data of a view:

```
let
    Source = Denodo.Contents("DenodoODBC", null, [DenodoDatabase="admin",DenodoView="view",CreateNavigationProperties=false])
in
    Source
```