Microsoft

Power BI Performance Guidance

Last updated May 04, 2022

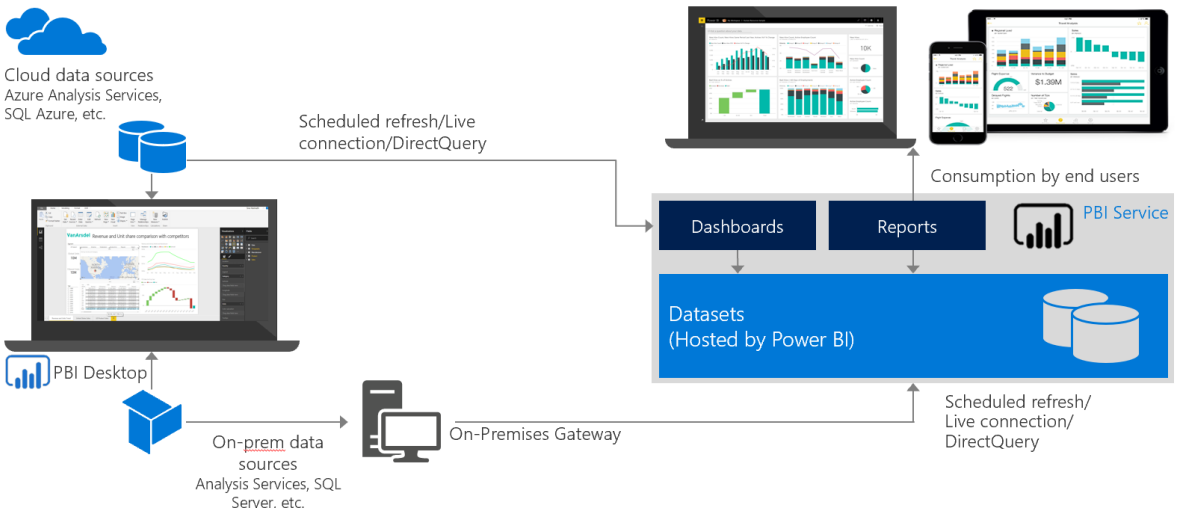
Kathryn Varrall

Contents

Problem Statement

As the popularity and usage of Power BI increases within organisations, it becomes more important to ensure that workloads are monitored with the ability to identify and remediate performance bottlenecks that may occur in Power BI Premium capacity, Power BI Embedded capacity, Power BI Gateways and datasets in general. Performance extends across multiple layers of the architecture when it comes to Power BI and this document is intended to provide overarching end-to-end guidance as far as possible. Additional links are provided in the document for further reading.

What are the Power BI Services



Power BI Desktop

Power BI Desktop is a free application you install on your local computer that lets you connect to, transform, and visualize your data.

Power BI On-Premise Gateway

The on-premises data gateway acts as a bridge to provide quick and secure data transfer between on-premises data (data that isn't in the cloud) and several Microsoft cloud services.

Power BI Service

The Microsoft Power BI *service* (app.powerbi.com), sometimes referred to as Power BI online, is the SaaS (*Software as a Service*) part of Power BI. In the Power BI service, *dashboards* help you keep a finger on the pulse of your business.

Dashboard

Dashboards display *tiles*, which you can select to open *reports* for exploring further. Dashboards and reports connect to *datasets* that bring all of the relevant data together in one place.

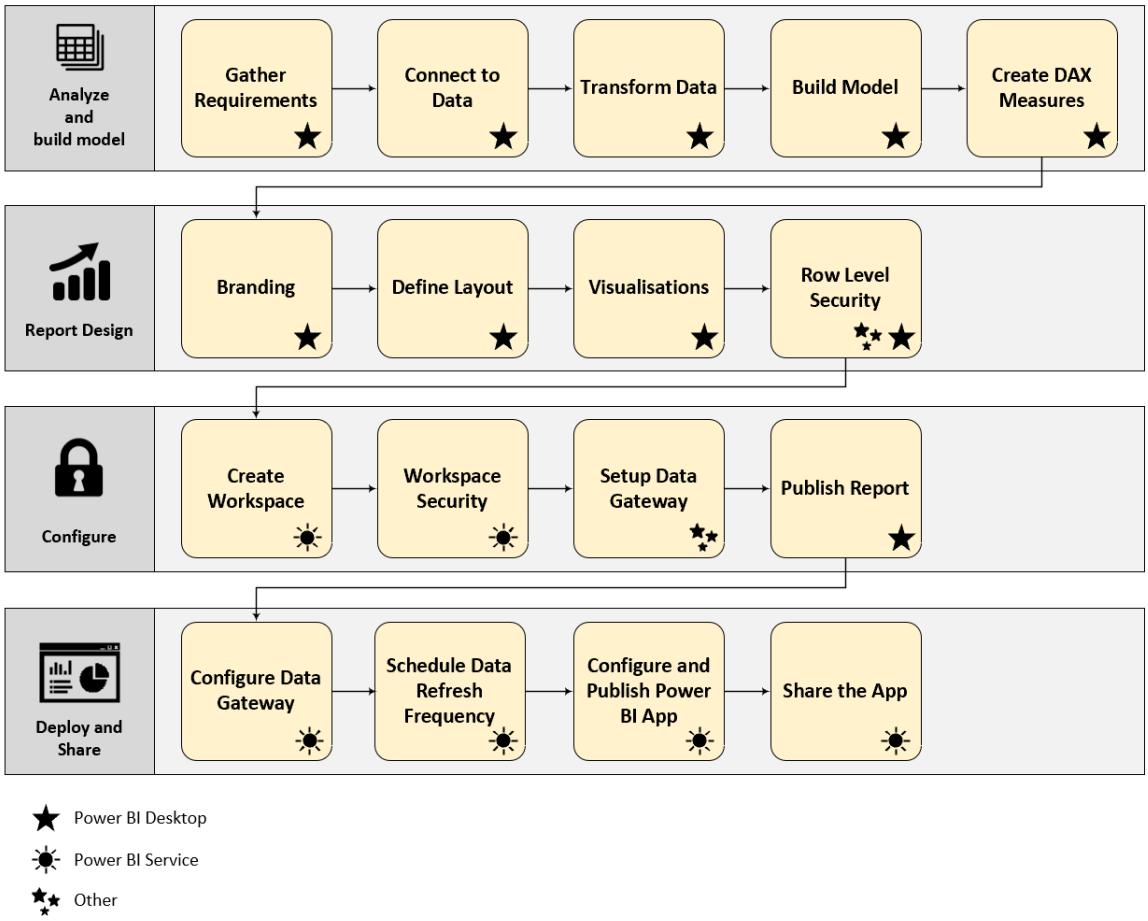
Workspace

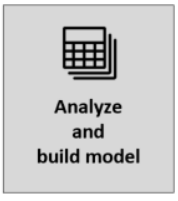
Workspaces are places to collaborate with colleagues to create collections of dashboards, reports, datasets, and paginated reports.

Dataflows

A dataflow is a collection of tables that are created and managed in workspaces in the Power BI service. A table is a set of columns that are used to store data, much like a table within a database. You can add and edit tables in your dataflow, as well as manage data refresh schedules, directly from the workspace in which your dataflow was created.

Development lifecycle





A report developer begins by gathering business requirements which will include determining the source of the data and which columns and metrics are required for the report.

Data sources are identified and the report developer will connect these data source(s) (in the cloud or on-premise) using Power BI Desktop on their local machine. Transformations on the data is applied where applicable such as changing a datatype or column name. A data model is built. Data models are one of the features that allow you to connect to multiple data sources using a relationship. Calculations are applied in the model using DAX to create additional measures or calculations that are applicable to the business case.

A report can be saved from Power BI Desktop to a physical or cloud location. This file is called a .pbix file. A .pbix is a physical file that contains the visuals, the data, dataset and connection information and should be safeguarded. Users can similarly choose to create a .pbit file which is a template file. This file does not keep any data or connection information and when opened, prompts the user for

connection credentials. A refresh of the data typically occurs after a successful connection. This is a great way to templatize Power BI reports.

