Data Import Best Practices in Power BI

When you create a data model in Power BI, you should consider how to properly use naming convention and what columns to include, in order to improve usability and performance. This article provides a quick list of best practices valid for both Power BI and Power Pivot.

Not all of the suggestions described can be applied to all of your data models. You should adapt these best practices to your specific scenario, looking at how to achieve the goals that are the reason for a certain pattern more than barely apply it without considering the pros and cons of each choice.

Even if the article mentions Power BI, all the best practices described are valid for Power Pivot and Analysis Services Tabular models, too. The demo file you can download (at the end of the article) contains a sample Power BI file and the corresponding views defined in a SQL file for AdventureWorksDW.

Use views

Always import views and never import tables in a data model.

If you are getting data from a relational database, such as SQL Server or Oracle, you should never import a database table directly in your data model. The reason is that this creates a strong dependency between the physical data model and the report. Over time, certain changes to the database structure might corrupt an existing report. For example, renaming a column or a table, or changing the cardinality of a table, are all operations that require a correspondent change to the Power BI data model.

The real issue is that nobody knows how many reports can be affected by a certain change. Even if a dependencies analysis would be possible from a technical point of view, today we do not have a standard tool and procedure to do that. Creating specific views for each data model corresponds to the introduction of an indirection layer who simplify the change management of the database structure.

The best practice for using views is:

- Create a schema for a certain data model: for example, it could be the name of the data mart, or the name of the group of reports that will share the same data model.
- Create one view for each table you want to create in the Power BI data model within that schema.
- Include in the view only the columns that are useful and will be used in the Power BI data model.
- When you import the tables in Power BI, remove the name of the schema and keep the name of the view only.

By following this best practice, you declare in the database what are the tables and columns used in a report, so that the database administrator is aware of existing dependencies from the database itself. Before publishing in production a change in the database structure, it is possible to adapt these views so that they will continue to work returning the same content, without breaking the refresh of existing reports. Moreover, it is much easier to track dependencies between views and tables in a single relational database. For example, in SQL Server you can use built-in features (such as View the Dependencies of a Table) or third party tools (such as SQL Dependency Tracker from Red Gate).

Keeping all the views for a data model in the same schema simplifies the tracking of the dependent reports. Changing the view to keep compatibility with existing reports is usually a first temporary step. If you modified the database structure, probably you want to reflect this change to the reports, but with a different timing. You will not delay the deployment in production of certain database changes, because you do not have to synchronize the deployment of a new version of all the existing reports. You just have to deploy a compatible version of the views that use the new structure, and notify to the BI analysts who owns the data model that they might use a new version of the data, coordinating with them how to provide the new structure (for example, by changing existing views or by providing different views).

The views created should include an explicit list of columns, and should not be a generic one such as:

```
SELECT * FROM Table
```

The views can include transformation of data. This is important when you want to include business logic that should be shared across different data models, so you do not have to duplicate the same transformation logic in several Power BI data models.

By importing views instead of tables, the data model might not recognize all the existing relationships between tables, because referential integrity constraints are applied to tables and not to views. However, adding the relationships manually to the data model is only a minimal cost

Use meaningful names

The names of both views and columns exposed in the views should be user friendly and identical to names exposed to the users.

You should remove any prefix and any suffix you might use in table names. For example, it is common to see Dim and Fact used as prefixes of tables in a relational star schema. There is no point in showing these prefixes to the user. You should also avoid prefixes of views such as “v” or “vw”. You should show “Customers” instead of “DimCustomers” or “vwCustomers”.

You should avoid abbreviations, prefixes, and suffixes in column names. However, an exception is possible to well-known acronyms. For example, you should use “Sales Amount” instead of “SalesAmt” or “SalesAmount”. You can use space and special characters in column names of a view. The goal is to simplify the life to the user, and not to simplify the life to a programmer who has to type a column name in the keyboard. In Power BI, you have Intellisense when you write a DAX formula.

It is a best practice to rename all the columns in the views, using exactly the names you will expose in the user interface of Power BI. You should avoid renaming tables and columns in a Power BI data model. The reason for that is to simplify maintenance and support, other than being a much more productive way to rename entities. If you rename a column in Power BI, if the user will see some wrong or missing data in a report, he will open a support call by mentioning the entities he knows. If these names are defined only within a Power BI data model, probably the support request will be redirected to the data modeler, who most of the times will open the data model just to find that a certain table in the database does not contains the right data. This happens only because most of the DBAs are not aware of Power BI, or simply do not have access to the data model definition. By moving the renaming in the views, you enable any DBA to analyze dependencies and to better triage the user request, raising the call to another level of support only when the issue is related to a calculation problem and not to a missing update of an underlying table (which can be solved by the DBA itself).

