BI360 Suite and Power BI

A Solver White Paper

Published: November 22, 2017

Introduction

The *Power BI* suite of data analysis tools from Microsoft offers exciting new ways to present and share your data on-line in an interactive and intuitive format that includes a variety of tables, charts and geographic visualizations.

This whitepaper is intended to provide background information and methods that can be used to bring data from a BI360 Data Warehouse (*BI360DW*) into a *Power BI* model for use in visualizations, reports and dashboards. We assume a basic familiarity with *Power BI*, though readers looking for a more general overview of *Power BI* features and functionality are encourage to visit the online resources listed below.

Topic	Link
Product overview	https://powerbi.microsoft.com/en-us/
Overview of *Power BI* gateways, which the user can use to connect to and refresh data from on-premises data sources including *BI360DW*	https://powerbi.microsoft.com/en-us/gateway/
Getting started tutorial using *Power BI* Desktop	https://powerbi.microsoft.com/en-us/documentation/powerbi-desktop-getting-started/

In order to meet diverse needs of the BI360 user community, this white paper has been divided into two sections:

- **The BI360 Data model**: This section provides a review of core data warehousing concepts and explains the *BI360DW* table structure in this context. This section should be helpful to readers not already familiar with how data is stored in *BI360DW*.

- **Examples using *BI360DW* with Power BI Desktop and Web Applications**: This section discusses the steps required to use *Power BI* with BI360 and covers several key issues to consider when using *BI360DW* data with *Power BI*. An example *Power BI* modeling exercise using the BI360 Corporate Demo database is provided.

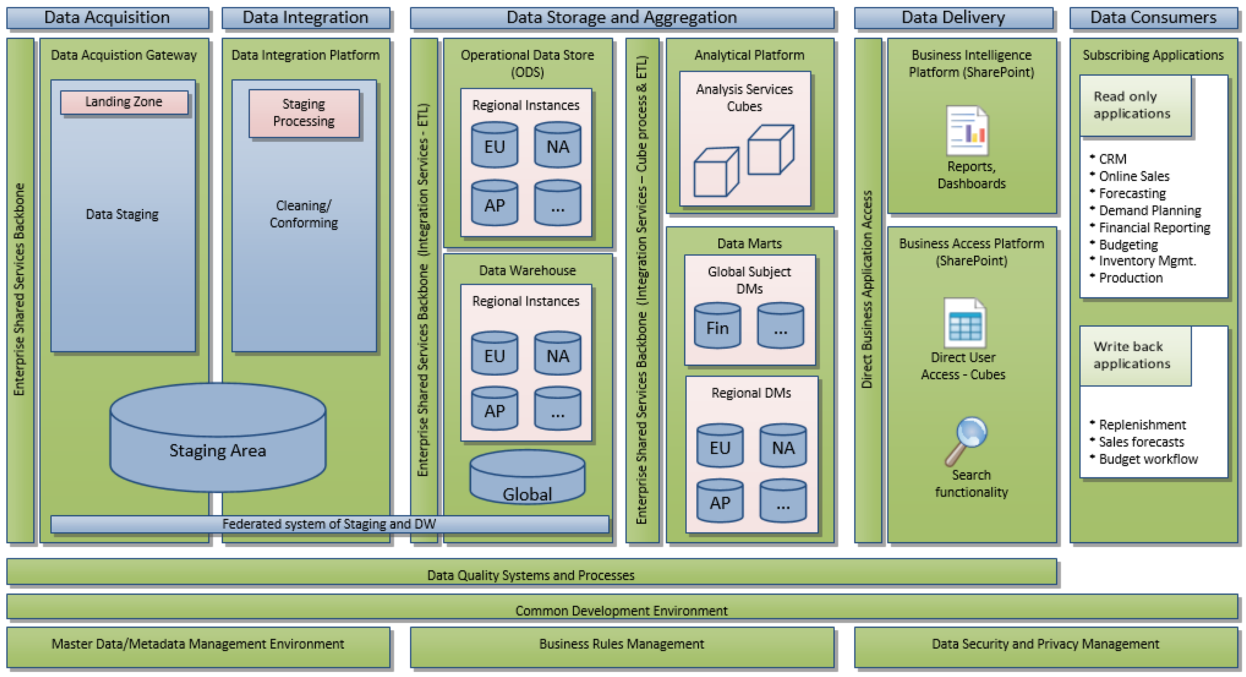
The BI360 Data Model

Introduction to Data Warehousing

Data warehouses can take many forms, and the specific business needs of the organization implementing the data warehouse will guide the choice of structure. In fact, the term "data warehouse" can be applied to a number of different database types that bear little resemblance to each other, from systems that are designed to accurately model extremely complex relationships to systems that are designed for optimized reporting performance.

At one end of the spectrum are the traditional "enterprise data warehouse" projects that people may be familiar with. These are often multi-year, multi-million dollar implementations that collect all of the organization's data into a central system for data storage, management, analysis, and reporting.

Below is an example technical architecture for a traditional Data warehouse ecosystem. The inherent complexity and multiple layers provide a tremendous flexibility to the system, albeit one that can come at a tremendous cost.



Data Acquisition
Data Integration
Data Storage and Aggregation
Data Delivery
Data Consumers
Enterprise Shared Services Backbone
Data Acquistion Gateway
Landing Zone
Data Staging
Staging Area
Data Integration Platform
Staging Processing
Cleaning/ Conforming
Enterprise Shared Services Backbone (Integration Services - ETL)
Operational Data Store (ODS)
Regional Instances
EU
NA
AP
...
Data Warehouse
Regional Instances
EU
NA
AP
...
Global
Federated system of Staging and DW
Enterprise Shared Services Backbone (Integration Services – Cube process & ETL)
Analytical Platform
Analysis Services Cubes
Data Marts
Global Subject DMs
Fin
...
Regional DMs
EU
NA
AP
...
Direct Business Application Access
Business Intelligence Platform (SharePoint)
Reports, Dashboards
Business Access Platform (SharePoint)
Direct User Access - Cubes
Search functionality
Subscribing Applications
Read only applications
CRM
Online Sales
Forecasting
Demand Planning
Financial Reporting
Budgeting
Inventory Mgmt.
Production
Write back applications
Replenishment
Sales forecasts
Budget workflow
Data Quality Systems and Processes
Common Development Environment
Master Data/Metadata Management Environment
Business Rules Management
Data Security and Privacy Management

Figure 1: Traditional Data Warehouse Architecture

The above diagram represents a view of a complete Business Intelligence solution, from Data Acquisition to Data Consumption. The concept of a data warehouse lives primarily in the central area, the Data Storage and Aggregation layer of the BI Stack.

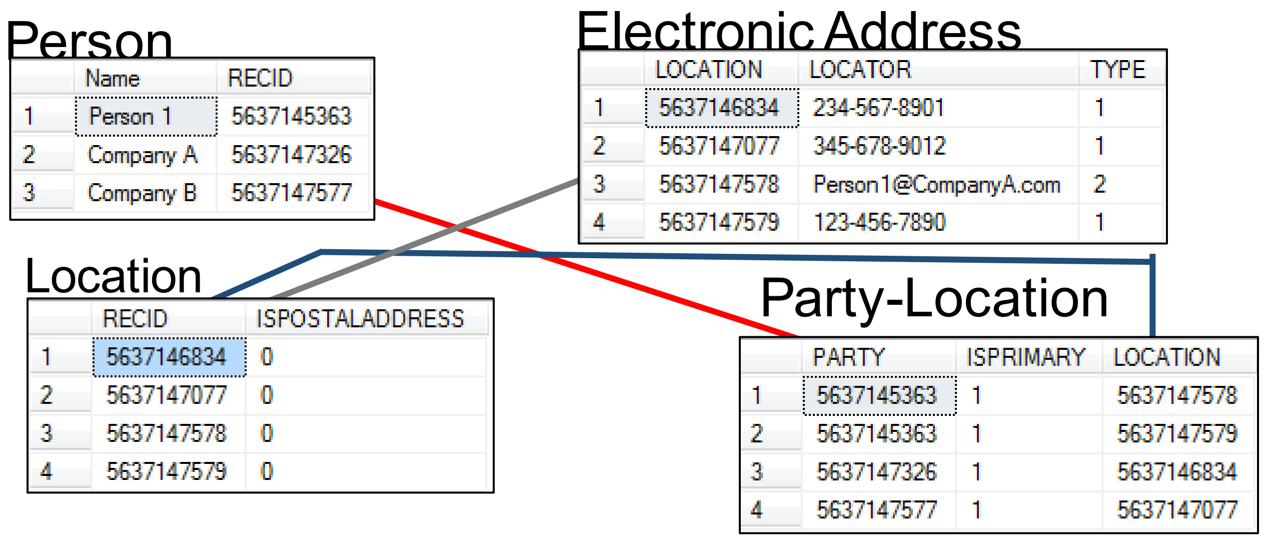
Within the Data Storage and Aggregation layer, there are several common types of databases that all can be thought of as different "flavors" of a data warehouse.

1. Operational Data Store (ODS): Operation data stores are designed to hold shorter term, operational data. The use case tends to be more towards tactical reporting in real-time or near-real-time on data aggregated from one or more operational systems.
2. Data Mart (DM): Data Marts are designed to synthesize data into structures that lend themselves to efficient access of stored information and clear, understandable information relationships. They typically have structures that are easier to understand, and are simpler to report against. Data marts are commonly business area or regionally specific.
3. Enterprise Data Warehouse (EDW): The term "Enterprise Data Warehouse" is typically applied to systems where a more general approach to data storage and management is taken. They often reside at the center of the BI stack, as the destination that all source systems feed into, and the source from which reporting systems and data consumers draw their data. They tend to be the most complex of the three general types, in that they can model any type of information relationship if correctly designed.

Each of the three common types of "data warehouses" have specific strengths and weaknesses which suit them to their designed tasks. One of the main distinguishing characteristics of each is the degree of normalization present in each type. Normalization in the context of database design refers to the level of specificity or redundancy of data within a particular table. Normalization can be a challenging concept to grasp at first, but simply put, it can be thought of as the degree of "flattening" of the table structure required to hold information about an object.

As an example, take the idea of a "customer". A customer, may have many distinguishing characteristics that typical Enterprise Resource Planning (ERP) systems need to track, such as Name, Phone number, email, et cetera.

In a normalized database, each of these pieces of information would occupy a separate table, with records of the same type. There would be an "Email Address" table, a "Person" table, a "phone number" table amongst others. Take this example from Microsoft Dynamics AX 2012:



Person

	Name	RECID
1	Person 1	5637145363
2	Company A	5637147326
3	Company B	5637147577

Electronic Address

	LOCATION	LOCATOR	TYPE
1	5637146834	234-567-8901	1
2	5637147077	345-678-9012	1
3	5637147578	Person1@CompanyA.com	2
4	5637147579	123-456-7890	1

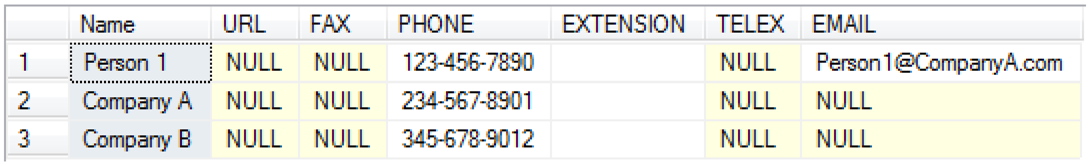
Location

	RECID	ISPOSTALADDRESS
1	5637146834	0
2	5637147077	0
3	5637147578	0
4	5637147579	0

Party-Location

	PARTY	ISPRIMARY	LOCATION
1	5637145363	1	5637147578
2	5637145363	1	5637147579
3	5637147326	1	5637146834
4	5637147577	1	5637147077

In order to pull together a customer's address from these tables there is a lot of logic that has to be used in the joining of the tables to produce a simple customer record set, such has below:



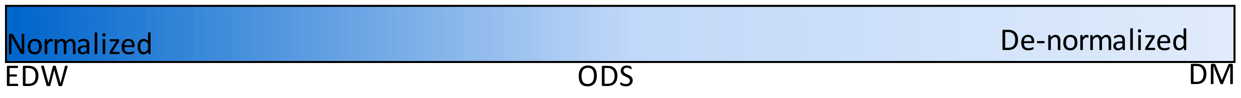
	Name	URL	FAX	PHONE	EXTENSION	TELEX	EMAIL
1	Person 1	NULL	NULL	123-456-7890		NULL	Person1@CompanyA.com
2	Company A	NULL	NULL	234-567-8901		NULL	NULL
3	Company B	NULL	NULL	345-678-9012		NULL	NULL

And this is a *simple* example.

This is an example of a normalized structure (there are many types of normalization, which we won't go into here), and would be representative of the kind of table structure found in a typical ERP system or an EDW.

At the opposite end of the spectrum is the de-normalized structure or "dimensionalized" structure. This is the "flattened" structure we mentioned above, and depicted in the simple customer record set above. All of the related pieces of information that are specific to a customer in this case, are contained on the same table.

So, where do our three database types fall on this continuum? It looks something like this:



The ODS, can float nearly anywhere on this line, but as a general rule, the EDWs and DMs keep mostly to opposite ends. This really stems from their general use cases, as illustrated above: EDWs tend to have a more complex structure and can more accurately and compactly represent complex data structures, whereas DMs are easier to understand structurally and are optimized for rapid reporting access.

These structures, have a simple pictorial mnemonic used to describe their schemas: The Star Schema and the Snowflake Schema, which we will describe below.

Records & Transactions vs. Facts & Dimensions

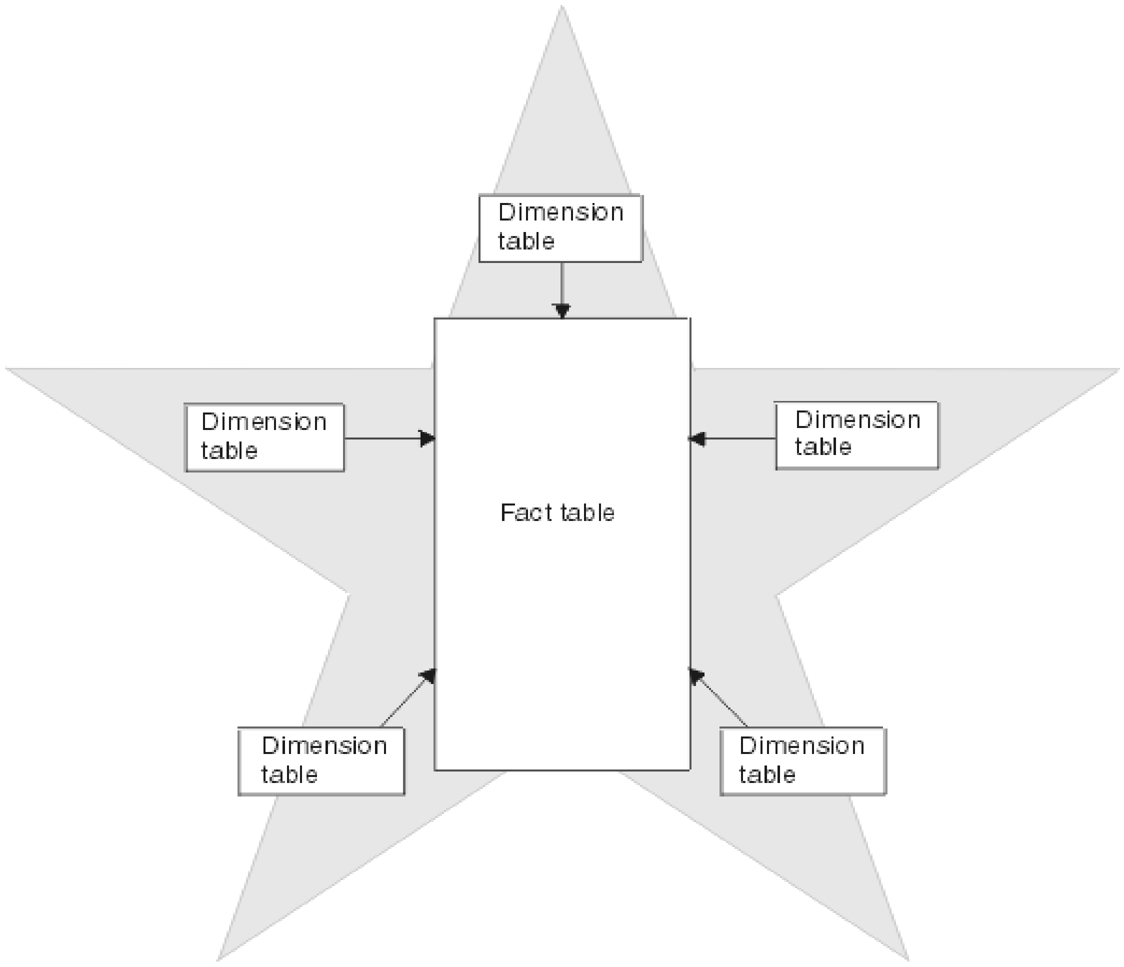
Stepping back for a minute from the general description of data warehouses, let's cover a bit of nomenclature related to the actual records in the different types of databases.

In a typical ERP database, some tables collect many thousands or millions of records of a particular type, such as a Ledger entry in a General Ledger table or an Inventory movement record in an Inventory table. Each of these records refer to a specific transaction type that has certain properties.

Take for example, the example of a customer buying a product from a company. The Sales record may have some relationship to an Order record and some other relationship to an Inventory record. Each of these separate objects, the Sale, the Order, and the Inventory reside in separate transaction tables. They are stored separately from each other, and each represents a part of a complete order fulfillment process to a customer. Together, these records can describe aspects of a Fact. The Fact, in this case, is that a customer bought some product from a company. The "Fact" is, in essence, made up of a number of separate transactions that all occurred in the company's business process during the sale.