Branding can be applied to the report and the layout is determined. Visualisations are built using the prepared data model. Finally row level security is applied. before deploying the report to the Power BI Service. Row Level security can either be applied in the Power BI report itself or in Analysis Services depending on the requirement.

To deploy into the Power BI Service a workspace is required. If no workspace is specified, the report will be deployed into the users Personal Workspace.

Report developers will create workspace to deploy the report and dataset in the Power BI Service. Configuration of the workspace is required to give users and developers the relevant access and permissions to the reports and datasets.

If a Gateway is required to access data, further configuration of a Data Gateway is required hosted either on an on-premise virtual machine or an Azure Virtual machine. A data gateway allows Power BI to securely communicate with the data sources.

On completion of development, the developer will publish the report to the workspace.

If the dataset requires the use of a Data Gateway to access data, this will need further configuration in the Power BI Service once the Data Gateway has been setup. Once the Data Gateway has been configured on both the virtual machine and the Power BI Service, a refresh can be scheduled to populate the dataset based on a specific interval (not applicable for direct query).

Reports and datasets can be deployed to a workspace either in Premium or Shared capacity. A user can choose to create an App within the Power BI Service which can be made available to consumers. Users would typically consume an App. If an App is created in Premium Capacity, the user consuming the App would not be required to have a Power BI Pro licence.

Other lifecycle management models may include:

1. **Analysis Services / XMLA Endpoint**
   A user builds a dataset in Analysis Services and deploys the XMLA model to the Power BI Service as an XMLA endpoint (dataset) or to Azure Analysis Services. These datasets still need to be refreshed in order to update the data in the model. The model can either be modified using Visual Studio or directly modified via Tabular Editor. A model can be direct query or in-memory.

2. **Power BI Dataset as a source**
   Users can consume existing datasets which reside in the Power BI Service from Power BI Desktop. In this scenario, a user would not build the dataset (model) but would consume an existing dataset which resides in the Power BI Service. The user would create visuals off the dataset and deploy the report back to the Power BI Service.

3. **Direct Query**

Users can opt to use direct query storage mode. This storage mode connects directly to the source and may traverse a Data Gateway for connectivity. There is no refresh schedule required as data is retrieved directly from the source database on demand. A Page refresh may be configured to keep the information updated especially in the case of reports that are for example are displayed on a monitor.

Power BI as a Self-service Tool for All

There is often a perception that Power BI is a self-service tool which can be given to all users to make use of without consideration. Whilst Power BI is a powerful and valuable self-service tool in any organisation, it is important to consider that governance and monitoring are still key to any Power BI implementation. Lack of governance will result in the organisation quickly running out of capacity and experiencing performance issues. It also raises a significant security risk if users are not securing data, allowing export of data and publishing data to the web.

There should be a balance between giving users the flexibility to create and build their own reports but also ensuring that whatever is deemed a production report adheres to the organisations security and best practice framework.

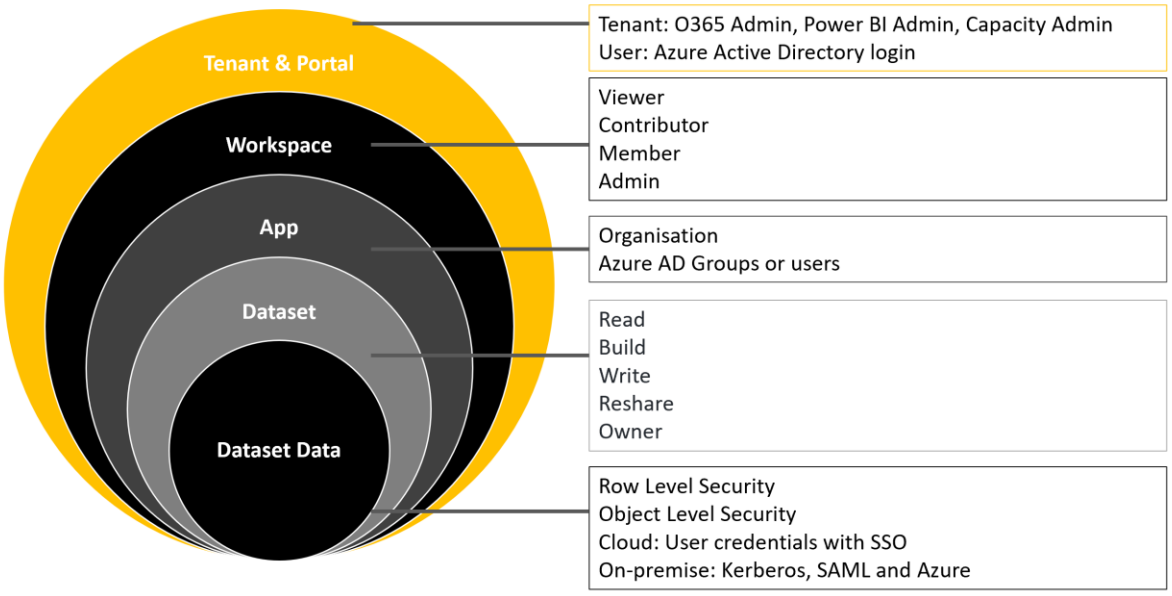
Setting up the Power BI Environment

Detailed guidance for best practice and guidance is provided in the Planning a Power BI Enterprise Deployment Whitepaper.

[Planning a Power BI Enterprise Deployment]

Plan security up front with consideration that security is multi layered in Power BI

[Power BI implementation planning: Security]
[Power BI implementation planning: Tenant-level security planning]
[Power BI implementation planning: Report consumer security planning]
[Power BI implementation planning: Content creator security planning]



Governance

As organisations onboard more users and more reporting use cases, governance becomes increasingly important. Organisations that lack governance may start experiencing increased utilization on their capacity with capacity thresholds (80% to 90%) being reached which often lead to a degraded user and developer experience.

Mistakes that are often made include the following:

- Lack of curation for reports and data models that are deployed to Premium Capacity
- Users and developers have the freedom to deploy models that are potentially sub-optimal. In some cases users may have limited knowledge of data modelling or visualisation techniques
- The Premium Capacity is not being monitored
- In many cases, multiple users have been given Power BI Admin or Power BI Capacity permissions allowing them to freely deploy reports into Premium Capacity and create Gateways with no change control or governance process in place
- There may be a proliferation of reports that are duplicated (one user not knowing that another user created a similar report), there may be little to no concept of Row Level Security and report refreshes often overlap i.e. all users have the flexibility to schedule their own reports so they most likely set up schedules for peek periods such as 7am or 8am so that "their" report is available for the work day -- not taking into consideration or knowing that many other users are doing the same causing contention.
- Too many users having access to a Power BI Pro account or have elevated permissions for Power BI Premium Capacity

Governance is a very broad topic, however the following guidance can be used as a starting point to implement a governance framework. This content will help an organisation achieve the goal of outlining a proper governance strategy in their organisation.

More often than not, teams within an organisation find themselves working in silos as organisations get more complex. In such cases, these teams work without sharing knowledge despite the evolution of their skills.

A Center of excellence (COE) is a team, a shared facility or an entity that provides leadership, best practices, research, support and/or training for a focus area.

A COE allows for sharing of resources which increases efficiency within the organisation whilst creating a more consistent experience for internal and external customers.

A COE can be used to address a wide range of organizational challenges, from department-specific to enterprise-wide. But in the end, perhaps the most compelling reason to consider a COE is that it brings together a cross-functional group of stakeholders from all levels of an organization, giving them the space to focus their talents and expertise on a single process area, business activity, or capability.

Power BI Adoption Roadmap
Power BI Adoption Framework
Establish a Center of Excellence
BI solution architecture in the Center of Excellence

A governance framework should be well documented.

