

Power BI refreshes reinvented

Streamlining Power BI Refreshes Advanced
Methods for Real-Time Refreshes

THANK YOU



Platinum



Gold



Lucient



Measure Killer



Silver



Bronze



After this session

Storage modes

Understand the different storage modes in Power BI and how this relates to refreshes.

Refresh operations

Understand which different options there are to refresh your Semantic Model manually as well as in an automated fashion.

Partitioning

Understand how different partitions in the data model are generated and can be triggered to refresh.

Orchestration

Understand how you can orchestrate your Power BI refresh and bind to other processes.

Marc Lelijveld

Technical Evangelist | Solution Architect
Macaw Netherlands



@MarcLelijveld



linkedin.com/in/MarcLelijveld



Data-Marc.com

FAVORITE STUFF:



What we cover today

- Refreshes?
- Incremental refresh
- High frequency refreshes & hybrid tables
- Refresh individual objects
- Orchestration
- Wrap-up



REFRESH



Refreshes

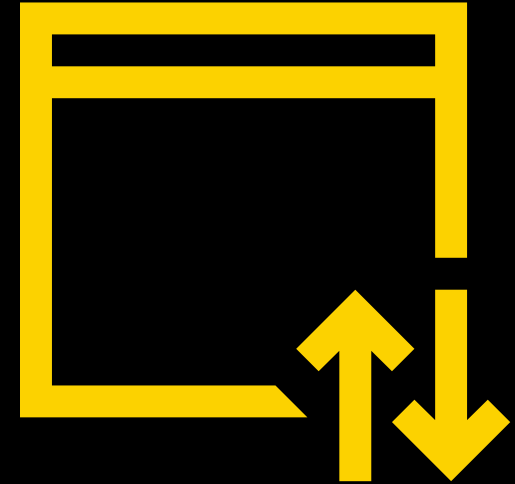
Understanding storage modes



Import



Dual



DirectQuery

Oh wait... we now also have DirectLake!

Storage Modes - Power BI Desktop

File Home Help External Tools

Clipboard Data Queries Relationships Security Q&A Language Linguistic schema Sensitivity Share

Product Subcategory

- English Product Subcategory N...
- French Product Subcategory Na...
- ProductCategoryKey
- ProductSubcategoryAlt...
- ProductSubcategoryKey
- Spanish Product Subcategory N...

Product

- Arabic Description
- Chinese Description
- Class
- Color
- Days To Manufacture
- Dealer Price

Internet Sales - Agg

- Count
- Order Calendar Year
- ProductSubcategory/Key
- Sales Amount

Internet Sales

- Carrier Tracking Number
- CurrencyKey
- Customer PO Number
- CustomerKey
- Discount Amount
- DueDateKey
- Extended Amount
- Freight Amount
- Order Calendar Year

Properties

General

Name: Internet Sales

Description: Enter a description

Synonyms: Enter a comma-separated list of synonyms for Q&A

Row label: Select a row label

Key column: Select a column with unique values

Is hidden: No

Is featured table: No

Advanced

Storage mode: DirectQuery

Understanding Storage Modes

Three storage modes

- Import – data cached in the model
- DirectQuery – queries are submitted to the back-end data source
- Dual – can act in both above storage modes, depending on query context

Configuring storage modes

- Storage modes are set on table level
- Setting storage mode to Import is an irreversible operation
- Data in DirectQuery mode cannot be displayed in the data tab