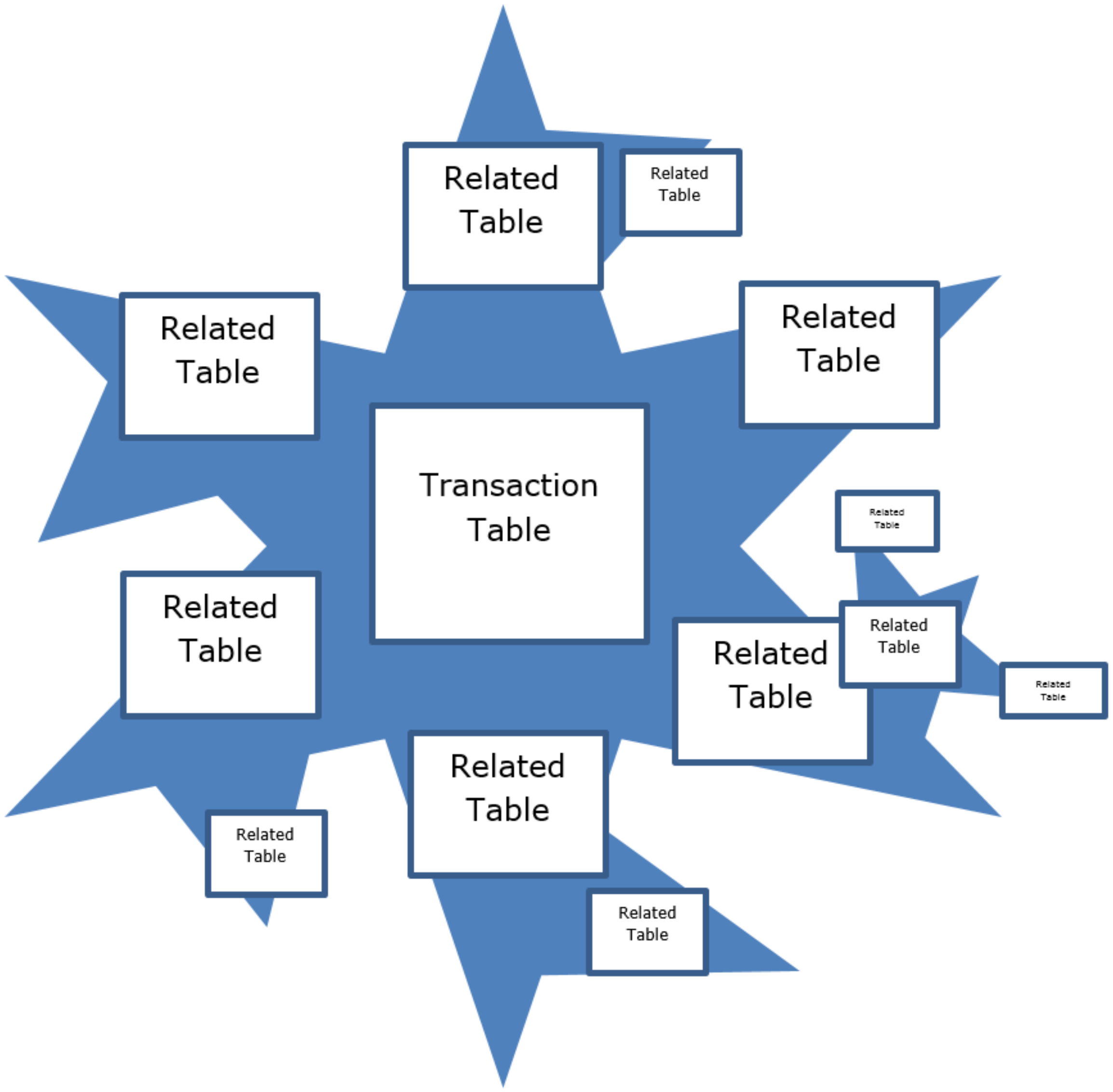
Strictly speaking, in the context of data warehouse design, a fact is a value or measurement that represents a meaningful data point regarding a specific activity within the organization. Facts can include individual transactions such as sales or payments, or aggregated data such as account balances or inventory stock levels. Fact tables are generally kept as narrow as possible. In other words, rather than loading fact tables with every possible detail pertaining to a transaction (such as the buyer's full name and address in a sales transaction), they contain only reference values and key fields that provide links to other tables where other pertinent details can be found. These other linked tables are called dimension tables. Customers, Sales Person, Regions, Products and Fiscal Periods are a few examples of data elements that might be stored in dimension tables. Linking each dimension table to the fact table is a *primary key* that identifies each member of the dimension table as unique.

The arrangement of data into fact tables and dimension tables is called a "Star Schema", where the fact table occupies the center and the dimensions represent the arms of the star, as shown here:



This is an example of the de-normalized structure mentioned earlier.

The other schema type we have discussed, the Snowflake Schema, can have a complex branching structure like that of a snowflake:



As the user can see, this can get complicated! The BI360 Data Warehouse most closely follows the Star Schema model and in the next section, we will go into more practical detail of BI360 warehouse design.

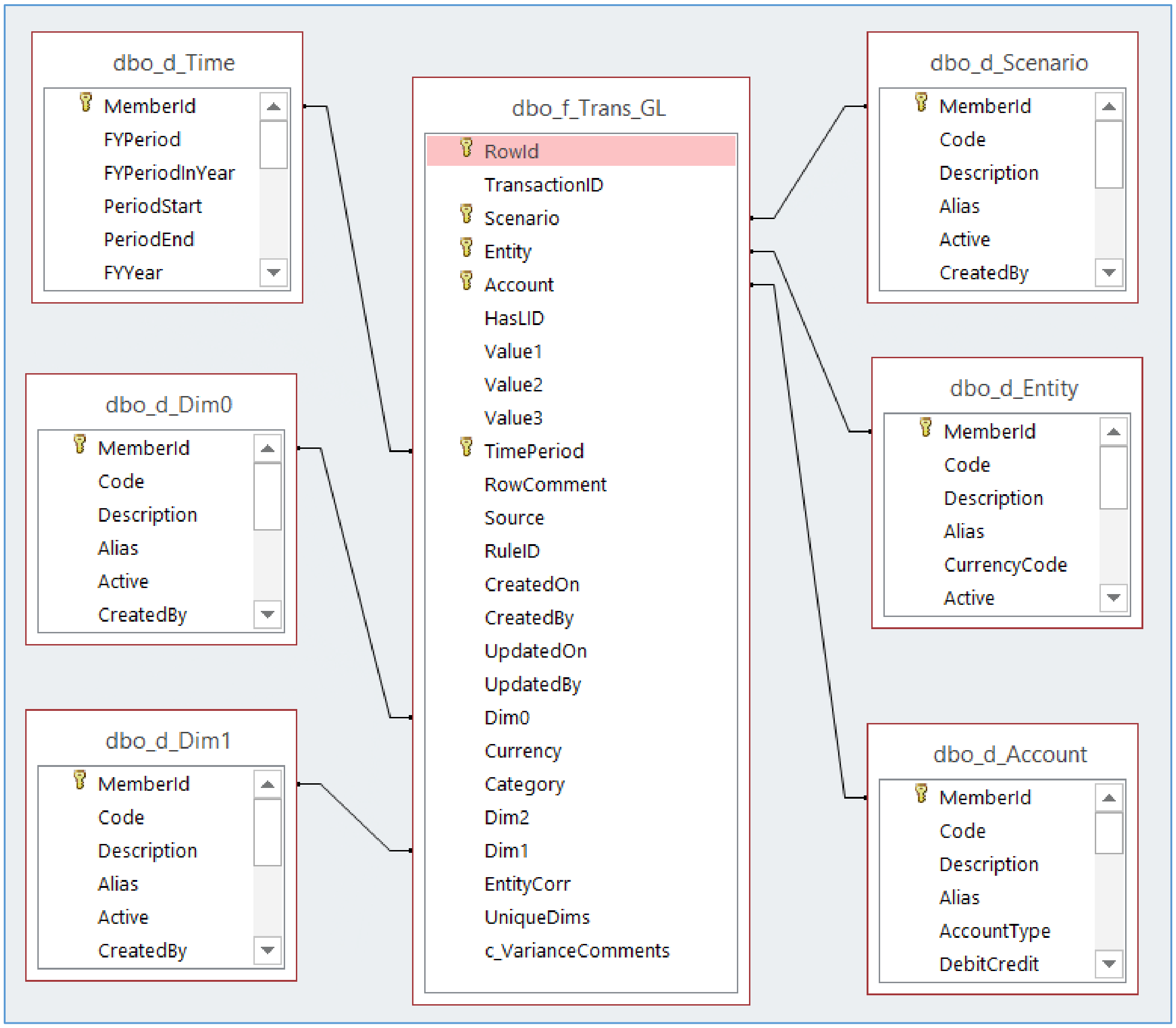
Fact and Dimension tables in the BI360 Data Warehouse

The user may be used to thinking of the BI360 Data Warehouse as a collection of subject matter specific modules such as General Ledger, AR, AP, Revenues, Fixed Assets, Capital, etc. However, each module is also a star schema consisting of one fact table and several dimension tables that are linked to that fact table. When viewing the tables directly in SQL Server (such as through SQL Server Management Studio), telling the fact and dimension tables apart is relatively straightforward:

- Fact table names begin with **f_Trans** followed by the system ID of the module. For example:
    - f_Trans_GL (General Ledger)
    - f_Trans_CP (Capital)
    - f_Trans_HR (Payroll)
- Dimension table names begin with **d_** followed the system ID of the dimension. For example:
    - d_Entity
    - d_Account
    - d_Employee
    - d_Salesperson
    - d_Dim0 (user defined dimension 0)
    - d_Dim1 (user defined dimension 1)

BI360DW also allows dimension tables to be shared across multiple fact tables. Thus the d_Account table can be linked to more than one module or fact table.

Below is a graphical representation of fact and dimension tables, as might be seen in a typical setup of the *BI360DW* General Ledger (GL) module:



Notice how in the above diagram only the ID (or Member ID) field for each dimension is referenced in the fact table. This structure minimizes redundancy in the fact table while creating ample possibilities for summarizing or slicing the data using the additional fields that are available in dimension tables. These additional fields, which provide information about each dimension ID, are called attributes.

The MemberID value in each dimension record is an integer that acts as a unique identifier or key field that is linked to one or more fact tables. For example, MemberID in the d_Account table is linked to the Account field in the f_Trans_GL table. Also notice that dimension fields in the fact table have same name as the dimension table to which they are linked.

The below table lists the more commonly used dimension tables in the *BI360DW*:

Dim. Table	Fact Table Equivalent	Description
d_Account	Account	Financial accounts
d_Asset	Asset	Fixed Assets
d_Category	Category	Transaction Categories
d_Customer	Customer	Customers
d_Dim0	Dim0	User defined dimension
d_Dim1	Dim1	User defined dimension
d_Dim2	Dim2	User defined dimension
d_Dim3	Dim3	User defined dimension
d_Dim4	Dim4	User defined dimension
d_Dim5	Dim5	User defined dimension
d_Employee	Employee	Employees
d_Entity	Entity	Business Entities/Companies
d_Item	Item	Inventory Items
d_Product	Product	Products
d_Project	Project	Projects
d_SalesPerson	SalesPerson	Sales Persons
d_Scenario	Scenario	Scenarios (such as Actual, Budget, Forecast, etc.)
d_Time	TimePeriod	Time Periods (in BI360, the lowest grain is day)
d_Vendor	Vendor	Vendor

Below are the examples of commonly used fact tables. Each fact table is the equivalent of one module in *BI360DW*.

Fact Table	Module	Description
f_Trans_AP	AP	Accounts Payable
f_Trans_AR	AR	Accounts Receivable
f_Trans_CP	CP	Capital
f_Trans_GL	GL	General Ledger
f_Trans_HR	HR	Payroll
f_Trans_OT	OT	Other - user defined
f_Trans_OT2	OT2	Other - user defined
f_Trans_OT3	OT3	Other - user defined
f_Trans_PJ	PJ	Projects
f_Trans_RV	RV	Revenues

Note: Additional user defined (OT) modules can be added in Data Warehouse Manager.

Listing of Fact Table Fields

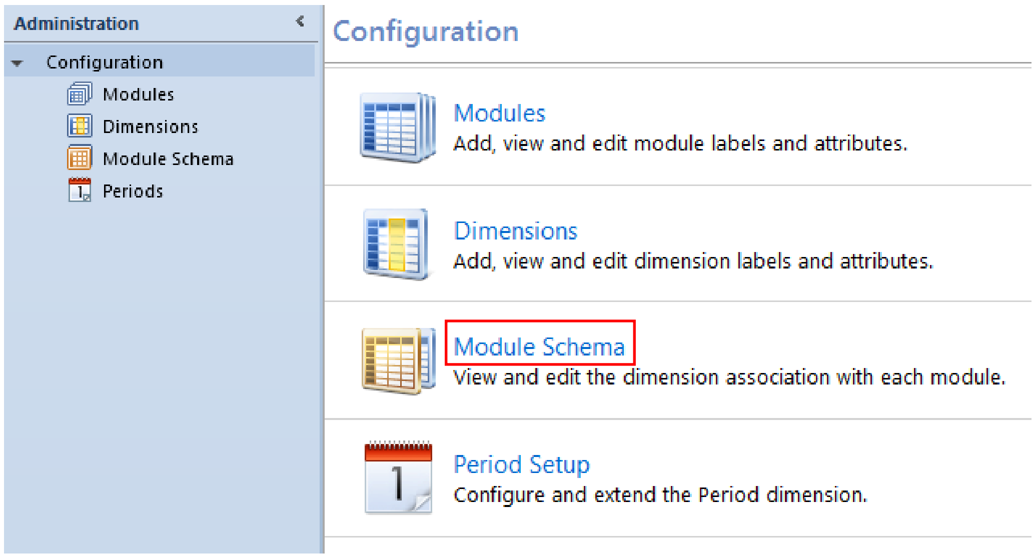
The below chart lists the fields used in the GL fact table. Fields that are required in all modules are shaded in yellow. Keep in mind that the dimension fields may vary according to the default configuration of the module and any additional changes made in the Data Warehouse Administration tool.

Column Name	Is Dimension (Y/N)	Data Type	Character Max	Description	Required
RowId	No	bigint	n/a	System generated row identifier. Not used as a key field.	Yes (system generated)
TransactionID	No	nvarchar	50	Unique identifier -- part of the composite key and can be used to distinguish records that have the same values in other key fields (e.g. dimensions). For example, we may use this to distinguish transactions that occurred on the same day and for the same Entity, Department, Account, etc.	No
RowComment	No	nvarchar	512	Record description	No
Scenario	Yes	bigint	n/a	Member ID for Scenario Dimension	Yes
TimePeriod	Yes	bigint	n/a	Member ID for Time Period Dimension Note: this column holds a date value formatted as an 8 digit integer comprised of a 4 digit year, 2 digit month and 2 digit day. For example “1/1/2015” would be represented as 20150101.	Yes
Category	No	bigint	n/a	Member ID for Category Dimension	No
Entity	Yes	bigint	n/a	Member ID for Entity Dimension	Yes
Currency	Yes	bigint	n/a	Member ID for Currency Dimension	No
Account	Yes	bigint	n/a	Member ID for Account Dimension -- this is a required field in the GL module; optional otherwise	Yes
Dim0 through DimX	Yes	bigint	n/a	User Defined Dimensions, added in Data Warehouse manager Note: These have to be enabled in Data Warehouse Manager before they appear within the module/fact table.	No
Value1	No	numeric	n/a	User Defined Numeric value	No
Value2	No	numeric	n/a	User Defined Numeric value	No
Value3	No	numeric	n/a	User Defined Numeric value	No

Column Name	Is Dimension (Y/N)	Data Type	Character Max	Description	Required
Source	No	nvarchar	50	Can be used to indicate source server and/or database	No
RuleID	No	bigint	n/a	Used when updated by BI360 'business rules' feature, otherwise can ignore	No
CreatedOn	No	datetime	n/a	Date/time the record was created	No
CreatedBy	No	nvarchar	50	User/process that created the record	No
UpdatedOn	No	datetime	n/a	Date/time the record was updated	No
UpdatedBy	No	nvarchar	50	User/process that updated the record	No
UniqueDims	No	varbinary	20	System generated key value, consists of combination of each dimension value	No (System Generated)

Using the BI360*DW* Manager to review fact table structure

The Administration menu in the BI360 *Data Warehouse Manager* (*DWM*) has a “Module Schema” that provides a useful chart to help the user visualize the links between fact and dimension tables:



Configuration

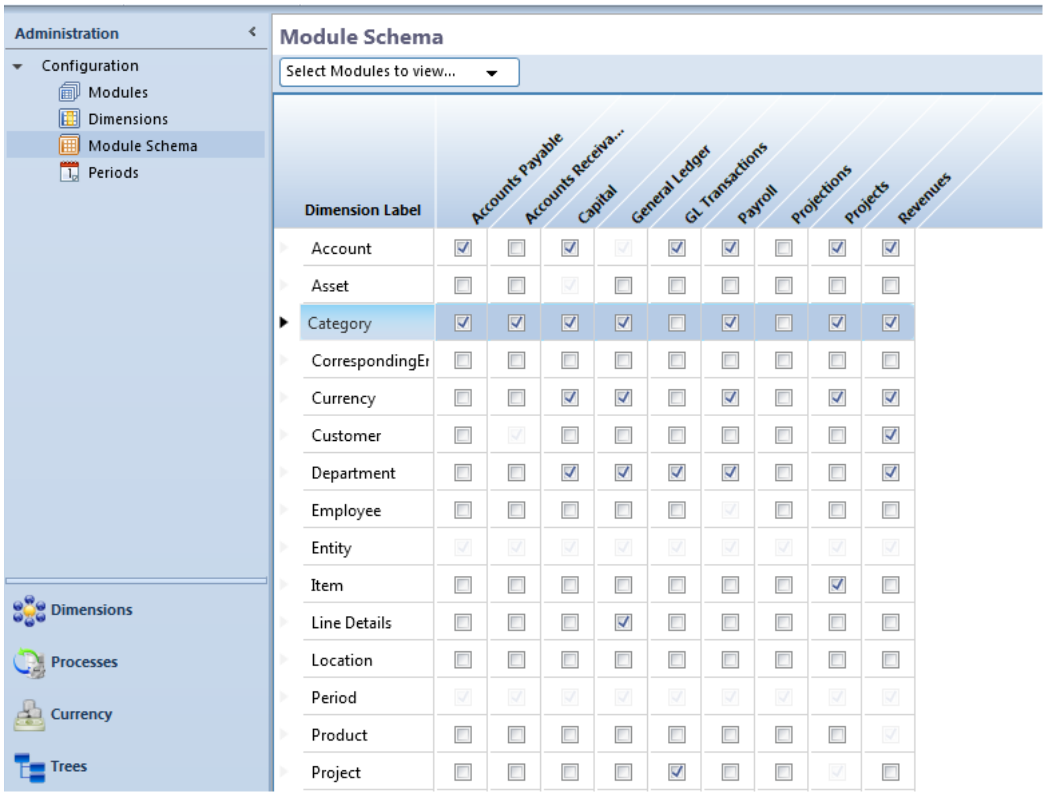
Modules
Add, view and edit module labels and attributes.