The following should be considered:

- Change control process and curation before deploying Power BI content to the organisations Premium Capacity
- Change control for addition of data sources to gateways
- Change control for addition of new gateways and data sources
- Installation of a data gateway
- The use of dataflows
- The use of templates
- The use of certified\promoted datasets
- The use of paginated reports
- The use of custom visuals within the organisation
- Auditing and monitoring of both the usage logs, Premium capacity and gateway monitoring
- Using Power BI Embedded Capacity (A SKU's) for testing
- CI\CD processes, testing performance and monitoring performance regression
- Create a data loading guide for the organisation
- Creating a metadata catalog for example in Purview
- Securing the data
    - Row Level Security
    - Data classification
    - Sensitivity labels
    - Data privacy settings \ levels

Educate and train users

Once the CoE is established, they should be responsible for educating report developers and users:

This can be done in a number of formal or informal ways including:

- Online training (free) MICROSOFT LEARN FOR POWER BI
- Paid for training by external providers

Report builders\developers should be well versed in data modelling techniques, DAX and be familiar with tools used to identify performance bottlenecks such as DAX Studio and Performance Analyzer in Power BI Desktop. Have one or two champions or subject matter experts in each business unit who are responsible for developing and writing reports. These users should be well versed, not just with the specifics of data and reporting needs within the business unit, but also have a very good understanding and background of correct data modelling techniques as above. Power BI Pro licences should be limited to these SME's and these users may with exception have Power BI Capacity Admin rights (allowing them to deploy and manage Premium Capacity). These SME's should have a good working relationship with the CoE or may be part of the CoE. They should also have a good understanding of the overall business strategy within the business such as Enterprise Data Warehousing, security and data sharing.

Create an internal Teams Channel where users can interact and ask questions.

Additional Online Content

- Basic concepts for designers in the Power BI service
- Datasets in the Power BI service

- Dataset modes in the Power BI service
- 10 Best practice tips to improve your Tabular Model performance
- Optimization guide for Power BI
- Guidance for Power BI
- Understand star schema and the importance for Power BI
- Data reduction techniques for Import modeling
- Introduction to dataflows and self-service data prep
- Use Performance Analyzer to examine report element performance

Performance Guidelines

When evaluating performance, every layer of the architecture must be considered, this includes:

- Gateways
- Network latency
- Storage Mode
- M Query
- DAX calculations
- Visuals
- Reports
- Dashboards
- Workspaces
- Caching
- Direct Query
- Model memory
- Refresh schedules
- Partitioning
- Row Level Security

Performance Baselines

Define a baseline for Power BI Desktop and the Power BI Service and track variation in performance

There can be a variation in performance after deploying from Power BI Desktop to the Power BI Service. This variation in performance can be due to a number of reasons such as activity in the shared / premium capacity (noisy neighbours), eviction of datasets, caching etc.

- Measure changes in Power BI Desktop
- Use Performance Analyzer
- Use the same spec machine when developing
- Set reasonable performance targets
- Consider Real time vs Scheduled (Batch)
- Consider which data needs to be in near / real time and what is acceptable for less frequent updates -- not everything should / can be real time. Always challenge the value of real / near real time data especially for descriptive analytics scenarios.
- Determine the performance for percentile of users i.e. at month-end 75% of report executions will be less than 12 seconds with exceptions

Testing

Build a baseline per report, per model and monitor for regression

Plan and test for scale before going live:

- Use Visual studio web testing Load tests - execute web performance or unit tests to simulate many users accessing a server at the same time.
  Create a load test project
- Load test using the Power BI Capacity Load testing tool
  Power BI Dedicated Capacity Load Assessment Tool
- Be systematic and be data driven when testing
- Use an equivalent Power BI A SKU (A4) in Azure for testing
- Test in PBI Desktop and track % improvement

Guidelines for optimization

There is no silver bullet to performance optimization and performance tuning should be approached collaboratively especially for larger organisations

It is key to evaluate all layers and review the entire architecture. Tracking changes over time will give an indicative benchmark of performance degradation.

Be prepared to that there may be some trade offs - such as putting aggregations in a model to improve performance.

The most effective technique to reduce a model size is to load pre-summarized data. This technique can be used to raise the grain of fact-type tables, however, there is a distinct trade-off, which is loss of detail. Refer to Group by and Summarize for more detail.

Tackle models that are most used

Look at utilizing other features such as:

- Mixed Mode In Power BI Desktop, a Mixed mode design produces a Composite model. Essentially, it allows you to determine storage mode *for each table*. Therefore, each table can have its Storage Mode property set as Import or DirectQuery (Dual is another option).
  *Note security requirements when using Composite models
  Use composite models in Power BI Desktop
- Incremental loads to reduce the amount of data being refreshed
  Incremental refresh in Power BI
- Reduce data for import modelling
  Data reduction techniques for Import modeling

Model Size/Complexity

Power BI model sizes range from 1MB to 1GB -1GB being the limit for a single model/.pbix file today.

Depending on the SKU, Power BI Premium supports uploading Power BI Desktop (.pbix) model files up to a maximum of 10 GB in size. When loaded, the model can then be published to a workspace assigned to a Premium capacity.

How many rows you can fit into a dataset is determined by the size of the data which is in turn compressed. The maximum number of rows ingested may also be determined by data source limitations for example an API you are calling may have a max rows threshold preventing you from ingesting more rows. Users of Power BI Desktop may also face limitations depending on available local memory. For example the dataset is too large to be put into the memory available on the machine hosting Power BI Desktop.

Row limit - When using DirectQuery, Power BI imposes a limit on the query results that are sent to your underlying data source. If the query sent to the data source returns more than one million rows, you see an error and the query fails. Your underlying data can still contain more than one million rows. You're unlikely to run into this limit as most reports aggregate the data into smaller sets of results.

Column limit - The maximum number of columns allowed in a dataset, across all tables in the dataset, is 16,000 columns. This limit applies to the Power BI service and to datasets used in Power BI Desktop. Power BI tracks the number of columns and tables in the dataset in this way, which means the maximum number of columns is 16,000 minus one for each table in the dataset.

Data update frequency

How often does the data need to be refreshed? Every day, every hour or minute. With Cloud models you can only update up to 8x per day and 48 x a day if you have Premium Capacity, If there is a requirement that data is more fresh, then consider using DirectQuery, Live Connection or use the REST API's to refresh models.

Active concurrent users

Approximately how many users are actively clicking on reports/dashboards at the same time. E.g. 10,000 users connected to a dashboard, but only 10 are actively clicking, which are triggering queries.

Concurrency limits are defined primarily by the **number of queries** against a dataset (and not the **number of users**)

Query latency

This is the advantage for Cloud models. how long does it take to query the model and/or data source to get the visuals? Number of hops, how long are the hops, in-memory or Direct query. On-premise data requires many more queries which takes more time.

- Look at storage mode and consider network latency
- Choose the appropriate storage mode
  Dataset modes in the Power BI service
- Reduce network latency

Network latency
- Understand and be aware of usage patterns -- factors can be multiplicative

What is a hop?

A hop is a term that refers to the number of routers that a packet (a portion of data) passes through from its source to its destination.

Every time packets flow from one computer or device to another, like from your computer to a website and back again (i.e., viewing a web page), a number of intermediate devices, like routers, are involved.

That processing-and-passing-along process takes time. More and more of that happening (i.e., more and more hops) adds up to more and more time, potentially slowing down your experience as the hop count increases.

Storage mode – Import vs Direct Query

Import should be always be the first choice (all data is in-memory which gives you the best speed with no DAX limits.

Imported data is always stored to disk. When queried or refreshed, the data must be fully loaded into memory of the Power BI capacity. Once in memory, Import models can then achieve very fast query results.

When refreshed, data is compressed and optimized and then stored to disk by the VertiPaq storage engine. When loaded from disk into memory, it's possible to see 10x compression. So, it's reasonable to expect that 10 GB of source data can compress to about 1 GB in size. Storage size on disk can achieve a 20% reduction from the compressed size. The difference in size can be determined by comparing the Power BI Desktop file size with the Task Manager memory usage of the file.