Caches and DirectQuery



Risks of mixing storage modes

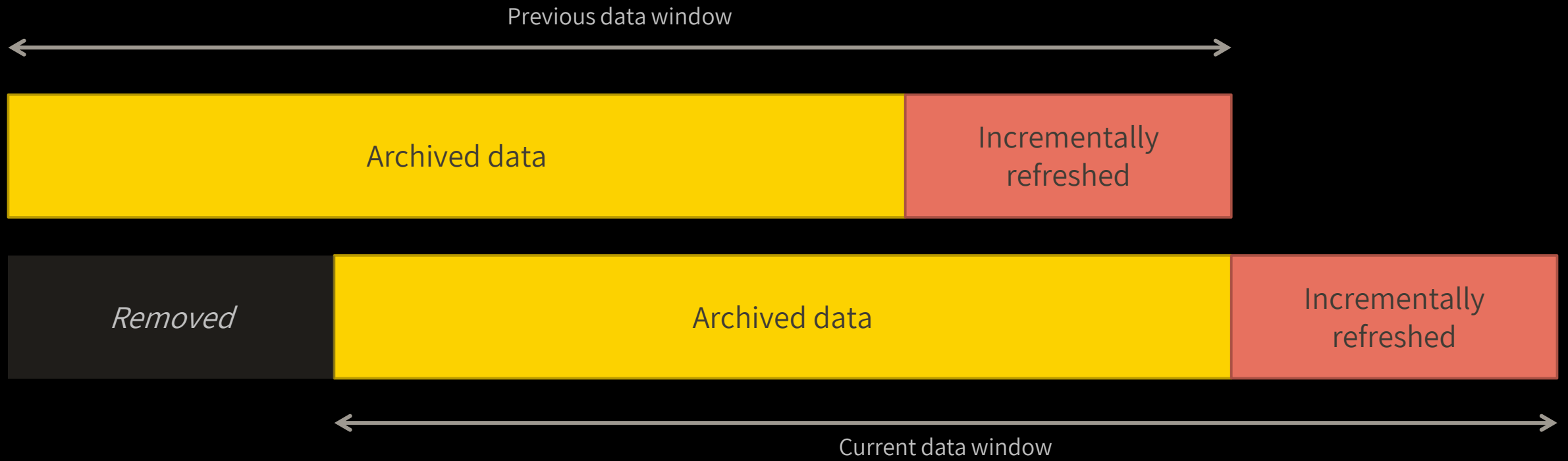
- Avoid mismatch in results when query bits DirectQuery compared to Import
- Data cached (import) could be behind compared to DirectQuery data
- Make sure cached data is kept in sync – regularly refresh!

Benefits to choose one or the other

- Improve query performance – cache data for faster end-user performance
- Data refresh optimization – no need to refresh for non-cached data
- Near-real time requirements – reduce query latency when in DirectQuery mode
- Large Semantic Models – choose to not import certain data by leveraging aggregations for example

Different refresh operation types

Type	What	Power BI Desktop 	Power BI Service (Interface) 
Manual	Individual Table	✓	✗
	Full model	✓	✓
Scheduled	Individual Table	✗	✗
	Full model	✗	✓

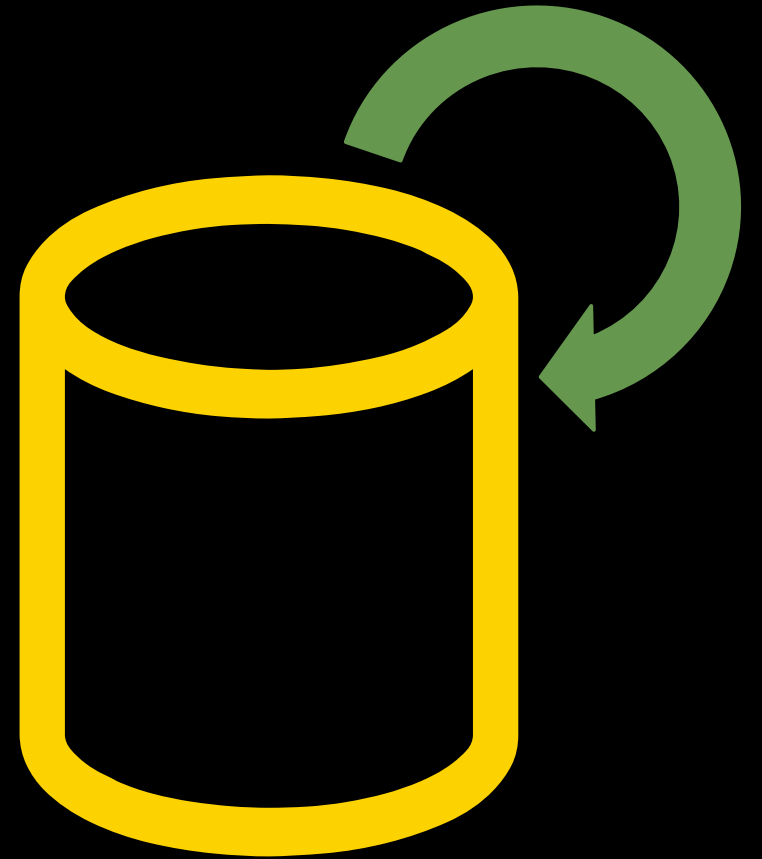


Incremental refresh

Considering incremental refresh

Why should you use it?