Dimensions
Add, view and edit dimension labels and attributes.

Module Schema
View and edit the dimension association with each module.

Period Setup
Configure and extend the Period dimension.

Scrolling down this screen, the user can see the dimensions that are used in each module as indicated by a check box. Modules are listed horizontally across the top of the screen.



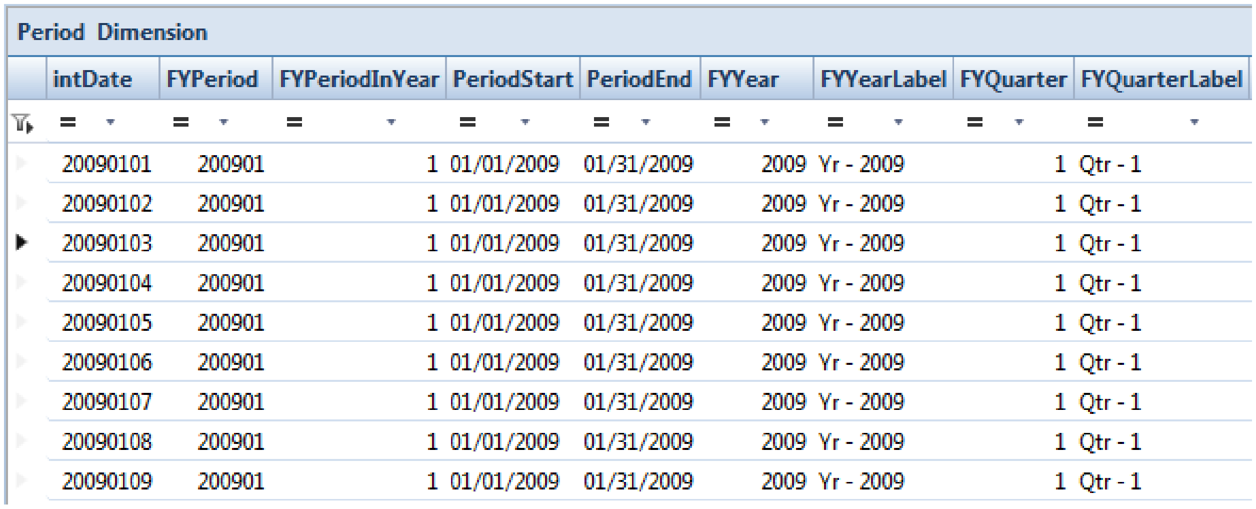
Module Schema

Dimension Label	Accounts Payable	Accounts Receiva...	Capital	General Ledger	GL Transactions	Payroll	Projections	Projects	Revenues
Account	☑	☐	☑	✓	☑	☑	☐	☑	☑
Asset	☐	☐	✓	☐	☐	☐	☐	☐	☐
Category	☑	☑	☑	☑	☐	☑	☐	☑	☑
CorrespondingEr	☐	☐	☐	☐	☐	☐	☐	☐	☐
Currency	☐	☐	☑	☑	☐	☑	☐	☑	☑
Customer	☐	✓	☐	☐	☐	☐	☐	☐	☑
Department	☐	☐	☑	☑	☑	☑	☐	☐	☑
Employee	☐	☐	☐	☐	☐	✓	☐	☐	☐
Entity	✓	✓	✓	✓	✓	✓	✓	✓	✓
Item	☐	☐	☐	☐	☐	☐	☐	☑	☐
Line Details	☐	☐	☐	☑	☐	☐	☐	☐	☐
Location	☐	☐	☐	☐	☐	☐	☐	☐	☐
Period	✓	✓	✓	✓	✓	✓	✓	✓	✓
Product	☐	☐	☐	☐	☐	☐	☐	☐	✓
Project	☐	☐	☐	☐	☑	☐	☐	✓	☐

The Period Dimension

A core requirement of almost any data warehouse implementation is the capability to support time-series based reporting such as annual trends, year over year comparisons, etc.

The Period (d_Time) dimension table contains a hierarchy that facilitates these types of analyses and can be used to summarize fact data by month, quarter and year:



Period Dimension

intDate	FYPeriod	FYPeriodInYear	PeriodStart	PeriodEnd	FYYear	FYYearLabel	FYQuarter	FYQuarterLabel
20090101	200901	1	01/01/2009	01/31/2009	2009	Yr - 2009	1	Qtr - 1
20090102	200901	1	01/01/2009	01/31/2009	2009	Yr - 2009	1	Qtr - 1
20090103	200901	1	01/01/2009	01/31/2009	2009	Yr - 2009	1	Qtr - 1
20090104	200901	1	01/01/2009	01/31/2009	2009	Yr - 2009	1	Qtr - 1
20090105	200901	1	01/01/2009	01/31/2009	2009	Yr - 2009	1	Qtr - 1
20090106	200901	1	01/01/2009	01/31/2009	2009	Yr - 2009	1	Qtr - 1
20090107	200901	1	01/01/2009	01/31/2009	2009	Yr - 2009	1	Qtr - 1
20090108	200901	1	01/01/2009	01/31/2009	2009	Yr - 2009	1	Qtr - 1
20090109	200901	1	01/01/2009	01/31/2009	2009	Yr - 2009	1	Qtr - 1

Using BI360 Data Warehouse with Power BI

In this section, a couple of examples using Solver's Corporate Demo database will be demonstrated, which can be found in the Downloads section on Solver's technical support website, https://support.solverusa.com

These examples will cover two scenarios for sourcing data from the *BI360DW* in the *Power BI* desktop application:

1. Selecting tables directly from the database schema and making the necessary formatting changes in *Power BI*
2. Using a query to retrieve data from the *BI360DW* already structured in the desired format.

There are pros and cons to each approach. Linking directly to the source tables gives the user maximum flexibility but will require hiding some extraneous columns and relabeling others to make them more recognizable to the end user. Using a T-SQL query requires a bit more knowledge of the underlying database schema, but can give the user exactly what the user needs, in the right format, for your visualizations. The best approach for your application will depend largely on your goals and your comfort level with writing T-SQL queries.

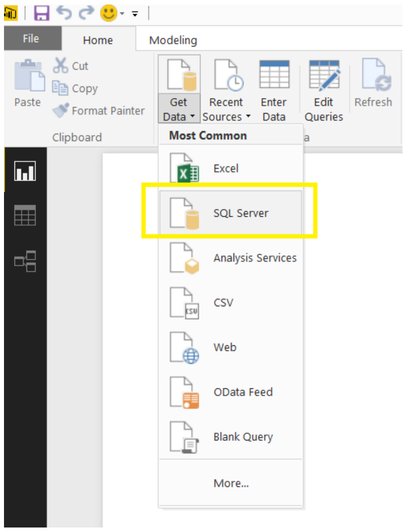
Overview: Building a Power BI Model

Both of our examples for creating and populating a new model in *Power BI* follow the same essentials steps:

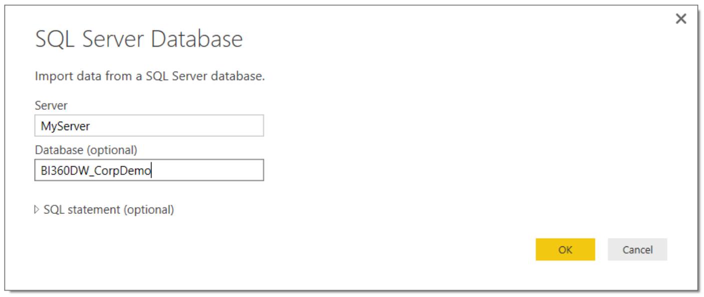
1. If the users has not already done so, sign up for the *Power BI* online service at https://powerbi.microsoft.com.
2. From the *Power BI* service website download, install and then launch the *Power BI* desktop app.
3. In *Power BI* Desktop, connect to a data source (Home ribbon, then Get Data)
4. Shape and Combine Data: use the query editor and data view screens to select source tables, create/modify table joins, etc.
5. Build Reports: go to the report view to add fields from your data set to visualizations
6. Publish the contents of your power bi desktop file to the cloud based *Power BI* service
7. Install and/or configure a personal or enterprise Gateway to manage on-demand or automated updates.

Example 1: Creating a Dataset Based on Database Tables

After launching the *Power BI* desktop app and creating a new *Power BI* desktop file, navigate to **Home > Get Data > SQL Server**.



Enter the server and database name when prompted.



SQL Server Database

Import data from a SQL Server database.

Server

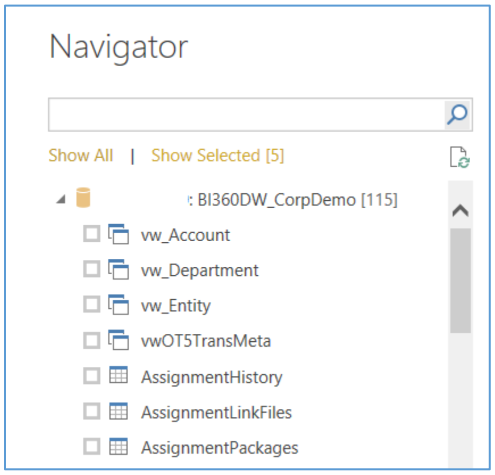
MyServer

Database (optional)

BI360DW_CorpDemo

SQL statement (optional)

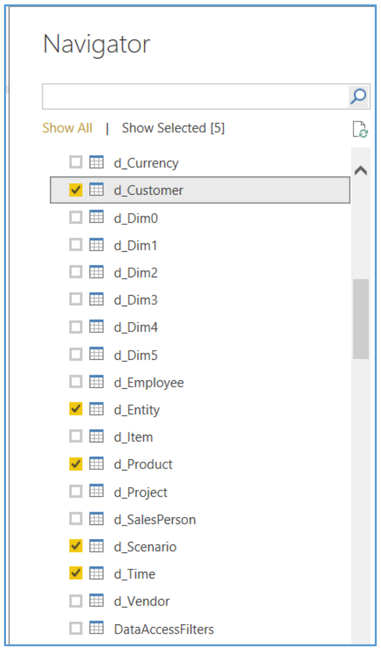
This will bring up the Navigator window, where users can select which tables to include by checking the box next to each one.



Navigator

: BI360DW_CorpDemo [115]
- vw_Account
- vw_Department
- vw_Entity
- vwOT5TransMeta
- AssignmentHistory
- AssignmentLinkFiles
- AssignmentPackages

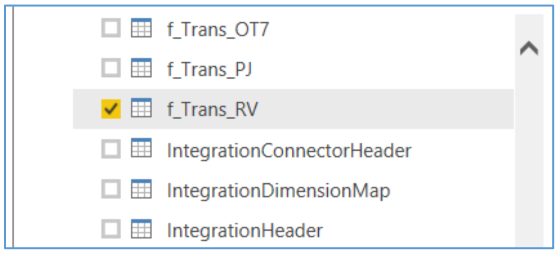
For the purposes of this example, we’re including the following dimension tables: Customer, Entity, Product, Scenario and Time.


Navigator

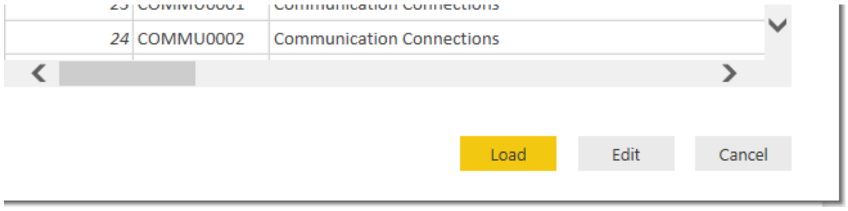
Show All | Show Selected [5]

- [ ] d_Currency
- [x] d_Customer
- [ ] d_Dim0
- [ ] d_Dim1
- [ ] d_Dim2
- [ ] d_Dim3
- [ ] d_Dim4
- [ ] d_Dim5
- [ ] d_Employee
- [x] d_Entity
- [ ] d_Item
- [x] d_Product
- [ ] d_Project
- [ ] d_SalesPerson
- [x] d_Scenario
- [x] d_Time
- [ ] d_Vendor
- [ ] DataAccessFilters

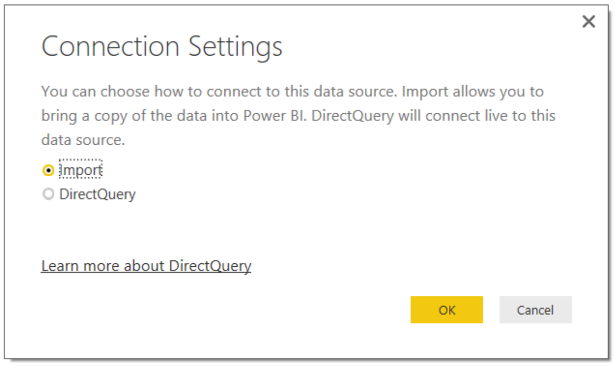
We’ll also include the **f_Trans_RV** fact table, which contains actual and forecast sales by product and customer.


- [ ] f_Trans_OT7
- [ ] f_Trans_PJ
- [x] f_Trans_RV
- [ ] IntegrationConnectorHeader
- [ ] IntegrationDimensionMap
- [ ] IntegrationHeader

We’ll then click the **Load** button to confirm these selections and import the data into *Power BI*.


24	COMMU0002	Communication Connections

Note that the user will also be given the option to either import the data into your *Power BI* model or use DirectQuery, which will connect live to the source.


Connection Settings

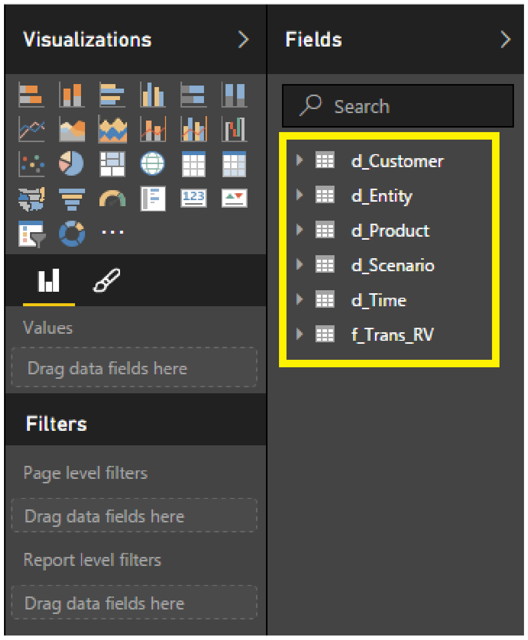
You can choose how to connect to this data source. Import allows you to bring a copy of the data into Power BI. DirectQuery will connect live to this data source.

[✓ Import] DirectQuery

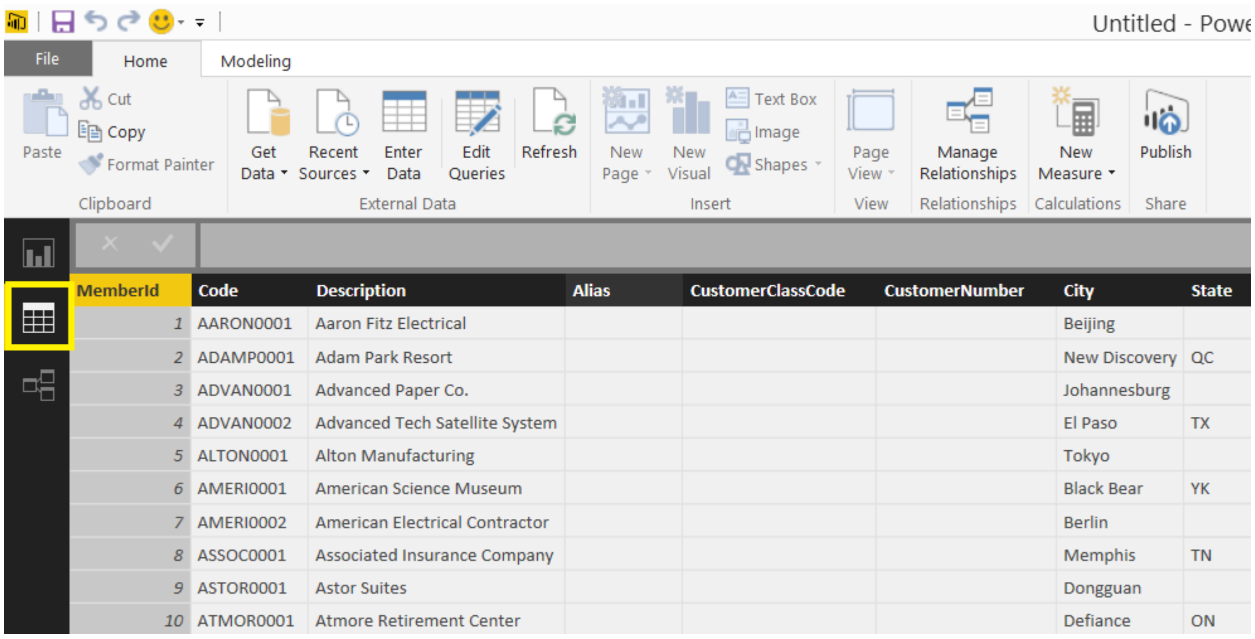
Learn more about DirectQuery

This choice is a matter of preference and depends on a number of considerations, such as storage space and network latency.

After the data is loaded, we're brought back to the Report View in *Power BI* desktop. On the right side of the page we can see the tables that were selected.