However there are some points to consider:
- Can you import this data set into Power BI, without query latency?
- Can you create a connection to AAS/SSAS? AAS/SSAS can cater to higher concurrency needs and provide more frequent data refreshes and large model requirements especially if Premium is not an option and can be significantly optimized.

Consider DirectQuery, if the import mode options do not work for you or if you have near real time requirements

You should not consider import mode if:
- There are extremely large volumes of data
- Near real time access to data is required from the source
- Considering you have an existing investment OLAP or SSAS
- There are regulatory and data sovereignty requirements

Tips

- Evaluate how much source data and how compressible is the source data -- expected compression can be 5x to 10x
- Is Premium an option which has support for large datasets
- Will a blended architecture suffice (composite models, aggregations for summary data)

- If direct query is a considering, has consideration been taken for the limits on DAX when using Direct Query mode (i.e. time intelligence)

Live Connect

All business logic is captured in the semantic model and made available for all reporting tools. Therefore, Power BI Desktop turns into a report authoring tool only and you cannot create any new business logic in the report. You can create report level measures, however, bear in mind that these are retained within the Power BI Report and are not stored in the underlying model, basically all the business logic resides in Analysis Services.

Live Connect also refers to establishing a live connection to a shared dataset in the *Power BI service*, and creating many different reports from the same dataset.

Connect to datasets in the Power BI service from Power BI Desktop

Tips

- Use Live Connect when the organisation has invested significantly in Analysis Services and have mature models
- Where a high level of control is needed over partitions, data refresh, query scale-out and workload splitting
- Customer does not have Premium Capacity
- If you are planning to host or migrate to Azure Analysis Services, consideration should be taken regarding the automated management and loading of partitions and can be achieved by using Azure functions (TOM \ AMO libraries) or invoking the REST API's using Logic Apps or Azure Data Factory

New features have become available such as DirectQuery for Power BI datasets and Azure Analysis Services. This is ideal for report authors who want to combine the data from their enterprise semantic model with other data they may own like an Excel spreadsheet, or who want to personalize or enrich the metadata from their enterprise semantic model.

DirectQuery for Power BI datasets and Azure Analysis Services

Plan Data and Cache refresh schedules

Do not refresh data / cache when the source hasn't changed It is a waste of resources refreshing data/caches needlessly and this can affect user experience due to concurrency and evictions Scheduled refreshes need at least 2 x model memory therefore memory pressures can occur when running parallel refreshes.

Refreshing a data set can also be particularly memory intensive, and the memory usage during a refresh could increase by more than double what might be normally required. This has an impact on normal operations during refresh, and means that capacity needs to be over-sized to accommodate refresh in some cases.

Power BI Premium Gen2 will resolve for this by abstracting the nodes for interactive reporting and refreshes. The Query able model is stored in memory and a second copy is refreshed in the background therefore:

- Be mindful of large refreshes running in parallel
- Slow sources can take a while to return data

Tips

- Align refresh schedules to eliminate extra processing
- Stagger parallel refreshes or alternatively use the scale up capabilities
- Consider implementing incremental loads to reduce the amount of data that is refreshed

Reduce network latency

Check for components in data pipeline are not separated geographically or in disparate networks (data sources, gateways, Power BI tenant) as this increases data latency and throughput is reduced

For DirectQuery, the load is on your organizational backend.

Query latency factors to consider include:

- Number of hops
- How long are the hops

Tips

- Co-locate services (at least for Production)
- Have cloud replicas of on premise sources
- Development can be performed on a VM in the same region as data sources
- Use Azure Express Route to more tightly control how the network traffic flows between your on-premises infrastructure and the Microsoft infrastructure that is hosting Azure
- Use Azure Speed test to tell you the latency between the machine and instances of Azure set up
- If network latency is your problem, consider setting up a VM closer to where the gateway is
- Focus on latency between Power BI Service and the underlying data source
- Use Power BI Desktop to gauge network latency as it sits on premises, close to the on-premises data sources

Run critical workloads in appropriate capacity

- Consider if refresh or report performance is highly variable - many datasets in the capacity and used concurrently

Tips

- Consider different Premium Nodes closer to the user population or consider multi geo capabilities available for Power BI Premium especially if you have geographically dispersed users
  Configure Multi-Geo support for Power BI Premium
- Datasets and reports that are highly used with known business peak hours and specific times of the month should be moved into the appropriate workspaces

Gateways

Having direct query and scheduled refreshes on the same gateway can create queuing and resource contention during scheduled refreshes

Gateways that have combined workloads cannot be optimized for specific times and scenarios

Tips

- Have a dedicated gateway for Direct Query and one for Scheduled Refreshes
- Use an Enterprise GW with minimum 8 cores and 16 GB of RAM
- Monitor gateways
- Scale up or out if performance counters show pressure
- Do not use the personal gateway
- Ensure the machine has a reliable network connection

Caching

Query caching can speed up reports associated with a dataset.

Query caching instructs the Premium/Embedded capacity to use its local caching service to maintain query results, avoiding having the underlying data source compute those results. Query caching can reduce load on your Premium/Embedded capacity by reducing the overall number of queries.

Cached query results are specific to user and dataset context and always respect security rules The service only does query caching for the initial page that you land on. When the query cache is refreshed, Power BI must run queries against the underlying data models to get the latest results. If a large number of datasets have query caching enabled and the Premium/Embedded capacity is under heavy load, some performance degradation may occur during cache refresh. Degradation results from the increased volume of queries being executed.

Query caching is only available on Power BI Premium or Power BI Embedded. It is not applicable to LiveConnect datasets leveraging Azure Analysis Services or SQL Server Analysis Services.

When data connections are made to an RLS-capable data source, such as an Analysis Services data source, only dashboard data is cached in Power BI.

Power BI M Query

Sub-optimal sort ordering and configuration can slow down development in Power BI Desktop and refresh once published

Complex or inefficient Power Query uses more memory during a refresh

Many tables / parameterized queries can cause slow / failed refresh because they are waiting in Power Query for CPU/Memory

Privacy settings may use data from one source to filter another. The default privacy settings prevent data from being leaked between sources.

If queries are not folded, the entire table may need to be loaded into memory before moving to the next step

Doing a merge in Query editor across 2 large tables (millions of rows) will load the entire table into memory to perform the join

Models may contain M queries that are only used in intermediate tables in other queries. These tables are usually hidden to prevent user access which is not secure and consumes memory but the table is never touched interactively Intermediate objects that are iterated by later steps i.e. sort then rank by position) or used multiple times are much slower and use more memory and CPU

Auto Date/Time is used to create date time hierarchies on many date columns with broad ranges, the default setting for Auto Date/Time is On and the system uses the 1/1/1900 or similar for a blank date. Auto created Date / Time tables are not visible to the developer or user and cannot be edited. Auto Date / Time can create many calculated columns and calculated tables in the model which can have a significant impact on model memory and size

Tips

- Push work to the data source including joining at the source Ensure your queries are folding. Query folding allows some transforms to be converted to a native source language. Operations are pushed down to source and can be much faster Ensure all foldable steps happen first, together such as grouping/aggregations/merges performed in M. Sources include relational sources, SQL, Oracle etc. OData Sources, Active Directory, Hadoop (HDFS) Operations include filters, joins, aggregates / grouping (un)pivot,numeric calcs, simple transforms such as uppercase
- Spread memory / CPU intensive operations out
- Disable the following in Power Query Editor Options Background Data -- which allows data preview to download in the background
- Use parameter flags to return smaller subsets of data when still editing queries
- If safe disable privacy or set sources to organisational
  Data Privacy Settings In Power BI Power Query Part 1 Performance Implications
- Avoid Grouping/aggregations/merges performed in M Consider DAX measures instead to replace groupings/aggregations/merge performed in M or customize the source query using the Advanced Editor
- For merge across large tables, rather join this at the source using a custom query and then manipulate further in Query editor
- Columns that are grouped is a contiguous source (distinct values appear together) and data loading with better local processing can be sped up by using GroupKind.Local using a custom M query in advanced editor - add a 4'th parameter to the Group
  GroupKind.Local runs much faster for large datasets than the default-setting. So if you are sure that your data will always be sorted accordingly, this can speed up your grouping-operations considerably.