- Large data volumes
- Faster refreshes
- Resource consumption reduced
(Both source & Power BI side)



Incremental refresh

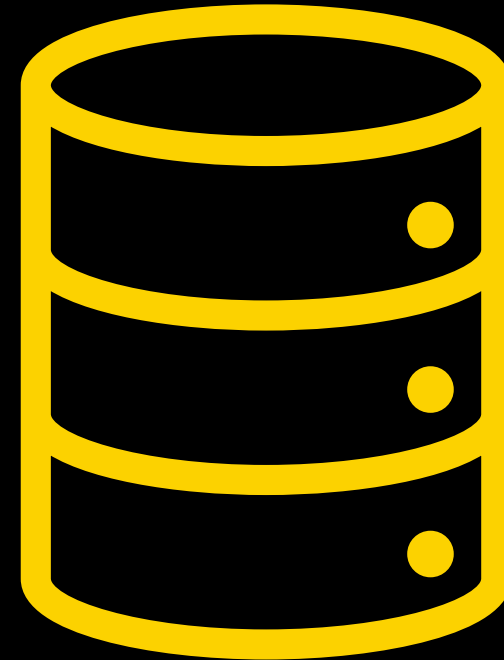
Incremental refresh is supported for Power BI Premium, Premium per user, **Power BI Pro**, and Power BI Embedded Semantic Models.

Getting the latest data in **real time** with DirectQuery is **only supported for Power BI Premium**, Premium per user, and Power BI Embedded Semantic Models.

Supported data sources

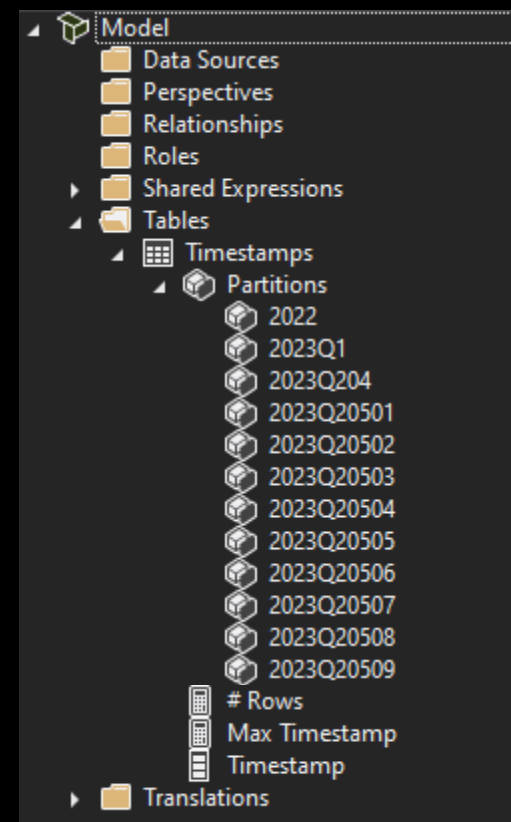
Data sources must be

- Structured
- Relational
- Allow query folding
- Typically – SQL like sources



Breaking up your data in partitions

- Each partition represents a time frame
- New partitions get added automatically
- Old partitions (in archive) get merged



Configure Incremental Refresh

Manage Parameters

RangeStart

RangeEnd

Name

RangeStart

Description

☒ Required

Type

Date/Time

Suggested Values

Any value

Current Value

1/1/2022 12:00:00 AM

OK

Cancel

Incremental refresh and real-time data

Refresh large tables faster with incremental refresh. Plus, get the latest data in real time with DirectQuery (Premium only). [Learn more](#)

① These settings will apply when you publish the dataset to the Power BI service. Once you do that, you won't be able to download it back to Power BI Desktop. [Learn more](#)

- Select table**

Orders
- Set import and refresh ranges**

☒ Incrementally refresh this table

Archive data starting 5 Years before refresh date

Data imported from 1/1/2017 to 11/17/2022 (inclusive)

Incrementally refresh data starting 3 Days before refresh date

Data will be incrementally refreshed from 11/18/2022 to 11/20/2022 (inclusive)
- Choose optional settings**

☐ Get the latest data in real time with DirectQuery (Premium only) [Learn more](#)

Selected table cannot be folded for DirectQuery.