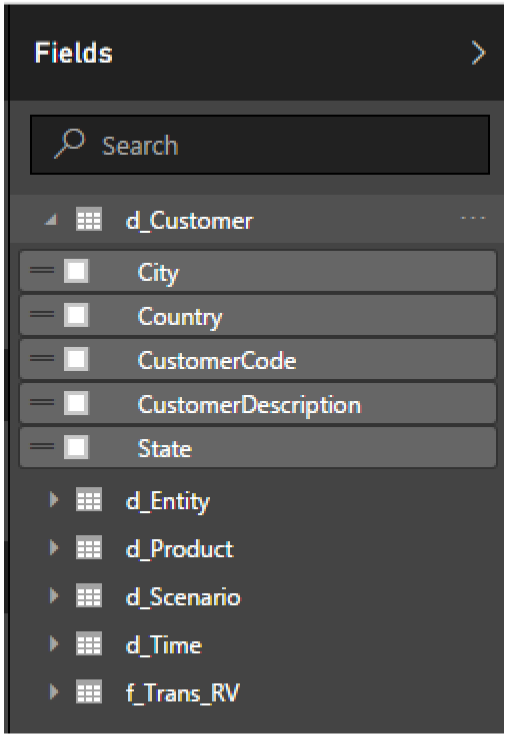
At this point, users can right click on any of these table names and then choose the **Rename** option in order to assign more user friendly names. Users can also do the same for individual fields within each table. For the sake of this exercise, the table names will be left as is but a number of other fields will be relabeled and hidden in order to de-clutter our workspace a bit. To do this, we'll select **Data View** on the left had side of the page.



MemberId	Code	Description	Alias	CustomerClassCode	CustomerNumber	City	State
1	AARON0001	Aaron Fitz Electrical				Beijing	
2	ADAMP0001	Adam Park Resort				New Discovery	QC
3	ADVAN0001	Advanced Paper Co.				Johannesburg	
4	ADVAN0002	Advanced Tech Satellite System				El Paso	TX
5	ALTON0001	Alton Manufacturing				Tokyo	
6	AMERI0001	American Science Museum				Black Bear	YK
7	AMERI0002	American Electrical Contractor				Berlin	
8	ASSOC0001	Associated Insurance Company				Memphis	TN
9	ASTOR0001	Astor Suites				Dongguan	
10	ATMOR0001	Atmore Retirement Center				Defiance	ON

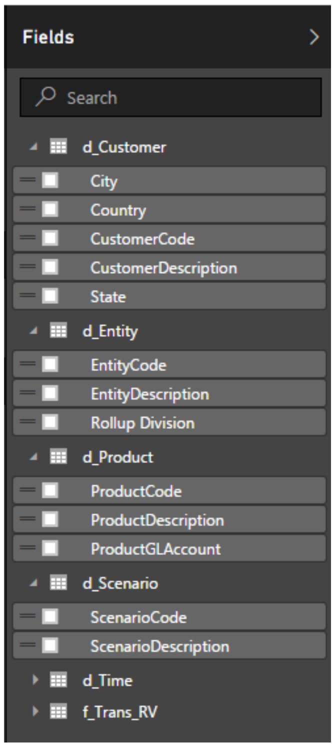
By default, the first table on our list, d_Customer, is shown. From here users can right click on any column heading to either rename it or hide it in Report View.

In this example, a few columns in the d_Customer table will be renamed and a few will be hidden so that they will no longer appear in Report View. The end result when switching back to Report View and expand the d_Customer table is as follows.

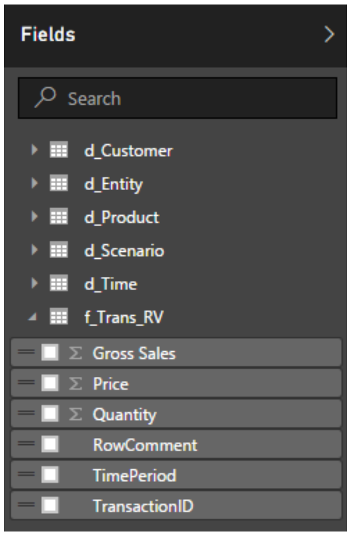


Note that the ‘Code’ and ‘Description’ column labels were changed to ‘CustomerCode’ and ‘CustomerDescription’. This may help to avoid confusion since Code and Description are common column names across virtually all of our dimension tables.

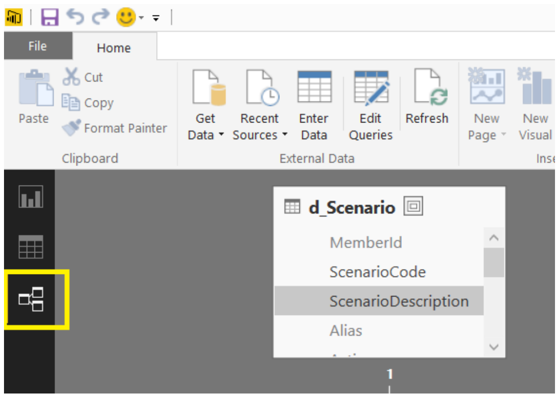
In the below example, the same has also been done for d_Scenario, d_Product and d_Entity.


Fields
Search
d_Customer
City
Country
CustomerCode
CustomerDescription
State
d_Entity
EntityCode
EntityDescription
Rollup Division
d_Product
ProductCode
ProductDescription
ProductGLAccount
d_Scenario
ScenarioCode
ScenarioDescription
d_Time
f_Trans_RV

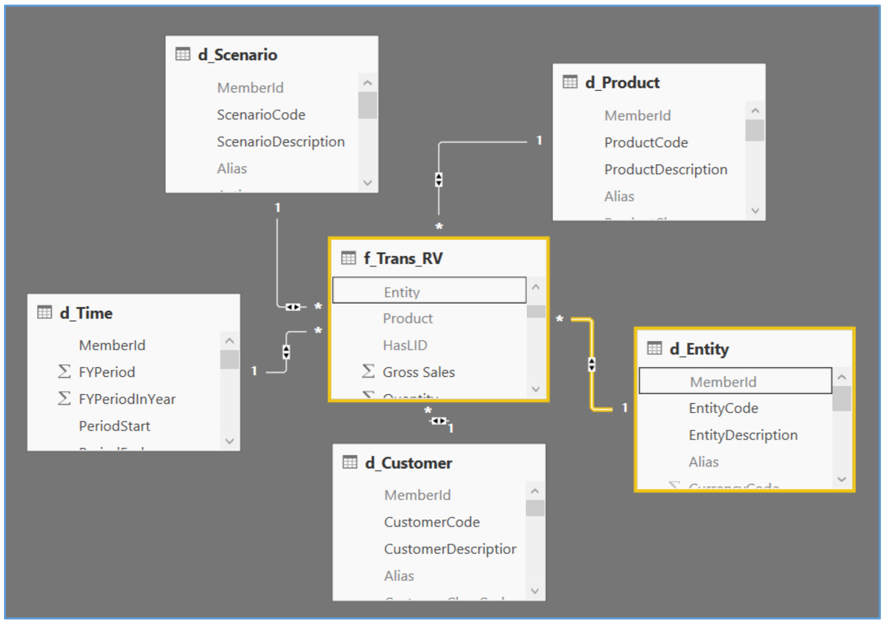
Finally, we’ll make a few changes to f_Trans_RV by relabeling the Value1, Value2, and Value3 columns to more accurately reflect their content and removing a number of columns that display the member ID’s for each dimension. This simplifies the task of finding relevant content in Report View.


Fields
Search
d_Customer
d_Entity
d_Product
d_Scenario
d_Time
f_Trans_RV
Σ Gross Sales
Σ Price
Σ Quantity
RowComment
TimePeriod
TransactionID

Before proceeding to build visualization of this data in Report View, users should first look at how the tables we selected are linked together in Relationship View.



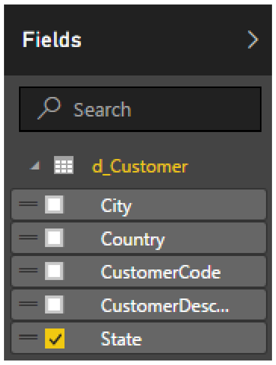
Because the data is coming from a relational source, *Power BI* automatically discovers any Primary/Foreign key relationships between source tables and creates the necessary links as shown here.



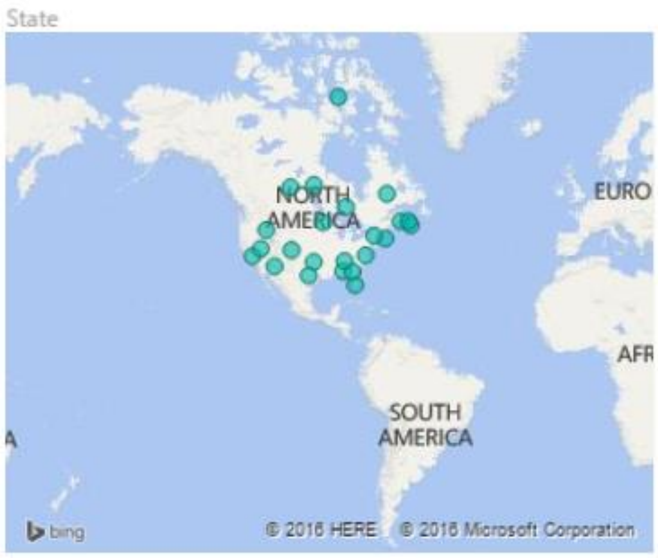
When combining BI360*DW* data with data from an outside source (say, from Excel) it is also possible to create a link manually between that source and a BI360*DW* table using Relationship View.

Returning to Report View, users can now create visualizations using the data set that was imported from the BI360*DW*. As a quick example, below a report that summarizes US domestic sales by state will be created.

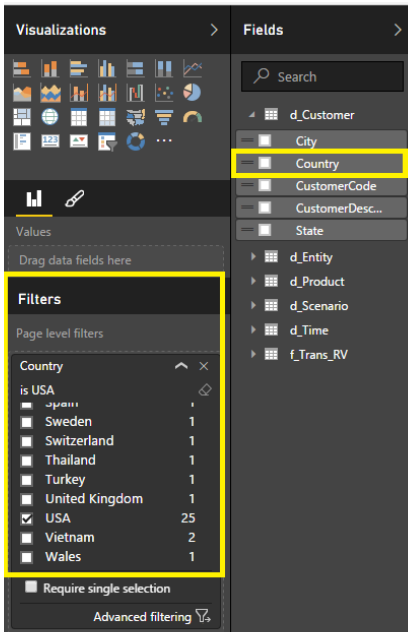
For starters, expand the field list under d_Customer and choose *State*.



Power BI automatically recognizes the geographic nature of the data in this field and inserts a map visualization on the report page.

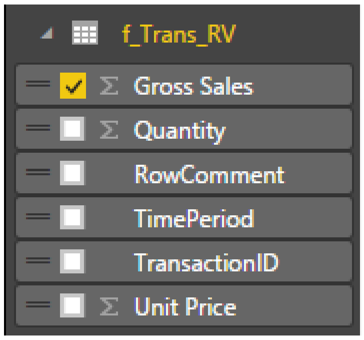


Next, modify this a bit, dragging the *Country* field from d_Customer into the Page level filters area, and then choose USA from the list.

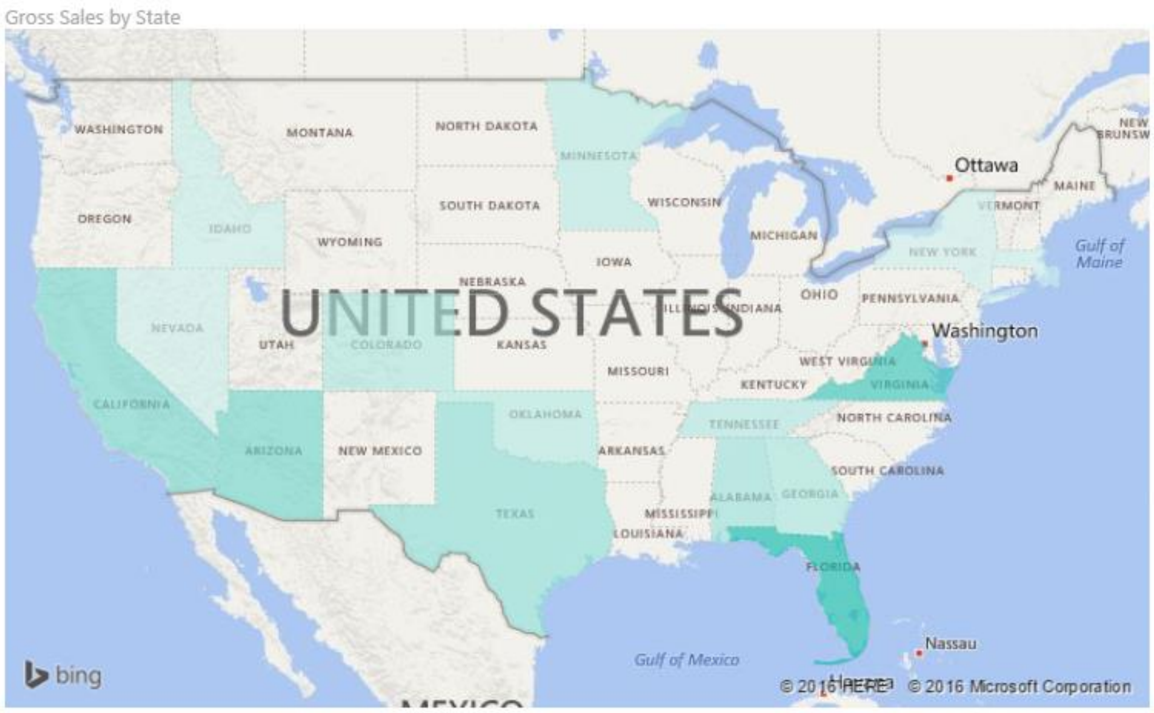


Sweden	1
Switzerland	1
Thailand	1
Turkey	1
United Kingdom	1
✓ USA	25
Vietnam	2
Wales	1

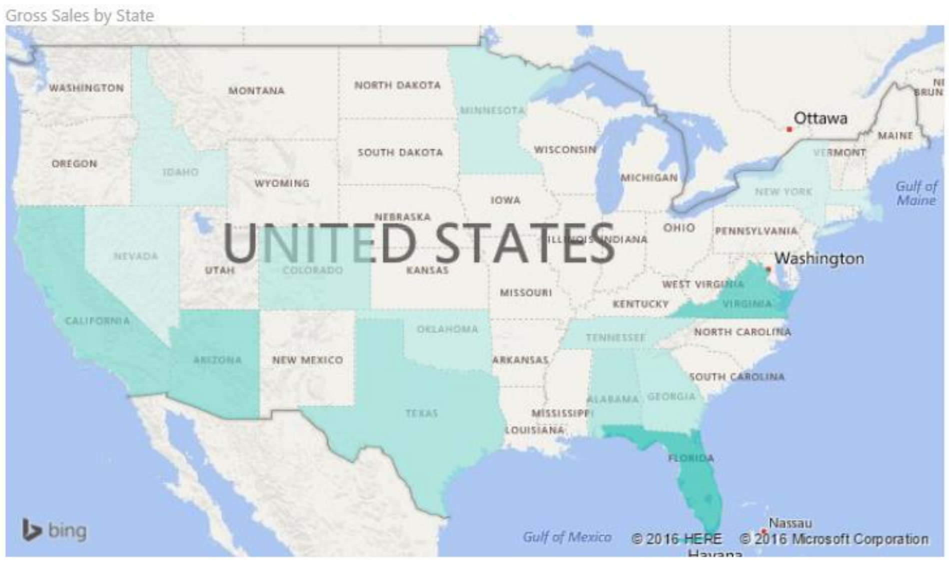
Next, expand the field list under f_Trans_RV and choose 'Gross Sales' (Value1).



Returning to the Report page, click on the map visualization, then click on the 'Filled Map' icon in the Visualizations area in order to show a more detailed map with individual states shaded according to Gross Revenue reported within each state.



To supplement this map view, add other visualizations such as a table showing the dollar amount of gross sales by state.

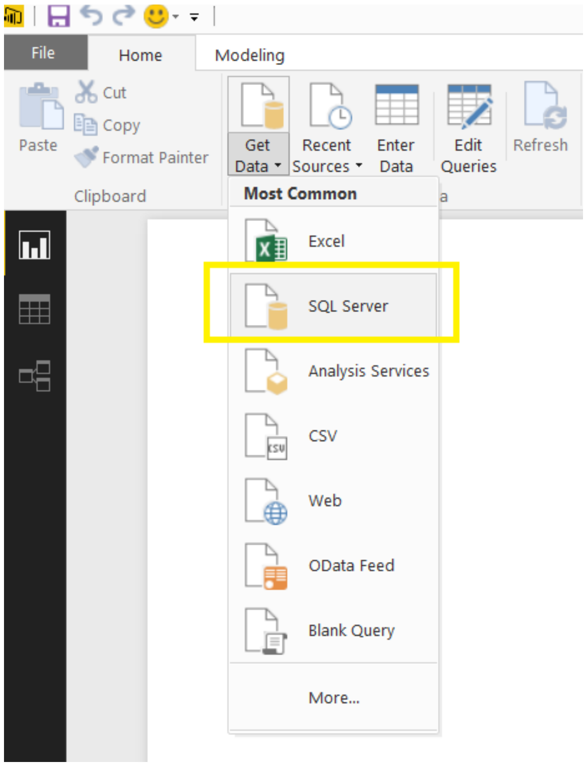


State	Gross Sales
FL	4,357,680.90
VA	3,371,861.81
AZ	2,134,977.40
CA	2,011,736.32
TX	1,487,407.53
AL	1,189,538.67
GA	934,375.22
CO	742,534.10
TN	728,546.00
OK	637,481.00
MN	546,416.00
ID	178,503.02
MA	99,840.47
NV	75,374.44
NY	75,256.83
Total	**18,571,529.71**

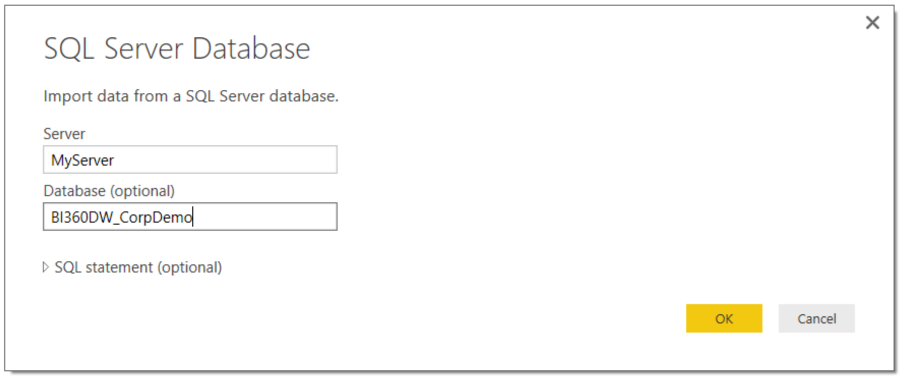
Example 2: Creating a Dataset Based on a SQL Query

In this second example, a T-SQL query is used to extract a dataset from the GL module in BI360*DW* and take care of tasks involved in 'shaping' the data into the desired format, such the column labeling and table joins.