  Example

Days employee worked

Date	Day Of Week	Activity
01/01/2014	Wednesday	Vacation
02/01/2014	Thursday	Vacation
03/01/2014	Friday	Vacation
04/01/2014	Saturday	Weekend
05/01/2014	Sunday	Weekend
06/01/2014	Monday	Vacation
07/01/2014	Tuesday	Working
08/01/2014	Wednesday	Working
09/01/2014	Thursday	Working
10/01/2014	Friday	Working
11/01/2014	Saturday	Weekend
12/01/2014	Sunday	Weekend
13/01/2014	Monday	Working
14/01/2014	Tuesday	Working
15/01/2014	Wednesday	Working
16/01/2014	Thursday	Working
17/01/2014	Friday	Sick
18/01/2014	Saturday	Weekend
19/01/2014	Sunday	Weekend
20/01/2014	Monday	Sick
21/01/2014	Tuesday	Sick
22/01/2014	Wednesday	Working
23/01/2014	Thursday	Working
24/01/2014	Friday	Sick
25/01/2014	Saturday	Weekend
26/01/2014	Sunday	Weekend
27/01/2014	Monday	Sick
28/01/2014	Tuesday	Working
29/01/2014	Wednesday	Working
30/01/2014	Thursday	Working
31/01/2014	Friday	Working

Simple Group By

	A	B
1	Activity	Count of Days
2	Vacation	4
3	Weekend	8
4	Working	14
5	Sick	5

GroupKind.Local

	A	B	C	D
1	Activity	Start Date	End Date	Number of Days
2	Vacation	01/01/2014	06/01/2014	4
3	Working	07/01/2014	16/01/2014	8
4	Sick	17/01/2014	21/01/2014	3
5	Working	22/01/2014	23/01/2014	2
6	Sick	24/01/2014	27/01/2014	2
7	Working	28/01/2014	31/01/2014	4

GroupKind.Local in this example produces one row for each consecutive range of days (ignoring weekends) spent either on vacation, working or off sick.

The GroupKind.Local is an optional parameter and is the key to getting the desired behaviour as depicted in the example.
The default type of grouping, GroupKind.Global, does a standard group by across the whole table. GroupKind.Local on the other hand aggregates only over consecutive sequences of rows, and this means you see three separate time ranges for the activity “Working” and two separate groups for “Sick”.

Further explanation can be found here: [Aggregating By Local Groups In Power Query]

- Disable the loading of intermediate tables that are hidden from users in the query editor to reduce memory consumption
- For intermediate tables, edit M manually and use the {Table | List}. Buffer functions to store the intermediate table in memory
- Turn off Auto Create Date/Time in Power BI Desktop

Direct Query

Direct Query issues parallel queries for each interaction that affects the report results

Sources may be overwhelmed by many/expensive queries and the gateway may also be affected

Query predicates won't be folded unless in a specific order.

When the Direct Query source PK/FK is enforced or known to have referential integrity but the default "Assume Referential Integrity" setting is off, then OUTER JOINS are used instead of INNER JOINS which is slower

When adding relationships on calculated columns, the joins are not pushed down to the source and need to be computed on the fly in Power BI which means it could potentially fetch all data

When the source is heavily loaded and performance testing and profiling shows higher concurrency, this could be because Analysis Services currently has a limit of 10 connections per data source object (10 concurrent queries). With many concurrent queries, the queries can get queued in Power BI before reaching the source.

Tips

- Monitor performance in desktop and ensure that it is reasonable (under 5-10 seconds). Performance will not get better in the service than it was in desktop.
- Exploit the data source strengths
- Reduce concurrency and queuing
- Reduce no./size of queries to benefit direct query
- Push transformations to the direct query source
- Avoid transformations in M
- Ensure all foldable operations are performed together first
- Write native code to source i.e. TSQL
- Turn "Assume Referential Integrity" setting to On in the case of Direct Query where there is known referential integrity in the source
- Add computed columns to the source table and create materialized views in the source to avoid having to create relationships in Power BI
- Profile and log incoming queries at the source
- Look at execution plans and statistics and consider the following on the source: Indexes Column storage technology Materialized views In-Memory technology Read-only reporting replicas

Scaling up/out

Be aware of data source connection limit for direct query sources and optimize all aspects of the report to avoid concurrency issues

Monitor gateways to ensure contention and bottlenecks are not occurring on the GW. Scale GW's out / up if required

Understand direct query specific constraints for example data sources that or don't support direct query, lack of time intelligence, limited DAX functions, no bi- directional filtering

Ensure your backend scales for concurrent usage

Limit parallel queries You can set the maximum number of connections Direct Query opens for each underlying data source. It controls the number of queries concurrently sent to the data source

Do not use Power Query relative date filtering This type of filter translates to an inefficient native query.

Whenever RLS filters are enforced on Direct Query tables and there are relationships to other Direct Query tables, be sure to optimize the source database

Report Visuals

Visuals generate queries against source data

There is a minimum of 1 query per visual and queries are sent in parallel

When creating visuals it should always be top of mind to reduce the number and complexity of queries and reduce the amount of data being returned to the report

A high number of visuals is considered 20+ An optimal number of visuals is 6-10 per report page:

- Inappropriate use of visuals/slicers can generate a large amount of queries and slow things down.
- Many visuals generate many queries and may overwhelm the source and create client contention
- Cross highlighting filters every other visual by default and generates many queries
- Table / Matrix showing many 100's ++ of rows causes high memory load
- Some custom visuals are poorly optimized and can be slow causing a sub optimal user experience
- Slicers in reports that contain thousands of values causes a high memory load especially with multiple interacting slicers
- Unfiltered visuals containing a high number of values cause high memory load and have more data to fetch and process
- Synced slicers on reports with many pages and many visuals cause a higher memory load and more queries
- Table visuals containing many fields and complex measures that are exported by a user create leaf level queries that are expensive and can consume a lot of memory especially in larger models
- Scatter charts with more than 3500 points applies high density sampling by default and can take some time in certain combinations

Tips

- Reduce the number of visuals on a page
- The landing page should be a high-level aggregation of key visuals which allow the user to then to drill down on key points of interest using drill down capabilities to limit the context
- Avoid visuals that return 100's + rows -- consider using a Top N in this scenario and provide a drill through to provided a detailed view with limited context
- Use default filters instead of slicers especially when slicers will contain a high number of values Slicers issue two queries -- 1 to populate and 1 to fetch selectiom details Use a filter instead to force context and limit values Filter pane only executes the queries to populate the fields when you open the filter pane so initial view of the report does not run all the queries to populate the slicers
- Consider disabling meaningless cross filters by selecting the visual and turning interactions off for these visuals
- Replace poor performing custom visuals with built in visuals or use certified custom visuals only.
- Test the performance of visuals in isolation
- Set default values of slicers and set the single/multi-select property and restructure the report to provide drill through to detail
- Consider no cross-highlighting and adding apply buttons to slicers and filters
- Limit export of data at design time
- Combine visuals -- for example instead of having three card visuals, each issuing an individual query, combine this into a matrix visual which will issue 1 query
- Avoid "data dump"-style reports with tables containing hundreds of columns and thousands of rows. This is usually created when the user expects to export this data to Excel. If the user intends to export data to Excel, consider designing the report directly within Excel to provide the data required but with a live connection to the dataset
- Consider using techniques such as bookmarks, drill through pages and tooltips to reduce the amount of data displayed on a page, also consider adding an "Apply filters" button to your

report, so that your visuals don't update automatically every time you change a filter in a page (especially useful for Direct Query!)