☐ Only refresh complete days [Learn more](#)

☐ Detect data changes [Learn more](#)
- Review and apply**

Archived

Incremental Refresh

5 years before 3 days before Refresh date

Moving time frame

Previous data window



Archived data

Incrementally
refreshed

Removed

Archived data

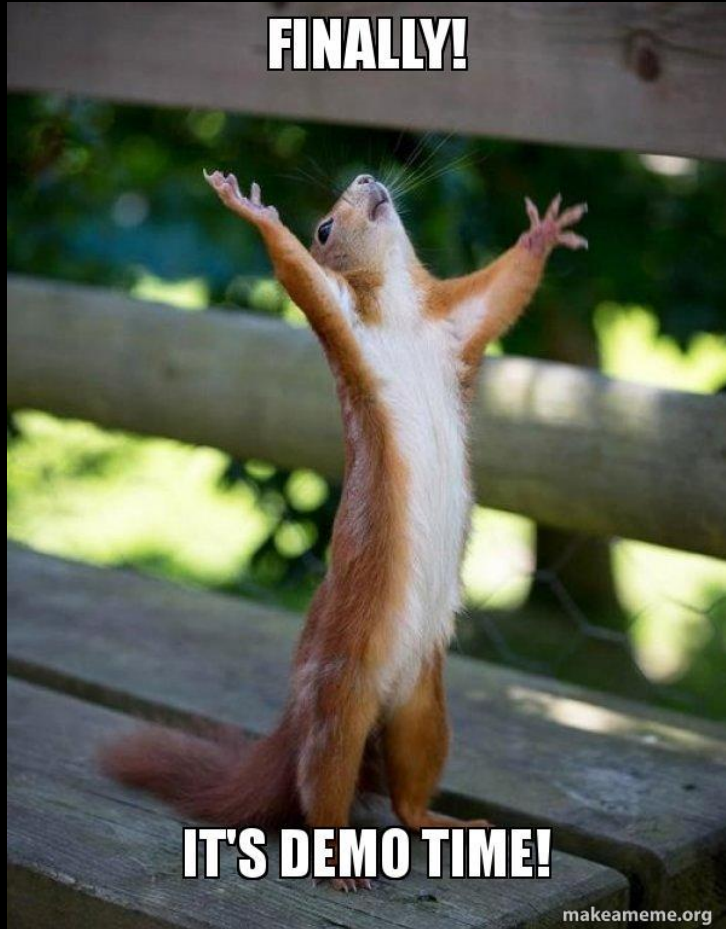
Incrementally
refreshed

Current data window

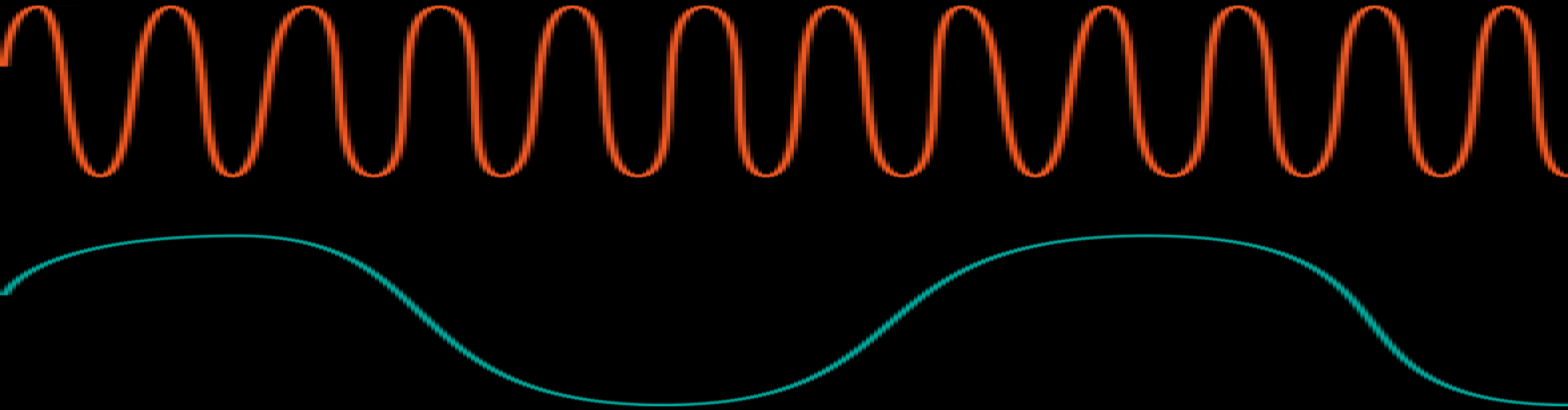


Detect data changes

- Even more selective refresh – only that what has been changed
- Based date/time column used to identify and refresh only those days where the data has changed
- Typically, system generated dates like Modified Date Time
- Should never be the same column as your incremental periods
- The maximum value is evaluated for each partition **in the incremental range** to detect whether the partition should be refreshed or not



Demo - configure
incremental refresh



High frequency refreshes & hybrid tables

What is high frequency?

Let's say real time?

- Hourly?
- Minutes?
- Seconds?

Or...

- Just daily batches



Refresh limitations

Frequency	Manual	Scheduled (via the service)
Free	Up to 8 times per day	
Pro	Up to 8 times per day	
Premium per User	Unlimited	48 times per day
Premium Capacity	Unlimited	48 times per day

Hybrid tables

- Live / Realtime data in Power BI
- Combines different storage modes on partition level in a single table
- Goes hand-in-hand with Incremental Refresh