After opening a new or existing *Power BI* desktop file, we'll go to **Home > Get Data > SQL Server**.



Then we'll enter the server and database name when prompted.

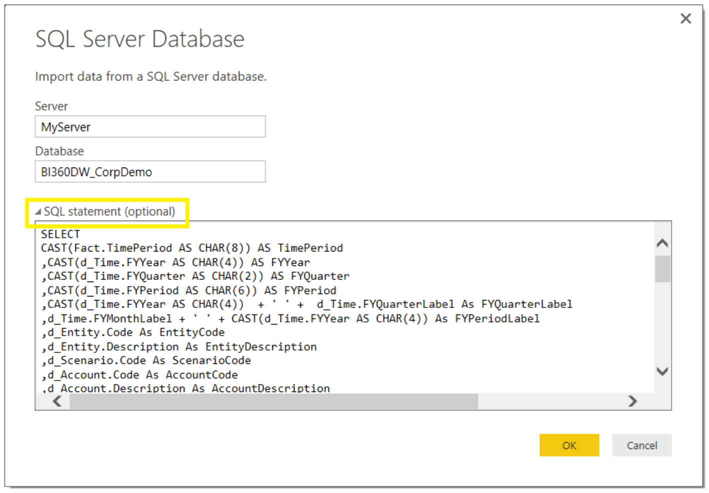

SQL Server Database

Import data from a SQL Server database.

Server
MyServer

Database (optional)
BI360DW_CorpDemo

We'll then expand the **SQL Statement** field and paste in a T-SQL query, as shown below.


SQL Server Database

Import data from a SQL Server database.

Server: MyServer

Database: BI360DW_CorpDemo

SQL statement (optional)

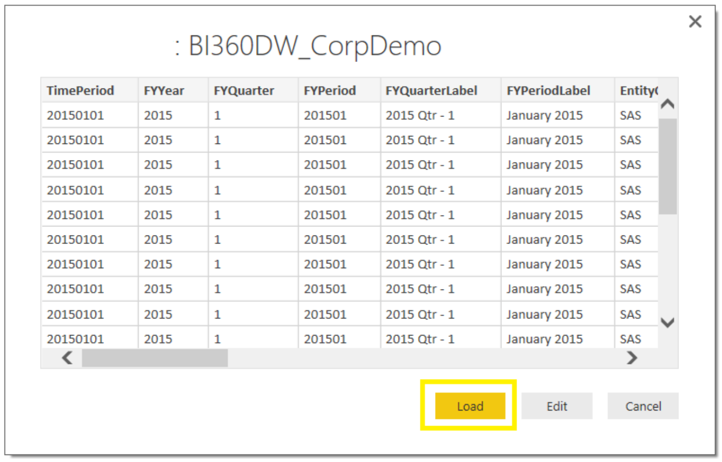
```
SELECT
CAST(Fact.TimePeriod AS CHAR(8)) AS TimePeriod
,CAST(d_Time.FYYear AS CHAR(4)) AS FYYear
,CAST(d_Time.FYQuarter AS CHAR(2)) AS FYQuarter
,CAST(d_Time.FYPeriod AS CHAR(6)) AS FYPeriod
,CAST(d_Time.FYYear AS CHAR(4))  + ' ' +  d_Time.FYQuarterLabel As FYQuarterLabel
,d_Time.FYMonthLabel + ' ' + CAST(d_Time.FYYear AS CHAR(4)) As FYPeriodLabel
,d_Entity.Code As EntityCode
,d_Entity.Description As EntityDescription
,d_Scenario.Code As ScenarioCode
,d_Account.Code As AccountCode
,d_Account.Description As AccountDescription
```

*For your reference, the full text of the query is shown in **Appendix A** of this document.*

The sample query in this example takes care of the legwork of formatting our data set so that we won't have to do so in *Power BI* desktop, specifically by:

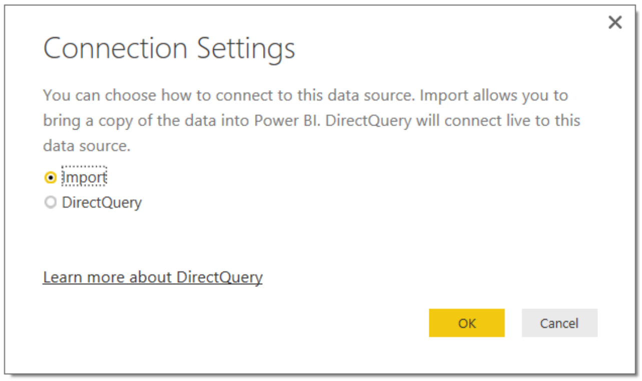
- Joining our source fact table, **f_Trans_GL**, to the relevant dimension tables such as d_Time, d_Entity, d_Scenario, etc.
- Using the CAST function to convert source columns to data types that will work best for our purposes. For example, the FYYear value (which represents the fiscal year) is stored in the source table as an integer. *Power BI* by default will try to treat this column as a measure on which mathematical operations can be performed. To avoid this, we'll cast it as text so that we can use it as an axis grouping or filter in charts and graphs.
- Using the AS keyword to assign user friendly labels to source columns.
- Adding a derived column that can be used to sort the Account Category column, which we'll use to build visualizations in the below examples.

Power BI then shows a preview based on the results of the query entered during the previous step. Click on **Load** to import the data into the model.


: BI360DW_CorpDemo

TimePeriod	FYYear	FYQuarter	FYPeriod	FYQuarterLabel	FYPeriodLabel	Entity(
20150101	2015	1	201501	2015 Qtr - 1	January 2015	SAS
20150101	2015	1	201501	2015 Qtr - 1	January 2015	SAS
20150101	2015	1	201501	2015 Qtr - 1	January 2015	SAS
20150101	2015	1	201501	2015 Qtr - 1	January 2015	SAS
20150101	2015	1	201501	2015 Qtr - 1	January 2015	SAS
20150101	2015	1	201501	2015 Qtr - 1	January 2015	SAS
20150101	2015	1	201501	2015 Qtr - 1	January 2015	SAS
20150101	2015	1	201501	2015 Qtr - 1	January 2015	SAS
20150101	2015	1	201501	2015 Qtr - 1	January 2015	SAS
20150101	2015	1	201501	2015 Qtr - 1	January 2015	SAS

As in the previous example, the options to either **import** the data into our *Power BI* desktop file or use **DirectQuery**, which creates a live connection to the source, are presented.



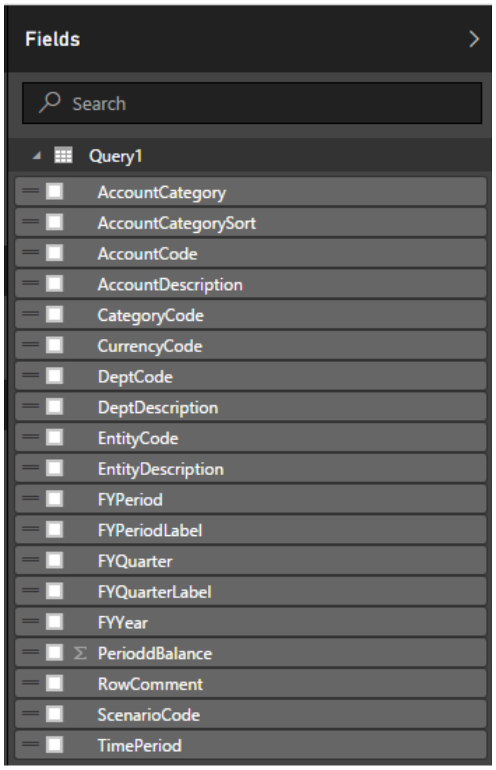
Connection Settings

You can choose how to connect to this data source. Import allows you to bring a copy of the data into Power BI. DirectQuery will connect live to this data source.

[✓ Import]
DirectQuery

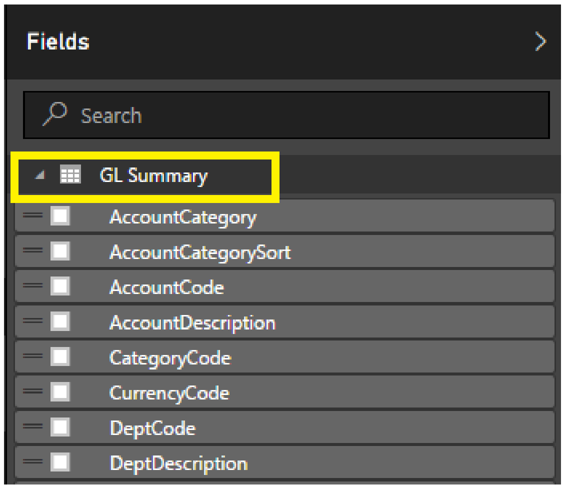
Learn more about DirectQuery

Our data set now appears in report view as a single set of columns.

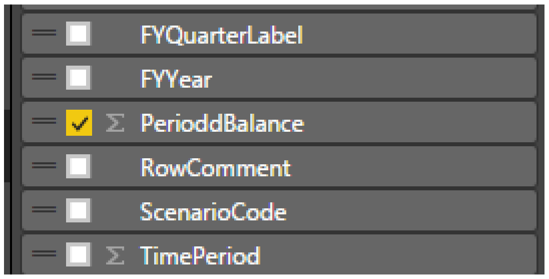


Query1
- AccountCategory
- AccountCategorySort
- AccountCode
- AccountDescription
- CategoryCode
- CurrencyCode
- DeptCode
- DeptDescription
- EntityCode
- EntityDescription
- FYPeriod
- FYPeriodLabel
- FYQuarter
- FYQuarterLabel
- FYYear
- Σ PerioddBalance
- RowComment
- ScenarioCode
- TimePeriod

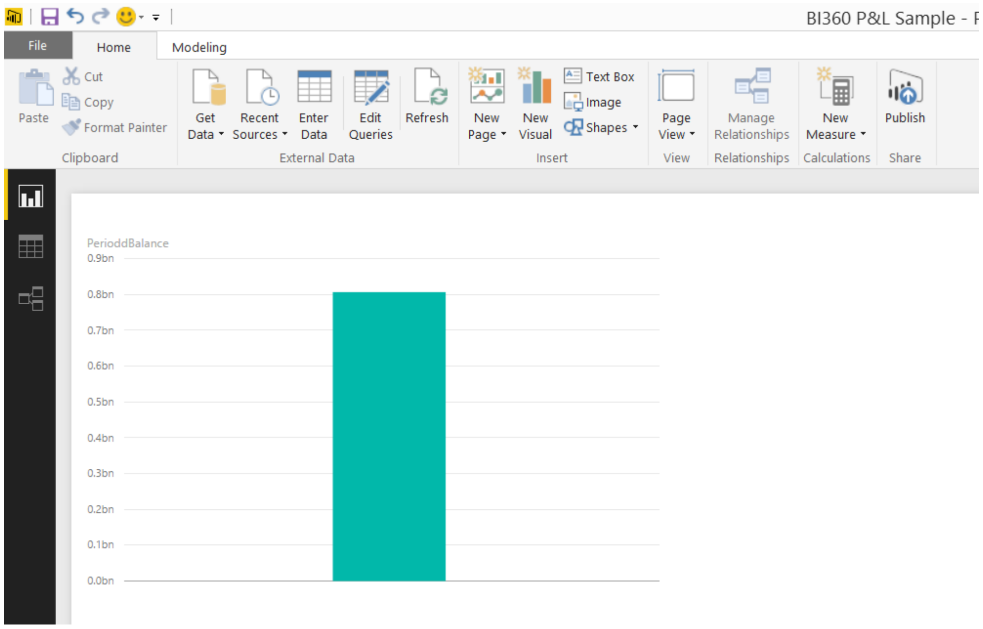
Rename this new dataset by right clicking on the default label, ‘Query1’, and changing it to ‘GL Summary’.



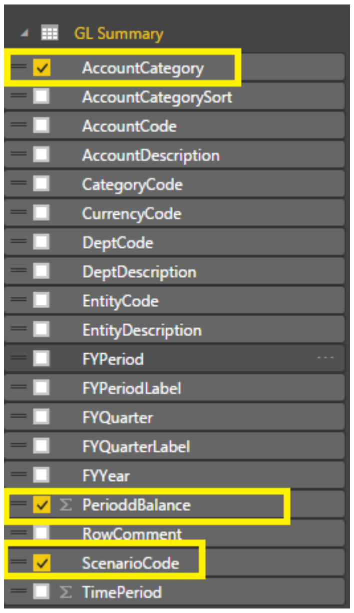
To get started with a new visualization using this data, click on **PeriodBalance**.



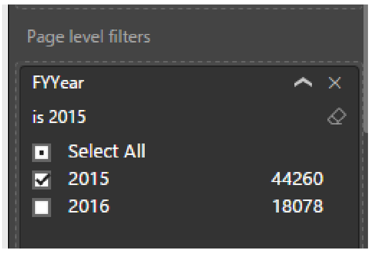
Since this field is a measure, *Power BI* automatically inserts a new bar chart into Report view.



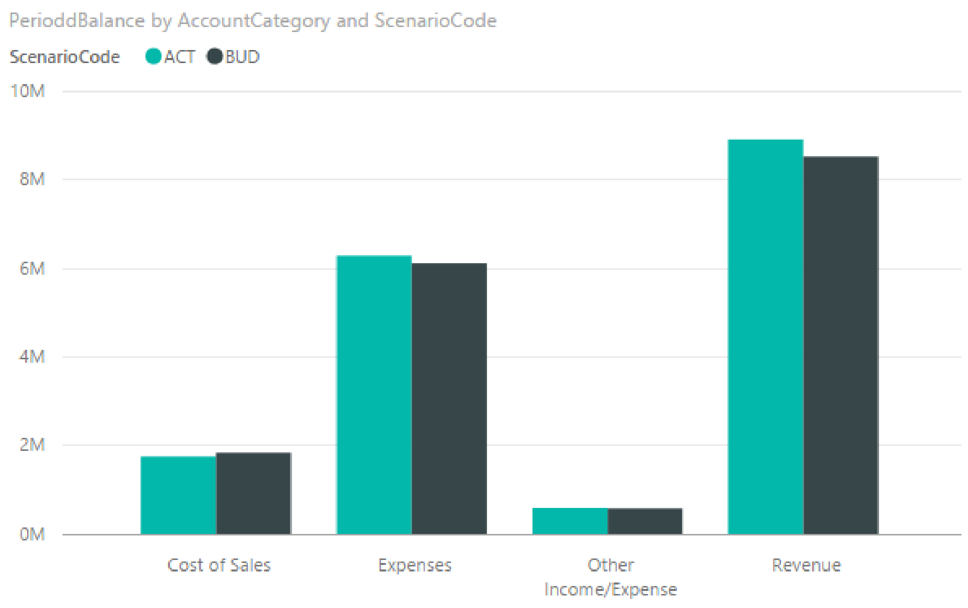
To add a bit more context, also select AccountCategory and ScenarioCode from the sample data set.



Also drag **FYYear** into the page level filers and select 2015.



The end result is bar graph that compares Actual to Budget amounts by Account Category for fiscal year 2015.

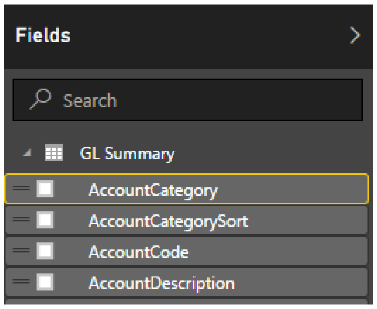


Note that the chart properties have been modified to show the Y-Axis labels as $M instead of $Bn.

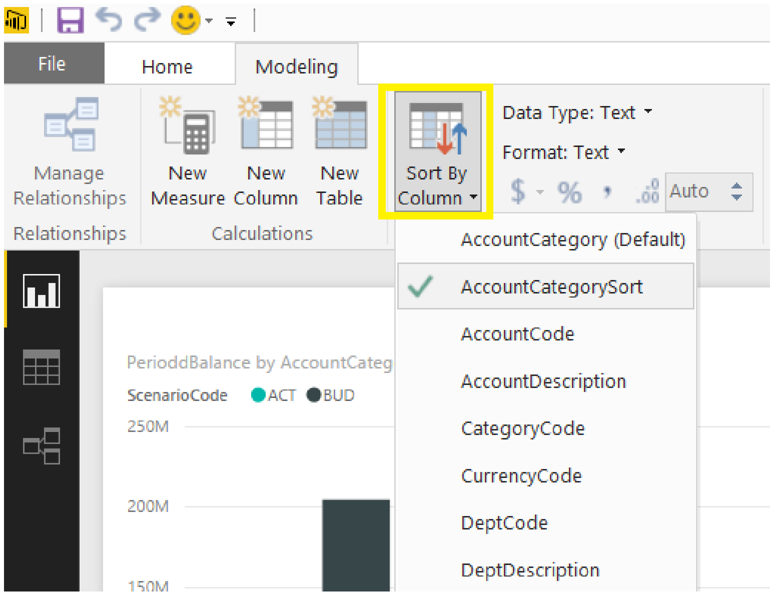
Also note that the account types are sorted alphabetically from left to right, so that the Cost of Sales, Expenses and Other Income/Expense categories all precede Revenue. This might not make much sense to an audience accustomed to seeing Revenue displayed first on financial statements.