Dashboards

Provide a fast summary view with drill to report

Pinning a report as a tile is executed on demand and will not be cached therefore it is slower and will negatively impact the dashboard load experience

Dashboards using direct query and RLS will by default refresh every hour and may overload the source

Tips

- Use Query cache where possible
- Employ a "launch pad" design principle
  A launch pad serves as the foundation from which the rest of your solution is built.
  In a reporting context, it means report development should be nimble, adaptable and agile. Instead of trying to complete everything at once, growth-driven design works in stages. It's agile, in-tune with the needs of the business and completely effective. Once you launch, you move into a new stage of growth-driven design.
- Always pin individual visuals instead of the report page
- Design metrics and dashboards to meet users information needs
- For direct query dashboard tiles using row level security consider reducing the cache refresh frequency by setting the scheduled cache refresh

Data modelling, DAX and Relationships

Tips

- Aim for a star schema and avoid M2M - Avoid snowflake schema architecture.
- Avoid relationships on calculated columns. This can cause the calculation expression to be embedded into the source query which is inefficient, and can commonly prevent the use of indexes
- Ensure tables have relationships
- Validate and Use Inactive Relationships purposefully
- Avoid bi-directional relationships against high-cardinality columns
- Avoid excessive bi-directional, many-to-many relationships and cross filtering
- Many-to-many relationships should be single direction
- Relationship columns should ideally be of integer data type
- Avoid using the LOOKUPVALUE DAX function when model relationships could achieve the same result
- Avoid complex RLS rules rather rely on well-designed relationships to ensure RLS filters propagate to other model tables
- In general, it's often more efficient to enforce RLS filters on dimension-type tables, and not fact-type tables
- Only include columns that are required
- Avoid fields with high precision and cardinality i.e. date/time split or reduction in decimal places
- Use calculated measures rather than calculated columns when possible

- Reduce usage of calculated columns and calculated tables
- Do not use floating point data types
- For very large datasets consider creating a subset of the model for the most common scenarios
- Large tables should be partitioned
- Reduce usage of long-length columns with high cardinality i.e. a field containing comments
- Set IsAvailableInMdx to false on non-attribute columns (Turn off column hierarchies IsAvailableInMDX column property)
  How The New IsAvailableInMDX Property For Analysis Services Tabular Can Reduce Memory Usage And Speed Up Processing - Chris Webb
- Filter is a row by row operation so use wisely
- Try to avoid DAX iterator functions (e.g. sumx, averagex...)
- Measures should not be direct references of other measures, meaning avoid duplicate the same measure with different names.
- Reduce usage of calculated columns that use the RELATED function.
  The **RELATED** function can only be used in a calculated column expression where the row context is clear or as a nested function in an expression that uses a table scanning function like SUMX().
- Use Divide function vs Divide Operator The DIVIDE function was designed to automatically handle division by zero cases. If an alternate result is not passed in, and the denominator is zero or BLANK, the function returns BLANK. When an alternate result is passed in, it's returned instead of BLANK.
  DAX: DIVIDE function vs divide operator
- Error Functions: It's better to avoid using the ISERROR and IFERROR functions. Instead, apply defensive strategies when developing the model and writing expressions
  DAX: Appropriate use of error functions
- SELECTEDVALUE achieves the same outcome as HASONEVALUE and VALUES more efficiently
  DAX: Use SELECTEDVALUE instead of VALUES
- Use COUNTROWS instead of COUNT as it performs better and is more efficient
  DAX: Use COUNTROWS instead of COUNT
- Compound expressions can involve the use of many nested functions, and possibly the reuse of expression logic. Use variables to improve performance and readability Variables are a construct in PBI. Use variables even for simple measures
- Variables are measure bound and can only be used locally in a measure
- Variables can store single values, columns or tables
- Variables help significantly in improving performance
  Note: you can only return a single value in a measure
  In this example:
  Margin = IF([Sales] > 0 && [Cost] > 0 , [Sales]-[Cost])
  Executes each time you reference the measure and goes to the engine and calculates the measure. In this scenario, measures will be evaluated twice because of the IF function
  Using Variables
  Margin = var Sales = [Sales] var Cost = [Cost] return IF(Sales > 0 && Cost > 0, Sales - Cost)
  Calculates the measure twice and puts them into variable and uses the reference values rather. In this scenario, measures will be evaluated once for each IF variable used, resulting in better query performance.
  DAX: Use variables to improve your formulas
- It is recommended that your measures return BLANK when a meaningful value cannot be returned.
  DAX: Avoid converting BLANKs to values
- Always use fully qualified column references and never use fully qualified measure references

Use Table[ColumnName] for column references and [MeasureName] for measure references (for column reference, be sure to put the table name before column name enclosed by []).
DAX: Column and measure references

- Use Boolean expressions as filter arguments, whenever possible
  Avoid using FILTER as a filter argument
- Data Analysis Expressions (DAX) Reference.
- 4 recommended practices for new DAX users
- Worry about the important functions first Focus on the 9 key functions that you are most likely to use SUM, AVERAGE, MIN, MAX, COUNT, COUNTROWS, CALCULATE, FiLTER, IF
- Focus on understanding concepts and patterns
- Keep your functions readable Keep internal column names hidden, rename calculated columns and measure to define user-friendly names
- Use explicit measures where possible - Simply put, implicit measures are measures that are automatically assigned an aggregation such as a Sum or a Count by Power BI
- Sort measures by category

Auditing

These are the key power shell cmdlets you can use to gather auditing information for Power BI. With the Powershell scripts you can convert to CSV and build a report from the CSV files.

Automate the scripts to build out a report that gets refreshed on a regular basis.

PowerBI-Powershell Examples
Microsoft Power BI Cmdlets for Windows PowerShell and PowerShell Core

Admin modules must be installed before using Powershell scripts to gather metrics

- MicrosoftPowerBIMgmt
- AzureAD

Pre-requisites:

- Windows PowerShell v5.0 and up with .NET 4.7.1 or above
- 'MSOLAP' provider should be registered on the local machine running the PowerShell scripts for querying Power BI Premium Metrics dataset using DAX (Powershell)
- Must be a Power BI Service Administrator
- Ensure internet connectivity and access to Power BI Service
- Run scripts close to PBI Environment to avoid latency
- Minimum Standard DS3 v2 (4 vcpus, 14 GiB memory)
- To collect data from the Power BI Premium Capacity metrics, you must install the Premium Capacity metrics app in advance.
- The process will then extract metrics from the dataset using DAX and must be populated with values
- Audit logs need to be enabled in Power BI and may take up to 48 hours for events to populate

Access your audit logs
Login to Power BI
Get-PowerBIAccessToken
Connect-PowerBIServiceAccount
Get-PowerBIActivityEvent

Retrieves the audit activity events for the calling user's tenant. Before you run this command, make sure you log in using Connect-PowerBIServiceAccount. This cmdlet requires the calling user to be a tenant administrator of the Power BI service.