Granularity	Name	Row Count	
Year	2011	295,489,717	Archived: Import
Year	2012	297,678,498	
Year	2013	295,575,442	
Year	2014	292,477,875	
Year	2015	297,780,469	
Year	2016	294,060,081	
Year	2017	300,419,682	
Year	2018	296,541,108	
Year	2019	292,787,420	
Year	2020	299,273,979	
Quarter	2021Q1	74,135,277	Incremental refresh: Import
Month	2021Q104	24,939,498	
Day	2021Q10501	820,805	
Day	2021Q10502	826,885	Real time: DirectQuery
Day	2021Q10503	821,043	
Day-DirectQuery	2021Q10504-DQ	271,110	
Total		3,063,898,887	

Hybrid tables

- Implementation with Incremental Refresh
- Customizable via 3rd party tooling like Tabular Editor (*unsupported?*)

Limitation:

Only 1 DQ partition per table allowed.

Incremental refresh and real-time data

ⓘ These settings will apply when you publish the dataset to the Power BI service. Once you do that, you won't be able to download it back to Power BI Desktop. [Learn more](#)

- Select table**

Sales Agg
- Set import and refresh ranges**

☒ Incrementally refresh this table

Archive data starting

10

Years

 before refresh date

Data imported from 3/2/2012 to 2/20/2022.

Incrementally refresh data starting

10

Days

 before refresh date

Data will be incrementally refreshed from 2/20/2022 to 3/2/2022.
- Choose optional settings**

☒ Get the latest data in real time with DirectQuery (Premium only) [Learn more](#)

☒ Only refresh complete days [Learn more](#)