In the sample dataset we have custom column that can be used to remedy this issue. This derived column was added to the query to control the sort order of the Account Category column, so that Revenue can be first followed by other categories according to their usual order on a P&L statement. To apply this custom sorting to the bar chart, we can select the field we want to sort differently, and then apply a custom sort order based on another column in our dataset.

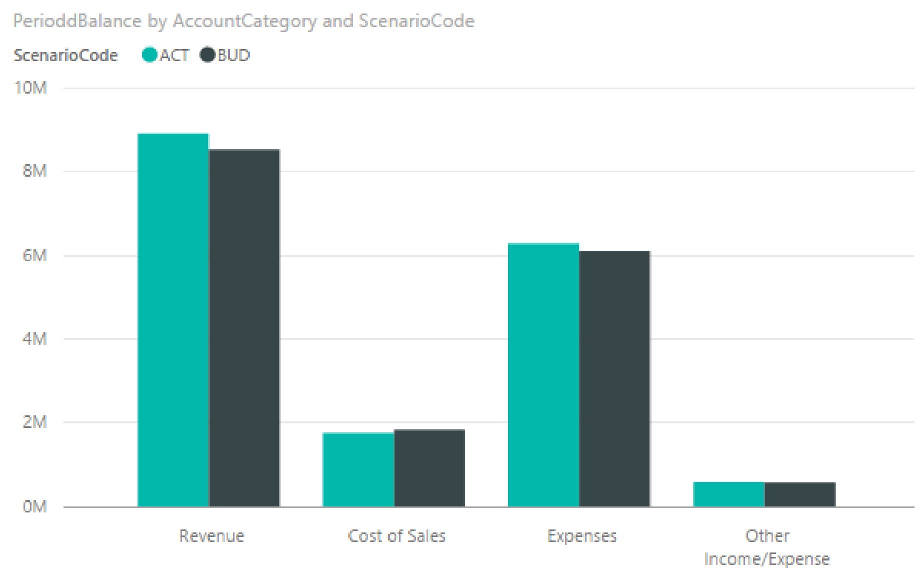
First, select the column to be sorted differently.



Then, click on the **Modeling** tab at the top of the screen and chose 'Sort By Column'. For the sake of this example, choose the custom column that was created for this purpose in the sample SQL query, AccountCategorySort.



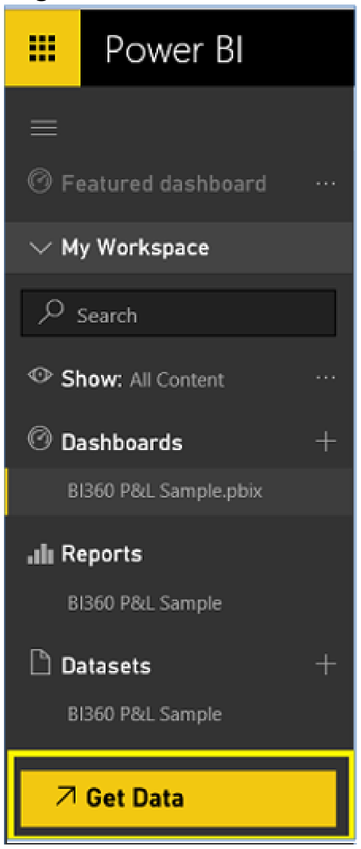
This change will then be reflected in the bar chart.



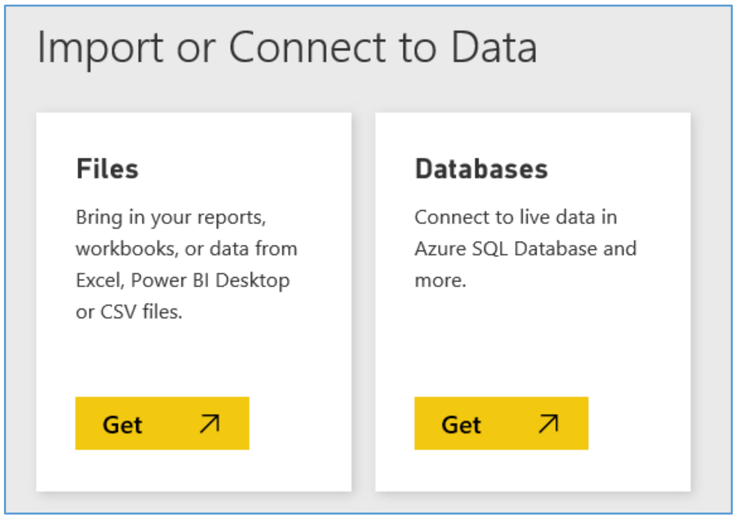
Publishing to the Power BI Service

Whenever the time has come to make the database and reports available to other users, users can deploy the content of a *Power BI* desktop file to the cloud-based *Power BI* service using either of two options.

1. Log in to the *Power BI* service and choose "Get Data".



The site currently provides two options that allow the user to import from files or selected database platforms.


Import or Connect to Data

Files

Bring in your reports, workbooks, or data from Excel, Power BI Desktop or CSV files.

Get ↗

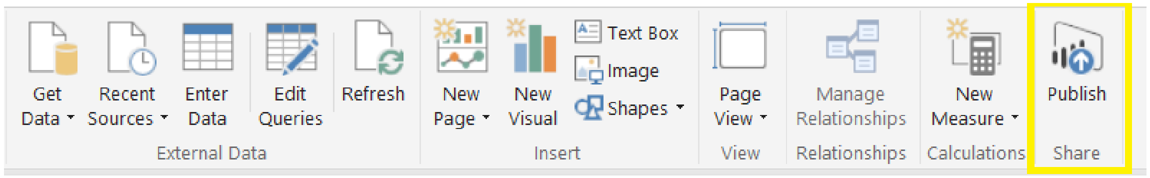
Databases

Connect to live data in Azure SQL Database and more.

Get ↗

Choose the **Files** option to import data and reports from a *Power BI* desktop (.pbix) file, such as the ones shown in the previous examples.

2. Users can also publish directly from the *Power BI* desktop client by clicking on the **Publish** option under the **Home** menu.



Users will be prompted to provide their user credentials for the *Power BI* service, after which the users will receive confirmation once the content of the model is uploaded.

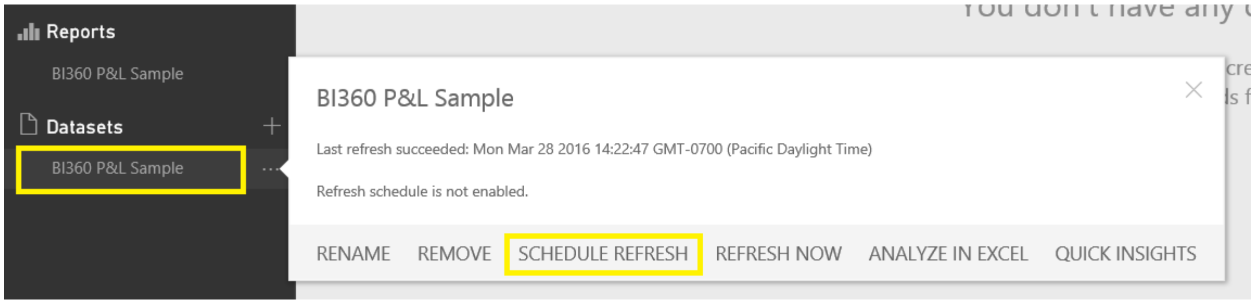
The users should now see any datasets and reports that were authored in this model included in the workspace the next time that user signs in to the *Power BI* service.

Using a Power BI Gateway to Schedule Updates

The *Power BI* Gateway is an application that is installed and run on-premises and serves as a bridge providing scheduled, secure data transfers between on-premises data sources and the *Power BI* service.

There are two types of gateways. The **Personal Gateway** runs on an individual workstation and supports a single user, while the **Enterprise Gateway** enables IT departments to deploy and manage central gateways to serve a large group of users. With the Enterprise Gateway, an IT department can centrally manage the set of users who have access to the underlying data sources and have visibility into gateway usage. Either gateway can be downloaded using links provided on the *Power BI* site.

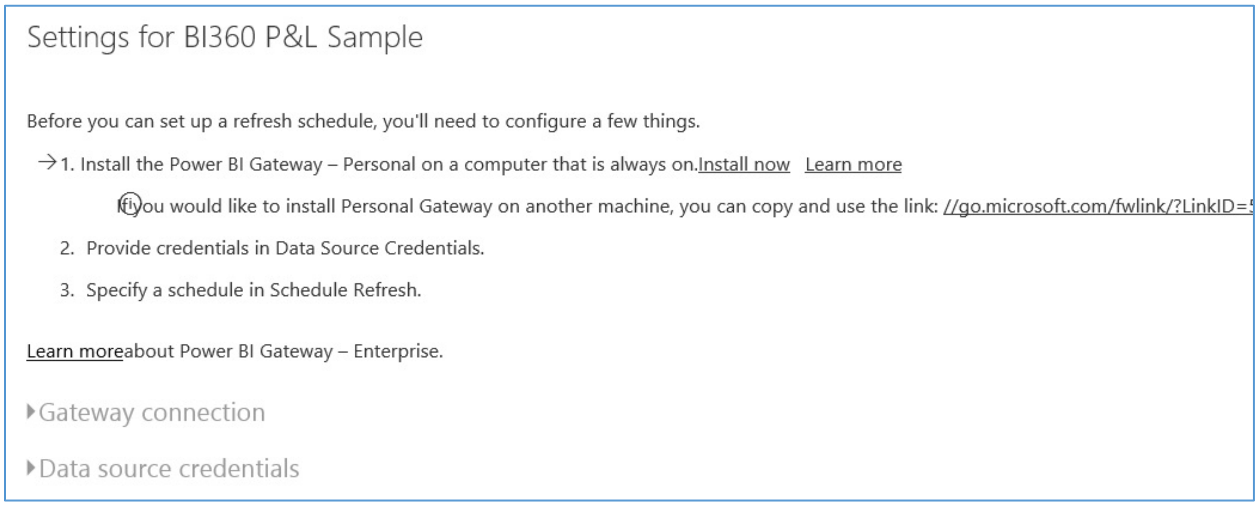
To configure scheduled updates of your data using a gateway, log in to *Power BI*, then right click on a dataset to be updated. Then choose **Schedule Refresh**.



Last refresh succeeded: Mon Mar 28 2016 14:22:47 GMT-0700 (Pacific Daylight Time)

Refresh schedule is not enabled.

This takes the user to a settings page specific to the selected dataset. As shown below, the user first need to install and/or configure a gateway if it has not been done so already.



Settings for BI360 P&L Sample

Before you can set up a refresh schedule, you'll need to configure a few things.

→1. Install the Power BI Gateway – Personal on a computer that is always on.Install now Learn more

If you would like to install Personal Gateway on another machine, you can copy and use the link: //go.microsoft.com/fwlink/?LinkID=

2. Provide credentials in Data Source Credentials.

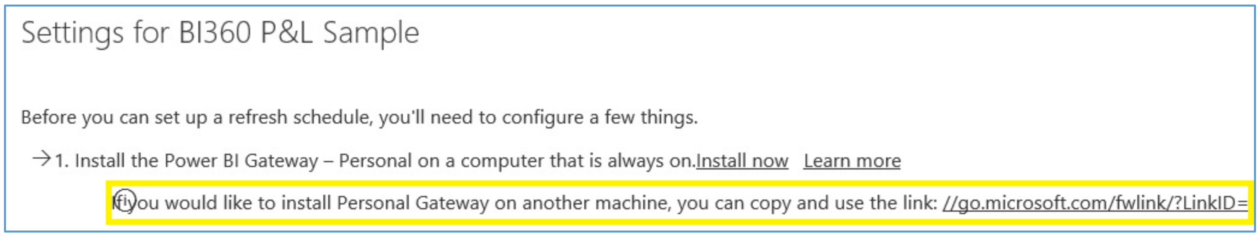
3. Specify a schedule in Schedule Refresh.

Learn moreabout Power BI Gateway – Enterprise.

This page also provides a link where the user can download the installer for the Personal Gateway. This section will review that option first and then discuss using an Enterprise Gateway later in this section.

Scheduled Updates Using a Personal Gateway

If the user has not already installed the Personal Gateway, they can do so by clicking the link provided on the Settings page.



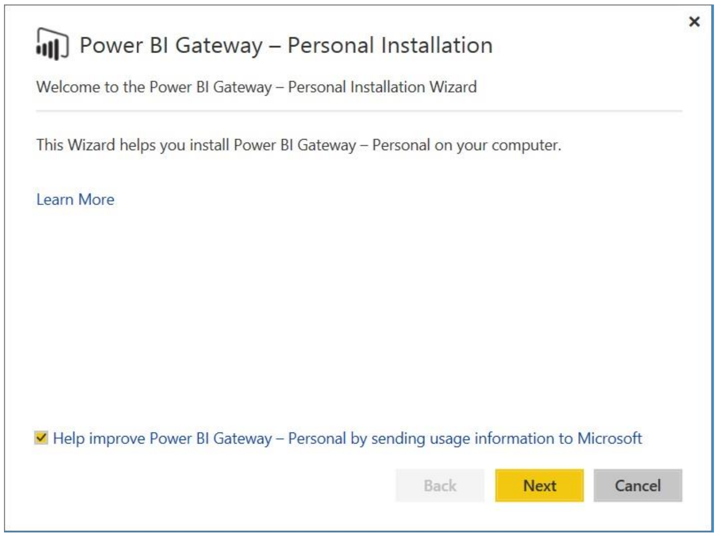
Settings for BI360 P&L Sample

Before you can set up a refresh schedule, you'll need to configure a few things.

→1. Install the Power BI Gateway – Personal on a computer that is always on.Install now Learn more

If you would like to install Personal Gateway on another machine, you can copy and use the link: //go.microsoft.com/fwlink/?LinkID=

After launching the installer file, a Wizard takes you through the steps of the installation.



Power BI Gateway – Personal Installation

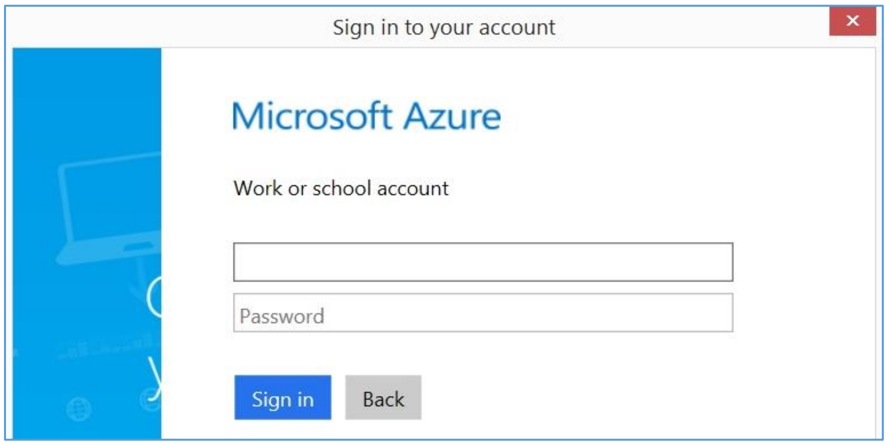
Welcome to the Power BI Gateway – Personal Installation Wizard

This Wizard helps you install Power BI Gateway – Personal on your computer.

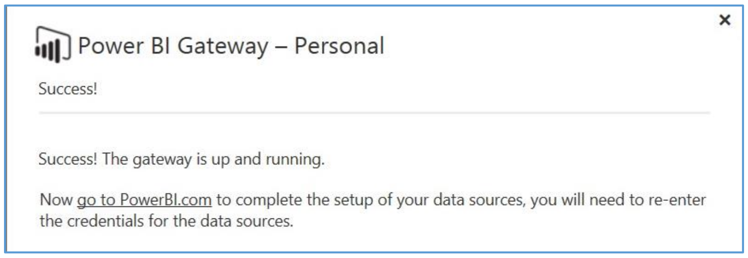
Learn More

✓ Help improve Power BI Gateway – Personal by sending usage information to Microsoft

During the installation, users will be prompted to sign in to the Microsoft Azure service using the same credentials used for *Power BI*.



The user will receive confirmation once the installation completes successfully. A link is also provided here that takes the user back to the *Power BI* site.

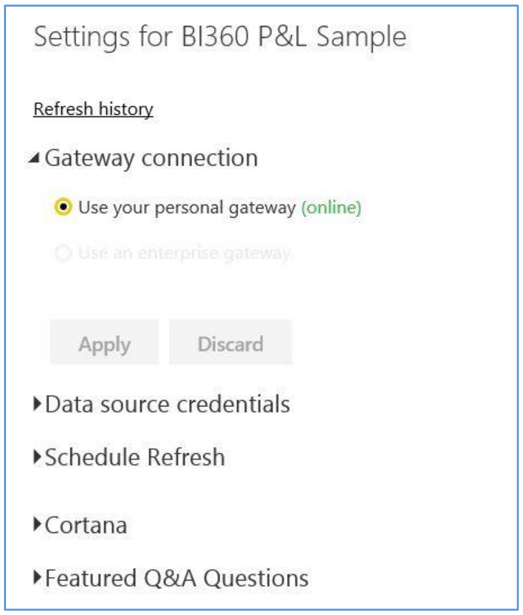

Power BI Gateway – Personal

Success!

Success! The gateway is up and running.

Now go to PowerBI.com to complete the setup of your data sources, you will need to re-enter the credentials for the data sources.

When the user return to the *Power BI* service and view the settings page for the data set, the user will notice that the personal gateway connection is now online.


Settings for BI360 P&L Sample

Refresh history

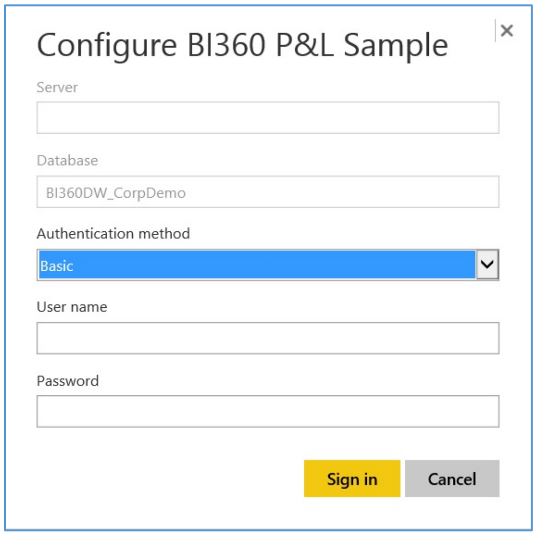
Gateway connection

[✓ Use your personal gateway (online)]

Use an enterprise gateway

Continuing on with the remaining steps necessary to schedule unattended updates.