If you are using a schema name to include all the views, remove the schema name from the imported names in Power BI. Unfortunately, there is no automatic way to do that, or to import view names without the schema name, so this is a rename operation you have to do in Power BI.

Use a technique that clearly violates naming convention when you import columns that should be hidden to the user. For example, if you have a star schema and you use surrogate keys, you might use the Key suffix in the column name without any space, for example by using “CustomerKey” instead of “Customer Key”. You might also consider adding a prefix that moves the column name at the beginning of an alphabetical order. For example, you might use “_CustomerKey” instead of “CustomerKey”. In this ways, these names will be at the beginning of the column names list, and it will be easier to check that they are all hidden. However, using a prefix is not a good idea if you want to enable the user to use such a column in transformations and/or joining data to other data sources. Moreover, if the column contains an application key instead of a surrogate key, maybe you want to keep the standard naming convention (using “Customer Key”) to keep the column visible in the data model (so the user can show it in reports).

Avoid ambiguity in names of columns and measures

Expose aggregatable numeric columns in a view using names that cannot be confused with measures, hide these columns in the data model and create explicit measures.

Think at measure names in advance. If you want to present the name “Sales Amount” to users for the SUM of all sales amounts, then you cannot use Sales Amount as a column name, otherwise the engine will refuse to create the measure, as its name conflicts with a column. Using weird names like “Sum of Sales Amount” for a measure is not a good solution. If you plan to aggregate a number versus showing it as it is, then it is better to import it into the model with some naming convention, then hide it and expose it as a measure. For example, you can import SalesAmount as LineAmount (without spaces, so you intentionally violate the rule of having space between words in column and table names), then hide it from the report and define the following visible measure:

```
[Sales Amount] := SUM ( Sales[LineAmount] )
```

If you want to keep the existing underlying data, mark them as “Do not summarize” and expose the column names following the naming convention. For example, use Line Quantity and Unit Price for the visible columns with the original column names OrderQuantity and UnitPrice, and create a measure that correctly sum the product of the two columns line by line:

```
[Sales Amount] := SUMX ( Sales, Sales[Line Quantity] * Sales[Unit Price] )
```

Beware that using measures instead of default aggregations has always been a good practice for many reasons but today, with the new Analyze in Excel feature, it is even more important. In fact, a pivot table in Excel does not have any implicit aggregation, so you rely only on measures defined in the data model in order to compute numbers.

Thus, choose your standard and use it, but do not import columns of metrics as they are in the original table if this is the very same name your user will use for the measure.

Remove useless columns

Do not expose in a view a column that is not necessary in the Power BI data model.

Even if you do not know in advance which columns will be useful in the data model, try to expose only the ones that are necessary. Adding columns later to a view has no side effects, and consider that the Power BI data model will probably have all the columns from a view (it is the default, after all).

By reducing the columns exposed in the view, you reduce the amount of data loaded in memory in Power BI, and more important you avoid to expose high cardinality columns that are only used for technical reasons (such as a timestamp and username for the last change applied to a row in a table).

A lower number of columns means a lower number of dependencies between physical table and reports. You will only pay the cost of maintenance for used columns if the physical model would change in the future.

Split date and time

Do not expose a DATETIME columns, always split it in two columns, one for DATE, and one for TIME. Reduce precision of TIME if necessary down to hour, minute, or seconds, according to business requirements.

High cardinality columns are expensive in Power BI, and a datetime column will likely have a unique value for each row. By splitting that information in date and time you will save memory, you will increase performance, and you will make the data model easier to use. You can do this transformation in a query in Power BI, but doing this in advance in the view will increase productivity using Power BI.

Apply Mark as Date Table to tables with dates

Time intelligence functions require to mark a table as a Date table if a surrogate key column (typically an integer) is used in the relationship between a fact table and a dimension.

Even if this is not required when a relationship is based on a DATE column, it is a best practice to always decorate a date table with the “Mark as Date Table” attribute. This way, UI and other features of Power BI are aware of the table role, improving the user experience.

Before February 2018, Power BI did not have the “Mark as Date Table” attribute. It is a good idea to apply this attribute in models created before February 2018.

Also read Time Intelligence in Power BI Desktop for more information.