☐ Detect data changes [Learn more](#)
- Review and apply**

10 years before refresh date

Archived

10 days before refresh date

Incremental Refresh

Real time

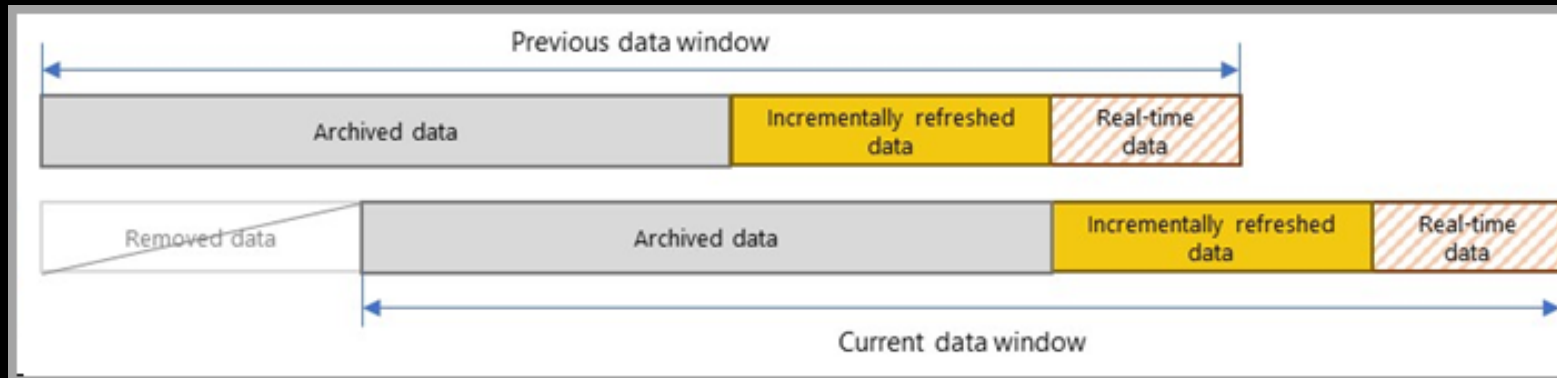
Refresh date

Apply

Cancel

Hybrid tables – what challenge does it solve?

- Realtime scenarios without full tables on DQ mode
- No complex refresh mechanisms needed with partition refresh and queries over XMLA
- No more multiple tables and complex DAX to combine to achieve the same goal



Hybrid tables – Keep in mind that...

- Premium feature
- DAX restrictions for DirectQuery apply
- Limited Power Query capabilities (due to DQ)
- Requires Large Semantic Model Format (storage) in workspace
- Performance hit on upstream data sources



Demo - configure
hybrid tables and
streaming data

Hybrid Table
...
Feb 2022
March 2022
...
...
...



Import



DirectQuery

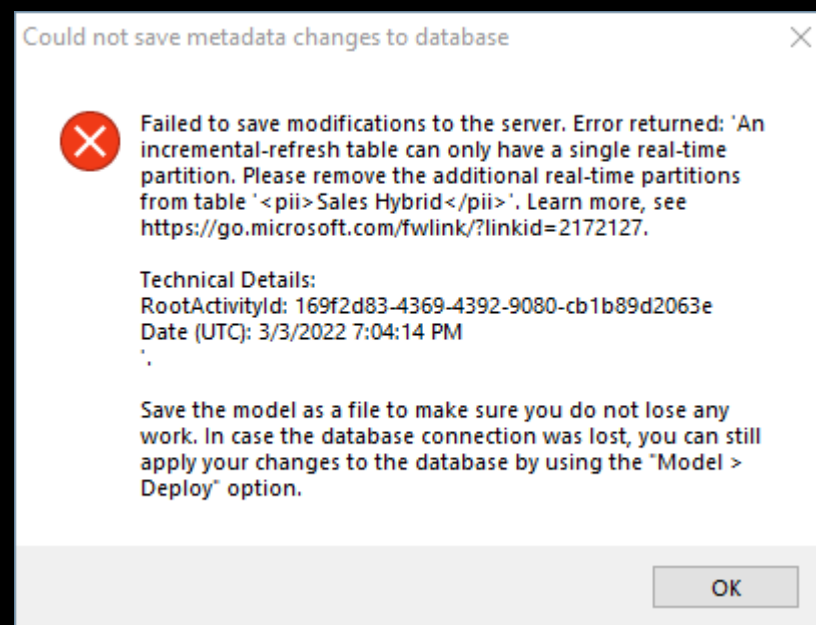


▼	📊	Sales Hybrid
▼	📦	Partitions
	📦	2021Q309
	📦	2021Q410
	📦	2021Q411
	📦	2021Q412
	📦	2022Q101
	📦	2022Q102
	📦	2022Q10301
	📦	2022Q10302
	📦	2022Q10303-onward

Can I change partition storage modes?

You cannot adjust tables with incremental refresh policies applied. However, there are other options to consider.