From the settings page, click on **Data source credentials** to enter login credentials to be used by the gateway when connecting to your data source.


Configure BI360 P&L Sample

Server

Database: BI360DW_CorpDemo

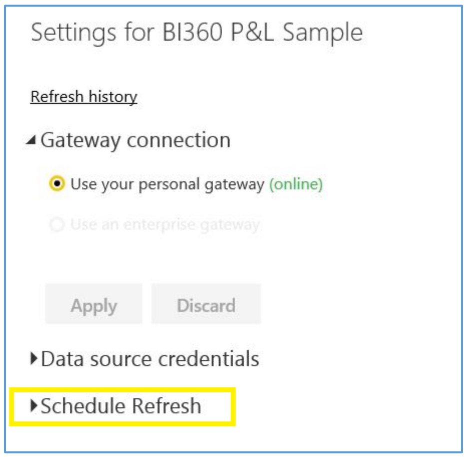
Authentication method: Basic

User name

Password

Note that these can be different from the ones normally use. For example, it might be prefer to create a SQL server login for this purpose rather than using Windows authentication.

Next, click on **Schedule Refresh** to schedule updates using the gateway and data source credentials that were configured in the previous steps.



Settings for BI360 P&L Sample

Refresh history

Gateway connection

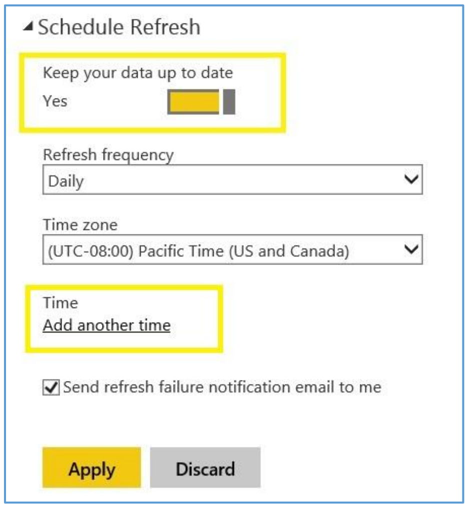
[● Use your personal gateway (online)]

Use an enterprise gateway

Data source credentials

Schedule Refresh

This exposes the available options for managing updates. First click the slider bar to enable updates, then change the Refresh frequency and time settings as needed.



Schedule Refresh

Keep your data up to date: Yes

Refresh frequency: Daily

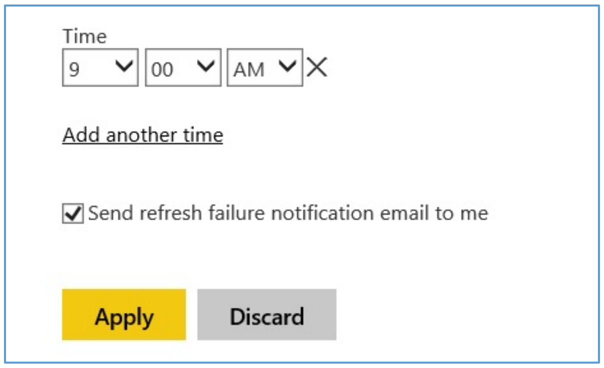
Time zone: (UTC-08:00) Pacific Time (US and Canada)

Time

Add another time

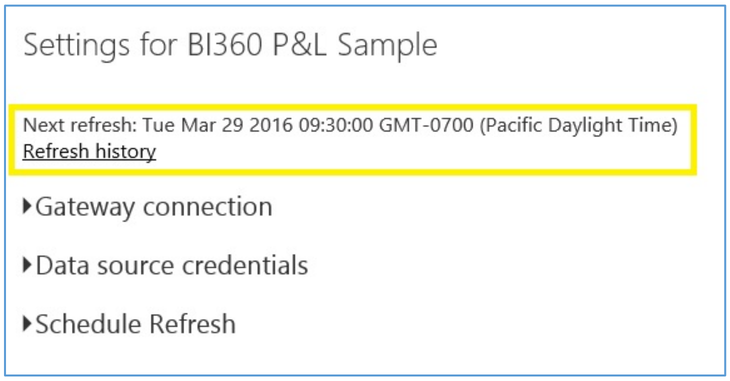
[✓] Send refresh failure notification email to me

Also note the "Add another time" link at the bottom of this section that allows you to schedule additional refresh times. This allows you to specify additional updates throughout the day. Changes are saved after clicking the **Apply** button.



☑ Send refresh failure notification email to me

When returning to the main settings page, the date and time of the next scheduled refresh will be shown.


Settings for BI360 P&L Sample

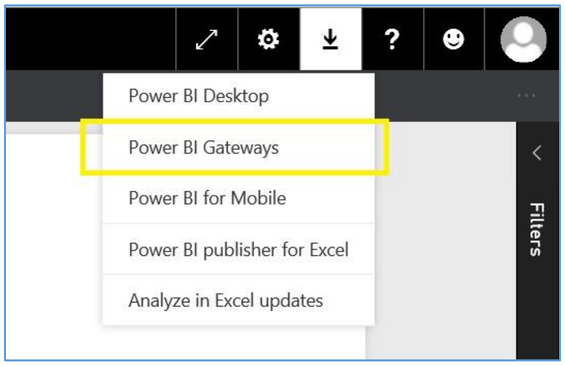
Next refresh: Tue Mar 29 2016 09:30:00 GMT-0700 (Pacific Daylight Time)
Refresh history

Note that the same personal gateway can be used when scheduling updates for other data sources and that separate installation will not be required for each source.

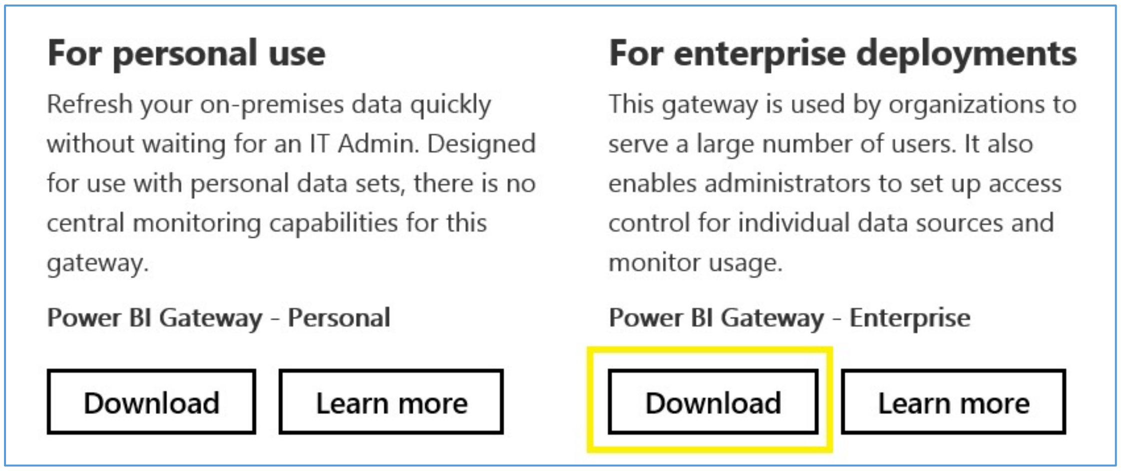
Scheduled Updates Using an Enterprise Gateway

Enterprise gateways are meant to be installed on a server and are intended to support multiple users. A separate download link and installer are provided for the Enterprise Gateway. The installation also involves a few additional options and steps.

If the organization does not currently have an Enterprise gateway, it can be downloaded and installed from within the *Power BI* service web site by clicking on the Downloads icon. If you are planning to install this in a Production environment, users will probably want to plan this deployment with your IT Team.



Choose "*Power BI* Gateways" from the menu, and then click "Download" under the heading, "For enterprise deployments".



For personal use

Refresh your on-premises data quickly without waiting for an IT Admin. Designed for use with personal data sets, there is no central monitoring capabilities for this gateway.

Power BI Gateway - Personal

Download | Learn more

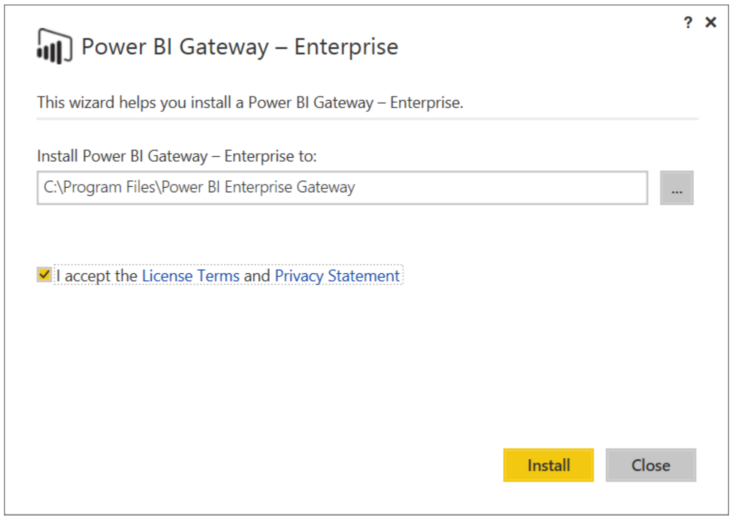
For enterprise deployments

This gateway is used by organizations to serve a large number of users. It also enables administrators to set up access control for individual data sources and monitor usage.

Power BI Gateway - Enterprise

Download | Learn more

After launching the installer file, a Wizard takes you through the steps of the installation.


Power BI Gateway – Enterprise

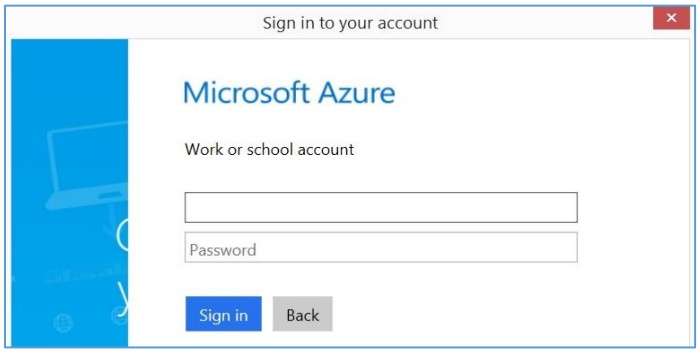
This wizard helps you install a Power BI Gateway – Enterprise.

Install Power BI Gateway – Enterprise to:

C:\Program Files\Power BI Enterprise Gateway

[✓] I accept the License Terms and Privacy Statement

When the installation completes, users will be prompted to sign in to the Microsoft Azure service using the same credentials used for *Power BI*.

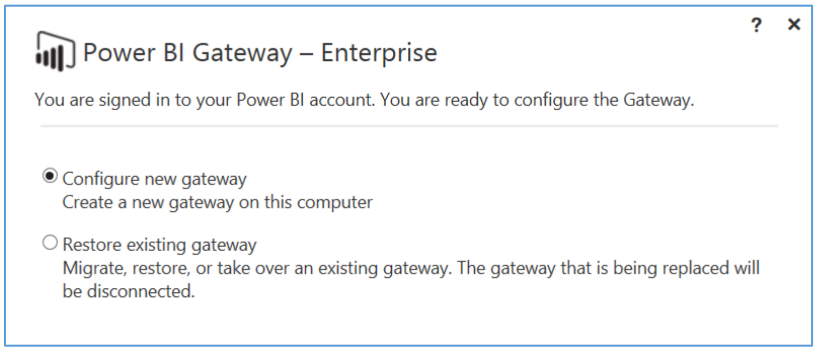

Sign in to your account

Microsoft Azure

Work or school account

Password

At the next screen, choose ‘Configure new gateway’ and then click ‘Next’.

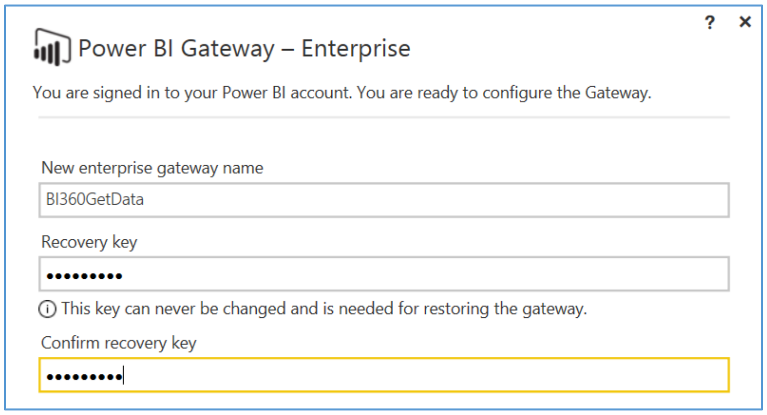

Power BI Gateway – Enterprise

You are signed in to your Power BI account. You are ready to configure the Gateway.

● Configure new gateway
Create a new gateway on this computer

○ Restore existing gateway
Migrate, restore, or take over an existing gateway. The gateway that is being replaced will be disconnected.

The user will first be prompted to enter a name and recovery key for the gateway. Click configure when this step is complete.


Power BI Gateway – Enterprise

You are signed in to your Power BI account. You are ready to configure the Gateway.

New enterprise gateway name

BI360GetData

Recovery key

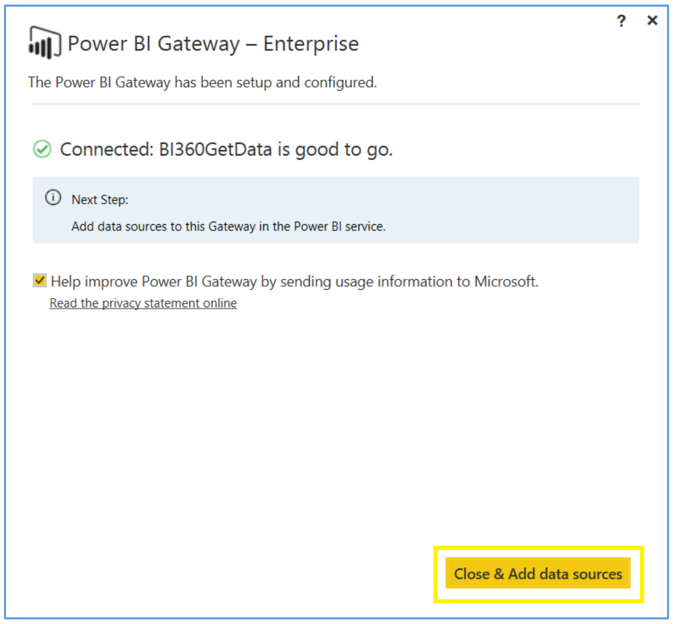
•••••••••

ⓘ This key can never be changed and is needed for restoring the gateway.

Confirm recovery key

•••••••••

The user will then receive confirmation when these changes are stored successfully. Now the user can proceed to adding new data sources.


Power BI Gateway – Enterprise

The Power BI Gateway has been setup and configured.

Connected: BI360GetData is good to go.

ⓘ Next Step:

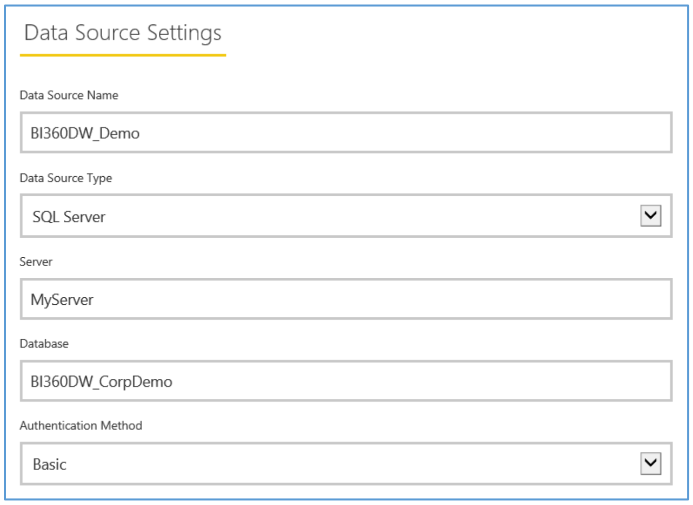
Add data sources to this Gateway in the Power BI service.

[✓] Help improve Power BI Gateway by sending usage information to Microsoft.

Read the privacy statement online

Clicking on the **Close & Add data sources** button takes you back to the *Power BI* site, where the user will again be prompted to log in.

After logging in, the user is then taken to a page where enter all the properties of the data source (*BI360DW*), including the server name, database name, and type of authentication used.


Data Source Settings

Data Source Name: BI360DW_Demo

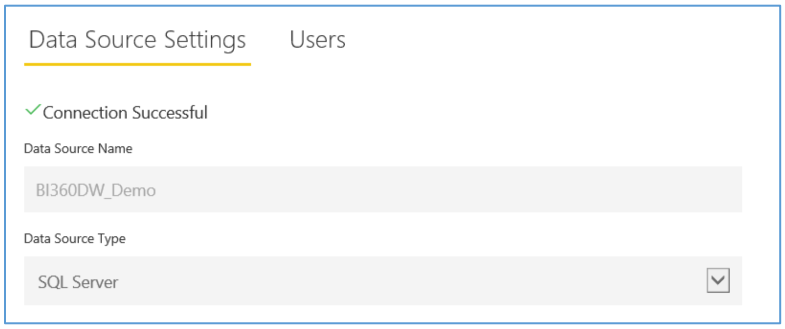
Data Source Type: SQL Server

Server: MyServer

Database: BI360DW_CorpDemo

Authentication Method: Basic

The user is also required to enter credentials to be used by the gateway when connecting to the data source. These are encrypted using a key stored on premises. The user will receive notification once a successful connection is made .