Reference

- You must be granted the Power BI Admin role.
- You must select a timeframe within a single day. You cannot pick a 24-hour period which spans two different days.
- You have to use a very specific date format: in UTC format and ISO 8601 compliant.
- Output the as either a JSON String or a JSON object - convert to CSV for reporting

Introducing the Power BI Activity Log
Access the Power BI activity log

To export Power BI Objects Use the: Invoke-PowerBIRestMethod to get inventory information for the following:

List

- Workspaces
- Reports
- Dashboards

Iterate

- Dataflows
    - currentDataflowId
    - currentWorkspaceId
    - dataflowDatasources
    - dfdatasources
- Datasets
    - DatasetID
    - DatasetName
    - WorkspaceID
- Refresh History
    - id
    - refreshType
    - startTime
    - endTime
    - serviceExceptionJson
    - status
    - DatasetID
    - WorkspaceID
    - DatasetName
- Users
    - currentWorkspaceId
    - members
    - users
    - currentWorkspaceId
    - emailAddress

    - groupUserAccessRight
    - identifier
    - displayName
    - principalType
- Datasources from Dataflows and Datasets
    - currentDatasetId
    - currentWorkspaceId
    - datasetDatasources
    - datasourcesId
- Tenant settings
    - switchId
    - switchName
    - isEnabled
    - isGranular
    - allowedSecurityGroups
    - deniedSecurityGroups
- Embed Codes
    - snapshotName
    - workspaceName
    - publisherUserName
    - disabledByUser
    - createDate
    - lastRefreshTime
- Gateways
    - clusterName
    - type
    - cloudDatasourceRefresh
    - customConnectors
    - version
    - status
    - versionStatus
    - contactInformation
    - machine
    - nodeId

As an alternative the following can be used for some of the inventory objects individually:

- Get-PowerBIWorkspace
- Get-PowerBIReport
- Get-PowerBIDataset
- Get-PowerBIDashboard
- Get-PowerBIDataflow
- Get-PowerBIDataflowDatasource
- Get-PowerBIDatasource

Export organisational licences

- Get-AzureADSubscribedSku Export users with pro licences -- filter on Power BI pro
- Get-AzureADUserLicenseDetail

Power BI Premium Gen2

Operations will always perform at top speed and won't slow down when the load on the capacity approaches the capacity limits. There are no limits on refresh concurrency, no longer requiring you to track schedules for datasets being refreshed on your capacity There are fewer memory restrictions Complete separation between report interaction and scheduled refreshes

Improved metrics

Clear and normalized capacity utilization data, that's dependent only on the complexity of analytics operations the capacity performs, and not on its size, the level of load on the system while performing analytics, or other factor.

Autoscale

Allows for automatically adding one v-core at a time for 24-hour periods when the load on the capacity exceeds its limits, preventing slowdowns caused by overload. V-cores are automatically removed when idle time is detected. Additional v-cores are charged to your Azure subscription on a pay-as-you-go basis. You can opt in or out of Autoscale You can limit the max number of cores to Autoscale to

Evaluate Gen2 capacity load such as *overload* and *autoscale*.
Understanding Power BI Premium Gen2

Partitioning

When to partition:

- Large fact tables should be partitioned
- When a full data cannot be loaded within the specified amount of time
- Power BI incremental loads partition tables transparently
- If fine grained control is required then it is recommended to partition the tabular model using TOM and AMO libraries to build out and automate tabular partitions both in Analysis Services and XMLA endpoint models hosted in Power BI

Available options to create partitions:

- PowerShell commandlets - code intensive PowerShell can use the Tabular Model Scripting Language (TMSL) where the objects are defined using JSON format PowerShell commandletscan be used for processing and managing of partitions Requires modules:
    - Az.AnalysisServices
    - Sqlserver
    - Ssashelper

To automate:

- Custom activity in ADF using a
    - Batch account: note: takes time to instantiate a Batch job (it would not be conducive to instantiate this every 5 min)
    - Azure Automation runbook: PowerShell scripts can run in an Automation Account, assign the webhook at do a post in Azure Data Factory
- Azure Functions - code intensive Use AMO and TOM libraries to manage and process the partitions. To automate Azure Functions, you can use the Azure Function Activity in Azure Data Factory

Azure Function Activity

- SSIS - code intensive Use a C# script within SSIS that will allow you to generate the XMLA script and add variables so that it can be dynamic. SSIS Analysis Services Execute DDL task to execute XMLA. Use the SSIS Analysis Services Execute DDL task to execute XMLA To automate: - Automate the job using SSIS in Azure Data Factory - Requires SSIS Integration runtime and Azure SQL database to host the SSIS DB Catalog to execute SSIS packages in ADF.
  The SSIS-IR can be paused and resumed, however, due to the fact that this can take time, it is better not to use this in scenarios where you will need to pause and resume often or frequently.
- Logic Apps Use the Azure Analysis Services Rest API and post The Rest API is aimed at processing and limited with regards to partition management functionality.
  Management i.e. creating or deleting partitions can be done using Azure Function or PowerShell

Tools

The following tools can be used utilized to evaluate performance:

Power BI Desktop Performance Analyzer

Power BI Desktop Performance Analyzer inspects and displays the duration necessary for updating or refreshing all visuals that user interactions initiate, and presents the information so you can view, drill down, or export the results. Performance Analyzer can help you identify visuals that are impacting the performance of your reports, and identify the reason for the impact.

Use Performance Analyzer to examine report element performance

Dax Studio

Dax Studio Stand-alone Product or accessible via Power BI Desktop Used to execute and analyze DAX queries against Microsoft Tabular models (Tabular modeling overview)

DAX Studio

Vertipaq Analyzer

Vertipaq Analyzer Stand-alone Product – Excel Based Analysis of models requires extract of VPAX file from DAX Studio VertiPaq Analyzer is useful to analyze VertiPaq storage structures for a data model in Power BI and Analysis Services Tabular

Vertipaq Analyzer

Tabular Editor

Tabular Editor Stand-alone Product or accessible via Power BI Desktop Tabular Editor is an editor alternative to SSDT for authoring Tabular models for Analysis Services even without a workspace server. The Tabular Editor is an open source project that can edit a BIM file without accessing to any data.

Tabular Editor 2

Tabular Editor 3

Best Practice Analyzer

Best Practice Analyzer A best practice rules engine which forms part of Tabular Editor

Best practice rules to improve your model's performance
Best Practice Analyzer | Tabular Editor Documentation

ALM Toolkit

ALM Toolkit Stand-alone Product or accessible via Power BI Desktop BISM Normalizer is a Visual Studio extension that works with Azure Analysis Services (Azure AS) and SQL Server Analysis Services (SSAS) tabular models. It allows a modeler to perform metadata merging across models. All tabular model objects are supported. As a Visual Studio extension, it is tightly integrated with source control and model management workflows

ALM Toolkit
BismNormalizer

Register for free training by SQLBI on Dax Tools including, Vertipaq Analyzer, DAX Studio and Analyze in Excel.
DAX Tools Video Course - SQLBI

How to use the Vertipaq Analyzer

Vertipaq Analyzer is a key tool to assist in understanding the underlying model and is probably one of the most simple yet comprehensive tools which give you great insight into the underlying model.

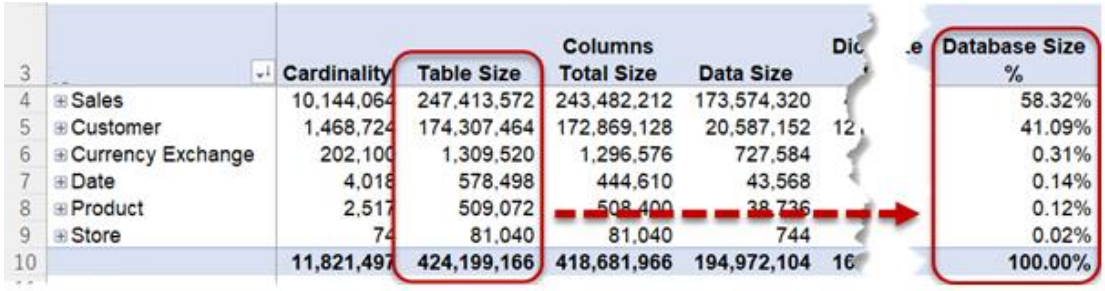
Here are some guidelines and tips on how to use this tool. Download Vertipaq Analyzer and follow the instructions to import the VPAX file.

Tables Page

The first page contain information about tables and columns and it contains all the relevant information in a single pivot table and provides a good overall summary of the model.

Cardinality represents the number of rows in the table, whereas table size is the size in bytes of the table. When optimizing a model, the larger tables will be where most of the effort will be and where there will be the most benefit by reducing the model size.