▼ Options	
Data View	Default
Mode	DirectQuery
Source Type	PolicyRange



Manually setup partitioning

Create your own hybrid table setup, using 3rd party tooling like Tabular Editor

The screenshot displays the Tabular Editor 2.16.5 interface. On the left, the 'Model' tree shows a hierarchy of data sources, with 'Sales Hybrid-DQ' selected under 'Partitions'. The main area shows the 'Expression Editor' with a DAX query. The query is as follows:

```
1 let
2   Source = Sql.Databases((SynapseEndpoint)),
3   #"WideWorldImportersDW-Standard" = Source[[Name=(DatabaseName)]]{[Data],
4   Fact_Sale = #"WideWorldImportersDW-Standard">{[Schema=(SynapseSchema), Item="Sales"]}[Data],
5   #"Filtered Rows" = Table.SelectRows(#"Fact_Sale", each [InvoiceDateKey] < #date(2022, 01, 01) ),
6   SplitByCharacter = Table.TransformColumnNames(#"Filtered Rows", each
7     Text.Combine(
8       Splitter.SplitTextByCharacterTransition({"a".."z"}, {"A".."Z"})(_)
9     , " ")
10 )
11 in
12   SplitByCharacter
```

Below the query editor, the 'Basic' metadata tab is active, showing details for 'Sales Hybrid-DQ'. The 'Mode' is set to 'DirectQuery', which is highlighted with a red box. The 'Source Type' is 'M'. The status bar at the bottom indicates '1 partition (m - directquery) selected.' and '0 BP issues'.

Property	Value
Description	
M Expression	let Source = Sql.Databases((SynapseEndpoint)). #"WideWorldImportersDW-Standard">{[Schema=(SynapseSchema), Item="Sales"]}[Data], Fact_Sale = Table.SelectRows(#"Fact_Sale", each [InvoiceDateKey] < #date(2022, 01, 01)), SplitByCharacter = Table.TransformColumnNames(#"Filtered Rows", each Text.Combine(Splitter.SplitTextByCharacterTransition({"a".."z"}, {"A".."Z"})(_), " ")) in SplitByCharacter
Name	Sales Hybrid-DQ
Annotations	0 annotations
Last Processed	31-12-1699
Object Type	Partition (M - DirectQuery)
State	Ready
Options	
Attributes	
Data View	Default
Mode	DirectQuery
Source Type	M

Hybrid Table
...
< 2022
>= 2022
...
...
...

} DirectQuery

} Import

DirectQuery Partition definition

```
let
    Source = Sql.Databases((SynapseEndpoint)),
    #"WideWorldImportersDW-Standard" = Source{[Name=(DatabaseName)]}[Data],
    Fact_Sale = #"WideWorldImportersDW-Standard"[[Schema=(SynapseSchema),Item="Sales"]][Data],
    #"Filtered Rows" = Table.SelectRows(#"Fact_Sale", each
        [InvoiceDateKey] < #date(2022, 01, 01) and
        [InvoiceDateKey] >= #date(2013, 01, 01)),
        SplitByCharacter = Table.TransformColumnNames(#"Filtered Rows", each
            Text.Combine(
                Splitter.SplitTextByCharacterTransition({"a".."z"}, {"A".."Z"})(_)
                , " ")
        )
in
    SplitByCharacter
```

Import Partition definition

```
let
    Source = Sql.Databases((SynapseEndpoint)),
    #"WideWorldImportersDW-Standard" = Source{[Name=(DatabaseName)]}[Data],
    Fact_Sale = #"WideWorldImportersDW-Standard"[[Schema=(SynapseSchema),Item="Sales"]][Data],
    #"Filtered Rows" = Table.SelectRows(#"Fact_Sale", each
        [InvoiceDateKey] >= #date(2022, 01, 01) and
        [InvoiceDateKey] < #date(2022, 03, 31)),
        SplitByCharacter = Table.TransformColumnNames(#"Filtered Rows", each
            Text.Combine(
                Splitter.SplitTextByCharacterTransition({"a".."z"}, {"A".."Z"})(_)
                , " ")
        )
in
    SplitByCharacter
```

Downsides

Potential performance impact

- Policy definitions and partitions defined by Power BI will be scanned more optimally when a DAX query is executed
- With manual setup partitions, your DAX expression might be broken up in multiple sub-queries which hit all partitions

⊗ Caution

At this time, a write operation on a dataset authored in Power BI Desktop will prevent it from being downloaded back as a PBIX file. Be sure to retain your original PBIX file.



Refresh individual objects

Effective refreshing

Considerations

- Refreshing the entire model takes too long with high load on sources
- Can we only refresh certain tables?
- Can we only refresh certain partitions?
- Can we use DQ tables/partitions (Hybrid Tables)

What do we need?