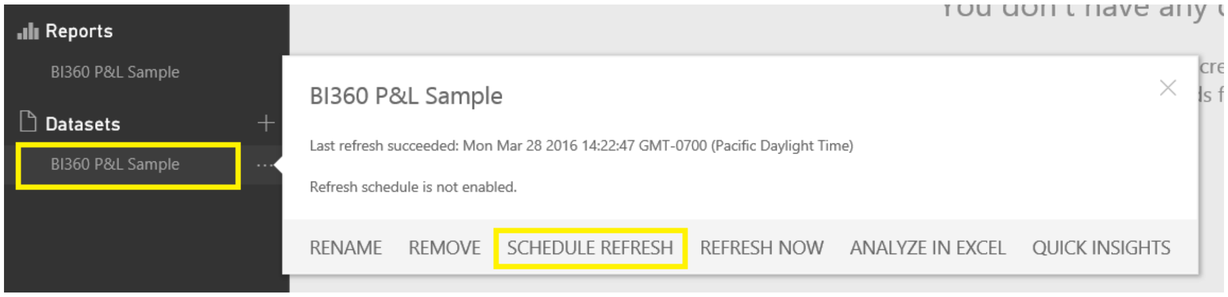
Connection Successful

Data Source Name: BI360DW_Demo

Data Source Type: SQL Server

Continue to the 'Users' tab to make this data source available to other *Power BI* users. By default, the logged in user's account will already be shown here.

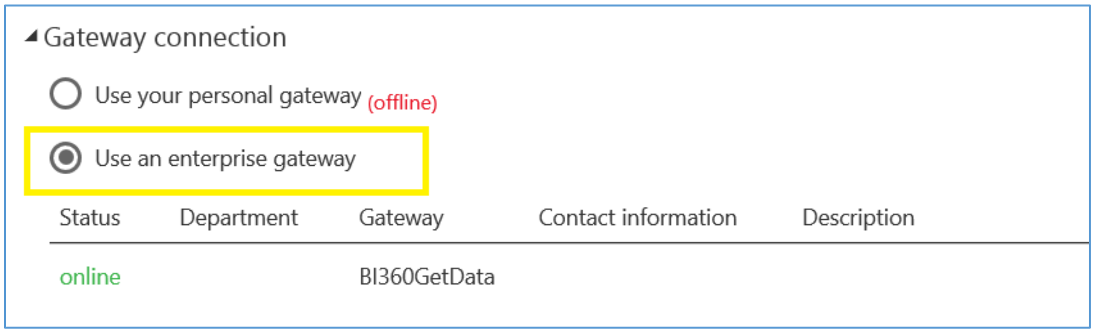
The procedure for updating a dataset using an Enterprise Gateway is similar to the one shown earlier for the personal Gateway. To schedule updates, log in to *Power BI*, then right click on a dataset to be updated. Then choose 'Schedule Refresh'.



Last refresh succeeded: Mon Mar 28 2016 14:22:47 GMT-0700 (Pacific Daylight Time)

Refresh schedule is not enabled.

This will take you to the settings page for the dataset you selected. If an enterprise gateway is available, it will be shown under the heading, 'Gateway connection'.

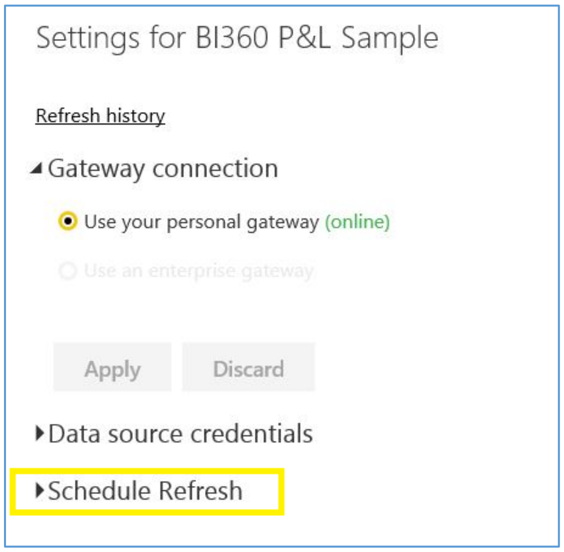

Gateway connection

Use your personal gateway (offline)

[✓ Use an enterprise gateway]

Status	Department	Gateway	Contact information	Description
online		BI360GetData		

Next, the user can click on **Schedule Refresh** to schedule updates using the gateway and data source credentials that were configured in the previous steps.


Settings for BI360 P&L Sample

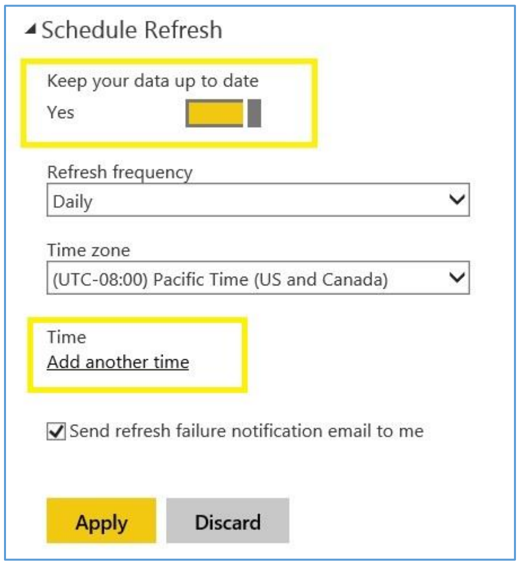
Refresh history

Gateway connection

[✓ Use your personal gateway (online)]

Use an enterprise gateway

This exposes the available options for managing updates. First click the slider bar to enable updates, then change the Refresh frequency and time settings as needed.


Schedule Refresh

Keep your data up to date: Yes

Refresh frequency: Daily

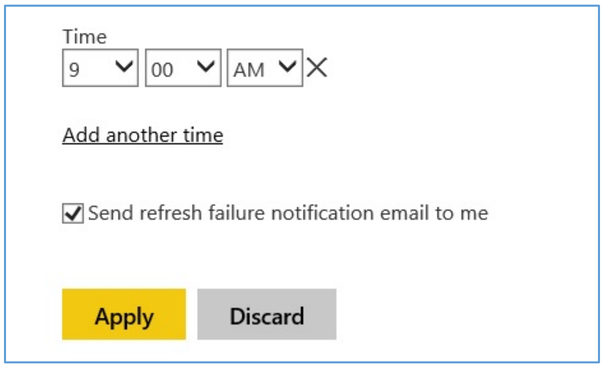
Time zone: (UTC-08:00) Pacific Time (US and Canada)

Time

Add another time

☑ Send refresh failure notification email to me

Also note the **Add another time** link at the bottom of this section that allows the user to schedule additional refresh times. This allows you to specify additional updates throughout the day. Changes are saved after clicking the **Apply** button.

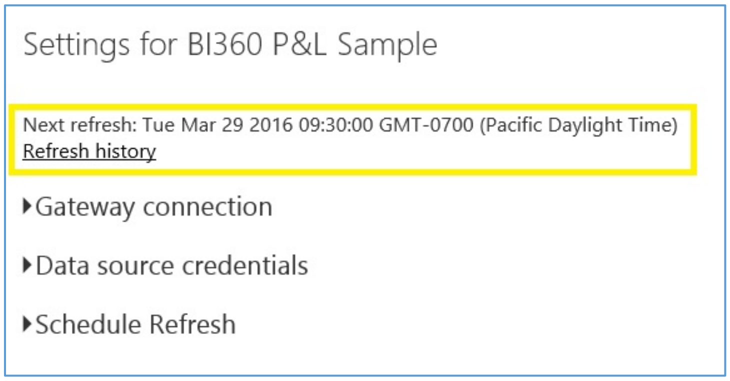

Time

9 | 00 | AM

Add another time

☑ Send refresh failure notification email to me

When returning to the main settings page, the date and time of the next scheduled refresh will be shown.


Settings for BI360 P&L Sample

Next refresh: Tue Mar 29 2016 09:30:00 GMT-0700 (Pacific Daylight Time)

Refresh history

Appendix

Additional Resources

The Solver Support Center (support.solverusa.com) is the centralized location for users to learn more about the BI360 Suite. From opening and managing your support tickets to reading knowledgebase articles about the product, the Solver Support Center has everything a user will need.

Users may contact Solver Support if they have questions about the BI360 Suite. One of our technical support consultants will gladly assist you.

Users can access the Solver Knowledgebase for more information about the entire BI360 Suite. From user guides, white papers, training manuals and much more, the Solver Support Center has everything a user will need to get started with the application.

Appendix A: Sample Fact Table Query

The below T-SQL query was used in our second example to generate a dataset based on the **f_Trans_GL** table and related dimension tables. The query includes a number of transformations in order to get the source data into the desired format:

- CAST statements to convert fields that are normally stored as integers, such as the FYYear and FYPeriod columns in the d_Time table, into strings. This prevents the *Power BI* report view from treating them as measures.
- Use of the AS keyword to assign user friendly column labels.
- A CASE statement that assigns a numeric sort order value based on the value in the Account Category column. This allowed us to control the order in which Account Categories were displayed in the bar chart shown in the example.
- A CASE statement that reverses the sign of the Value1 column in certain cases so that credit balances will be displayed as positive numbers.
- A WHERE clause to limit our dataset to P&L accounts and the current and prior fiscal years.

```
/*
Sample Query for f_Trans_GL (GL module) to show P&L Account Balances for Current and
Prior Years
*/

SELECT
CAST(Fact.TimePeriod AS CHAR(8)) AS TimePeriod
,CAST(d_Time.FYYear AS CHAR(4)) AS FYYear
,CAST(d_Time.FYQuarter AS CHAR(2)) AS FYQuarter
,CAST(d_Time.FYPeriod AS CHAR(6)) AS FYPeriod
,CAST(d_Time.FYYear AS CHAR(4)) + ' ' + d_Time.FYQuarterLabel As FYQuarterLabel
```

```
,d_Time.FYMonthLabel + ' ' + CAST(d_Time.FYYear AS CHAR(4)) As FYPeriodLabel
,d_Entity.Code As EntityCode
,d_Entity.Description As EntityDescription
,d_Scenario.Code As ScenarioCode
,d_Account.Code As AccountCode
,d_Account.Description As AccountDescription
,d_Account.AccountCategory
--Adding a sort key for Account Category
,CAST(Case AccountCategory
    When 'Revenue' Then 1
    When 'Cost of Sales' Then 2
    When 'Expenses' Then 3
    When 'Other Income/Expense' Then 4
End AS CHAR(2)) as AccountCategorySort
,d_Category.Code As CategoryCode
,d_Currency.Code As CurrencyCode
,d_Dim0.Code As DeptCode
,d_Dim0.Description As DeptDescription
,Fact.RowComment As RowComment
--Reversing sign of amounts for Revenue accounts which by default display as negative
numbers
,Case When (d_Account.Code Like '4%' And d_Scenario.Code = 'ACT') Then Fact.Value1 *-1
Else Fact.Value1 End As PerioddBalance

FROM f_Trans_GL Fact
--Joins to Dimension tables are made using the MemberId column
LEFT OUTER JOIN d_Account ON Fact.Account = d_Account.MemberId
LEFT OUTER JOIN d_Category ON Fact.Category = d_Category.MemberId
LEFT OUTER JOIN d_Currency ON Fact.Currency = d_Currency.MemberId
LEFT OUTER JOIN d_Dim0 ON Fact.Dim0 = d_Dim0.MemberId
LEFT OUTER JOIN d_Dim1 ON Fact.Dim1 = d_Dim1.MemberId
LEFT OUTER JOIN d_Dim2 ON Fact.Dim2 = d_Dim2.MemberId
LEFT OUTER JOIN d_Entity ON Fact.Entity = d_Entity.MemberId
LEFT OUTER JOIN d_Scenario ON Fact.Scenario = d_Scenario.MemberId
JOIN d_Time on Fact.TimePeriod = d_Time.MemberId

Where
d_Account.AccountType = 'PLC' and
d_Scenario.Code IN ('ACT','BUD') and
d_Time.FYYear >= YEAR(GETDATE()) - 1
```

Appendix B: Additional Code Samples

The below T-SQL queries can be useful in getting information about *BI360DW* modules and fields.

List Attributes for all dimensions used in the GL module

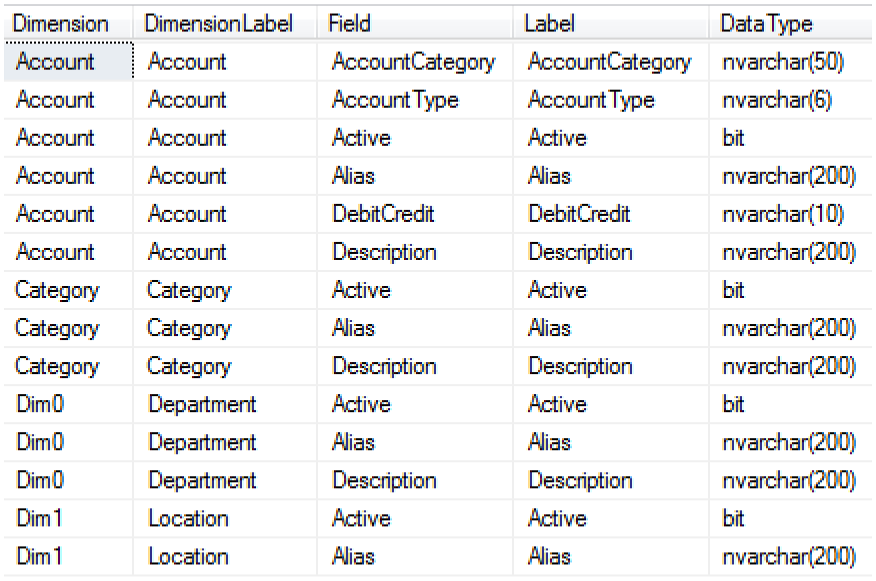
Description: Lists all dimensions linked to a selected module, along with their attributes.

Uses: when considering which tables and fields to include in a view to use with an external BI tool, this query provides a complete listing of dimensions and attributes that are available. Change the 'Where' clause (see highlight below) depending on the source module you're interested in.

Code Sample:

```
Select a.Dimension, b.Label As DimensionLabel, a.Field, a.Label, a.DataType
from DimAttribute a
Left Outer Join DimLabel b on a.Dimension = b.Field
Where b.Module = 'GL'
And b.InUse = 1
```

Sample Output:



Dimension	DimensionLabel	Field	Label	DataType
Account	Account	AccountCategory	AccountCategory	nvarchar(50)
Account	Account	AccountType	AccountType	nvarchar(6)
Account	Account	Active	Active	bit
Account	Account	Alias	Alias	nvarchar(200)
Account	Account	DebitCredit	DebitCredit	nvarchar(10)
Account	Account	Description	Description	nvarchar(200)
Category	Category	Active	Active	bit
Category	Category	Alias	Alias	nvarchar(200)
Category	Category	Description	Description	nvarchar(200)
Dim0	Department	Active	Active	bit
Dim0	Department	Alias	Alias	nvarchar(200)
Dim0	Department	Description	Description	nvarchar(200)
Dim1	Location	Active	Active	bit
Dim1	Location	Alias	Alias	nvarchar(200)

List Attribute Labels for the GL module

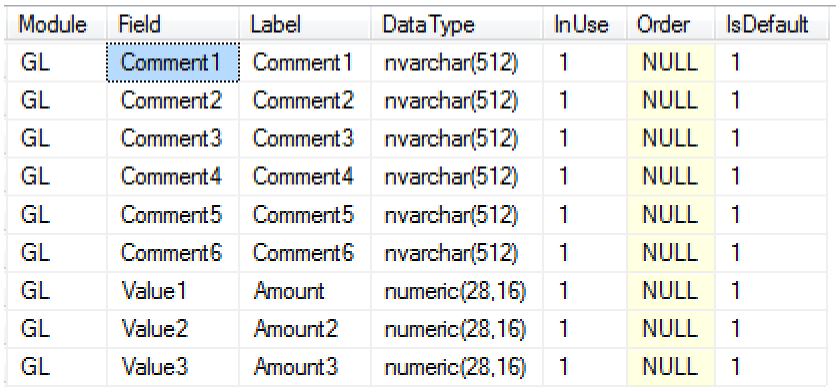
Description: Lists all module attributes, along with their labels and data types.

Uses: Provides a list of module attributes (such as Comments or UDF's) that are currently in use for a selected module. Change the 'Where' clause (see highlight below) depending on the source module you are interested in.

Code Sample:

```
Select * from ModuleAttribute
Where InUse = 1
And Module = 'GL'
```

Sample Output:



Module	Field	Label	DataType	InUse	Order	IsDefault
GL	Comment1	Comment1	nvarchar(512)	1	NULL	1
GL	Comment2	Comment2	nvarchar(512)	1	NULL	1
GL	Comment3	Comment3	nvarchar(512)	1	NULL	1
GL	Comment4	Comment4	nvarchar(512)	1	NULL	1
GL	Comment5	Comment5	nvarchar(512)	1	NULL	1
GL	Comment6	Comment6	nvarchar(512)	1	NULL	1
GL	Value1	Amount	numeric(28,16)	1	NULL	1
GL	Value2	Amount2	numeric(28,16)	1	NULL	1
GL	Value3	Amount3	numeric(28,16)	1	NULL	1

Appendix C: Additional Reading

The following are some of the many available on-line resources that offer tutorials and additional information about *Power BI* features:

Getting Started Tutorials:

Power BI Desktop

https://powerbi.microsoft.com/en-us/documentation/powerbi-desktop-getting-started/

Power BI Web Service

https://powerbi.microsoft.com/en-us/documentation/powerbi-service-get-started/

Power BI Channel on YouTube:

Offering instructional videos and other background material.

https://www.youtube.com/user/mspowerbi

Power BI Gateways:

Personal Gateway

https://powerbi.microsoft.com/en-us/documentation/powerbi-personal-gateway/

Enterprise Gateway

https://powerbi.microsoft.com/en-us/documentation/powerbi-gateway-enterprise/

Sorting by an Alternative Column in Power BI Report View:

The below link provides more details about the alternative sorting used in our second sample dataset.

https://powerbi.microsoft.com/en-us/documentation/powerbi-desktop-sort-by-column/