Database size percentage: This is the division of table size by the total table size. This tells you what percentage of total database size is being used by the table



	Cardinality	Table Size	Columns Total Size	Data Size	Dic	.e	Database Size %
Sales	10,144,064	247,413,572	243,482,212	173,574,320			58.32%
Customer	1,468,724	174,307,464	172,869,128	20,587,152	12		41.09%
Currency Exchange	202,100	1,309,520	1,296,576	727,584			0.31%
Date	4,018	578,498	444,610	43,568			0.14%
Product	2,517	509,072	508,400	38,736			0.12%
Store	74	81,040	81,040	744			0.02%
	11,821,497	424,199,166	418,681,966	194,972,104	16		100.00%

Drill down at the column level in order to find more information. Cardinality at the column level does not represent the number of rows. Instead, it is the cardinality of the column, the number of distinct values of the column. The larger the number of distinct values, the more problematic the column may be.

	Cardinality	Table Size	Columns Total Size	Data Size	Dictionary Size	Columns Hierarchies Size	Encoding_
Sales	**10,144,064**	**247,413,572**	**243,482,212**	**173,574,320**	**40,894,276**	**29,013,616**	**Many**
CustomerKey	1,468,724		78,230,368	27,050,888	39,429,640	11,749,840	HASH
Order Number	4,230,271		57,497,560	40,576,336	120	16,921,104	VALUE
Net Price	24,734		20,506,832	19,682,072	626,840	197,920	HASH
Delivery Date	3,360		16,420,200	16,216,008	177,264	26,928	HASH
Order Date	3,281		16,170,304	15,968,344	175,672	26,288	HASH
ProductKey	2,517		16,134,252	16,038,160	75,916	20,176	HASH
Unit Cost	1,956		14,459,352	14,402,024	41,632	15,696	HASH
Unit Price	1,761		14,399,068	14,344,584	40,356	14,128	HASH
Exchange Rate	5,068		9,522,440	9,177,528	304,320	40,592	HASH
StoreKey	74		109,280	105,984	2,656	640	HASH
Currency Code	5		17,848	648	17,120	80	HASH
Quantity	10		9,872	8,368	1,376	128	HASH
Line Number	7		4,836	3,376	1,364	96	HASH
Customer	**1,468,724**	**174,307,464**	**172,869,128**	**20,587,152**	**121,783,880**	**30,498,096**	**HASH (All)**
Currency Exchange	**202,100**	**1,309,520**	**1,296,576**	**727,584**	**424,616**	**144,376**	**Many**
Date	**4,018**	**578,498**	**444,610**	**43,568**	**351,554**	**49,488**	**Many**
Product	**2,517**	**509,072**	**508,400**	**38,736**	**402,912**	**66,752**	**Many**
Store	**74**	**81,040**	**81,040**	**744**	**77,616**	**2,680**	**Many**
	11,821,497	**424,199,166**	**418,681,966**	**194,972,104**	**163,934,854**	**59,775,008**	**Many**

For example, in the Sales table in the photo above, the column with the highest cardinality is Order Number with 4.23 million, whereas, the higher the number is, the larger the Columns Total Size and Data Size become, leading to a larger model size.

The total size is the sum of data size plus dictionary size plus columns hierarchies size. Depending on the compression that the engine decided to use column by column, you will have a larger size for the data size or for the dictionary size because the dictionary is useful for hash encoding, it contains the dictionary of the column, whereas the data size is the size of individual segments. They depend basically on the kind of compression and the number of rows which are stored in the model.

Inspect the size of each column, size of user hierarchies, the size of the relationships and the number of referential integrity violations, which is the number of times that there is a value that references a value in the target table that does not exist.

Tips

Review the first page of Vertipaq Analyzer

- Determine total size of the model
- Determine against total size of the model, which tables are the biggest in terms of total memory against the total model footprint. This will give you a good indication which of the tables are consuming the most amount of memory and start by analyzing these first. It is pointless analyzing smaller tables that represent a tiny fraction of the model

The column page contains the same information that is already present in the tables page. The only difference is in the way data is organized. All the columns are presented together in a single pivot table so there is no longer a division by table. If may be useful to sort the columns by size. This view is useful when focusing on problematic columns. Typically the first five to 10 columns, are responsible for the vast majority of the size of the database.

Tips

Review the number of columns in the table.

- Are there many unused columns that could be potentially removed
- Review the cardinality of the columns – this represents the distinct values of the column and the size. Review column size first as there is typically a relationship between cardinality and size. Typically look out for low cardinality but high memory, this could indicate a problem with partitioning or sorting or may require investigation into reducing the cardinality increase the duplicate
- Look out for calculated columns and determine if its better to do the calculated column at runtime using measures or move this calculation into the ELT layer Partition gives information about the number of partitions and number of segments for each table. For each table you have the size in number of rows, the number of partitions, the number of segments and the data size. The ratio between the number of rows and the number of partitions and segments gives a quick understanding of whether or not the model is over partitioned or under partitioned. Use this page to understand the quality of the partitions.

Tips

- Review the ratio between segments and partitions to determine if the model is over pr under partitioned
- Aim for approximately 10 segments per partition

User hierarchies

Shows all the user hierarchies which are defined in the model and shows the size of each hierarchy. The smaller the hierarchy the faster it will be to process and usage.

Tips

- Look out for many hierarchies created when Auto Date Time is enabled

Relationships

The relationship page contains all the relevant information about relationships and lists all the relationships with the from and the to table. The from and to column cardinality is the cardinality of

the column in the fact table (the from column) and the cardinality of the column in the target table (the to column)

4	Row Labels	1:M Ratio %	Missing Keys	Invalid Rows
5	⊞**Currency Exchange**	**1.99%**	**4,066**	
6	'Currency Exchange'[Date] ∞←1 'Date'[Date]	1.99%	4,066	101,650
7	⊞**Sales**	**14.48%**	**0**	
8	'Sales'[CustomerKey] ∞←1 'Customer'[CustomerKey]	14.48%	0	0
9	'Sales'[Delivery Date] ∞←1 'Date'[Date]	0.04%	0	0
10	'Sales'[Order Date] ∞←1 'Date'[Date]	0.04%	0	0
11	'Sales'[ProductKey] ∞←1 'Product'[ProductKey]	0.02%	0	0
12	'Sales'[StoreKey] ∞←1 'Store'[StoreKey]	0.00%	0	0
13	**Grand Total**	**14.48%**	**4,066**	

Missing keys indicates missing values in the fact table that point to values in the dimension that do not exist. Invalid rows indicates how many rows are affected by this.

- Review the difference between from and to and check that all values in the dimension are actually used in the fact table and there are no useless rows in the dimension
- Check the size of the relationship to make are no relationships which are too large and which may slow down the entire model
- Keep relationships small in order to have faster queries
- A relationship may grow because a column with a lot of distinct values is involved in a relationship
- A big difference between the To and From cardinalities is an indication that the dimension may contain useless data that can be removed
- Resolve missing keys and invalid rows – the impact is of missing or invalid rows is that DAX will generate a blank row in the dimensions which could cause problems in DAX calculations. Missing rows or invalid keys implies an issue with the ETL and should be rectified to make sure the rows are correctly referenced

Compression

Provides information on the kind of compression that has been used for different columns and the number of bits. The larger the number of distinct values, the more bits are needed and therefore the larger the column will be

Data Types

Tips

- Ensure that what is the engine is using is what you expect based on the source data. Check that the data type chosen for each column is correct and that data has been correctly imported into the model i.e. if a value is expected to be imported as an integer but has instead been imported as a floating point
- Check the compression type
- Strings will have a larger dictionary size Encoding

There are two types of encoding - Hash encoding or value encoding and these are important for performance.

Tips

- Typically columns that are used in a lot of calculations should be value encoded and columns that are used in relationships should be hash encoded
- Changing encoding would not typically impact the performance of smaller models
- It is not suggested to change encoding. Always test to determine if encoding changes yield performance gains
- Use hints in order to change encoding

Dax Expressions

Contains all the DAX expressions that are present in your model including measures, calculated columns, calculated tables and calculation groups (where applicable).

- Review quality of code written in DAX expressions
- Review overuse of calculated columns that consume memory
- Review hidden expressions