- Enhanced refresh API
- XMLA Endpoints

Smaller increments



Bind your refresh operations in Power BI to refresh operations in for example your data platform



Run smaller increments to speed up the end-to-end process



Trigger one (or multiple) selective table(s) or partition(s) at a time.

Smaller increments

Smaller increments can only be triggered to refresh via:

- Manual processing via Management Studio (or other tools)
- Power BI Enhanced Refresh API - Previously known as Async refresh API
- XMLA endpoints

Enhanced refresh API

More granular controls

- type - full / clearValues / calculate / dataonly / automatic / defragment
- Commitmode – default is transactional
- maxParallism – 10 max
- retryCount – number of retries before failing
- objects – which objects to refresh
- applyRefreshPolicy – true/false whether you want to apply potential incremental refresh if configured
- effectiveDate – in case of incremental refresh, this parameter overwrites the current date

Enhanced refresh API

Specify the objects to refresh

POST

<https://api.powerbi.com/v1.0/myorg/groups/f089354e-8366-4e18-aea3-4cb4a3a50b48/datasets/cfafbeb1-8037-4d0c-896e-a46fb27ff229/refreshes>

```
{
  "type": "Full",
  "commitMode": "transactional",
  "maxParallelism": 2,
  "retryCount": 2,
  "objects": [
    {
      "table": "DimCustomer",
      "partition": "DimCustomer"
    },
    {
      "table": "DimDate"
    }
  ]
}
```



Demo – Refresh one table or partition

More than refreshes

Enhanced refresh API also allows you to

- **GET** refresh history
- **GET** individual refresh details like progress or failures
- **DELETE** a running refresh

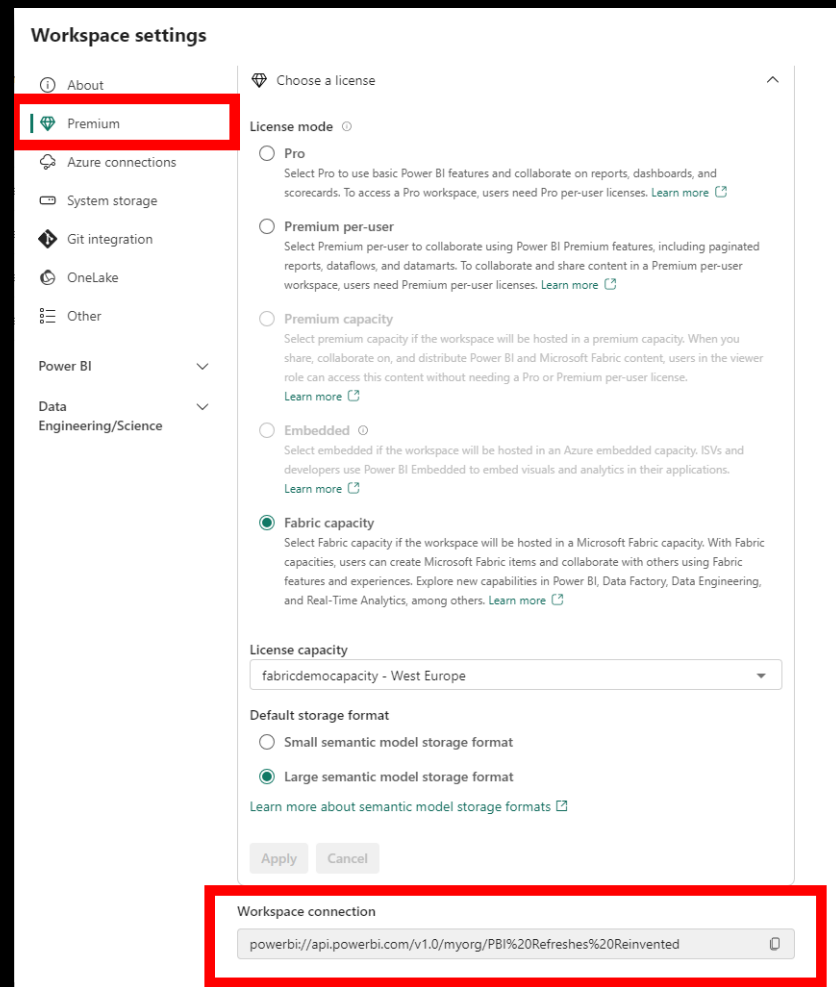
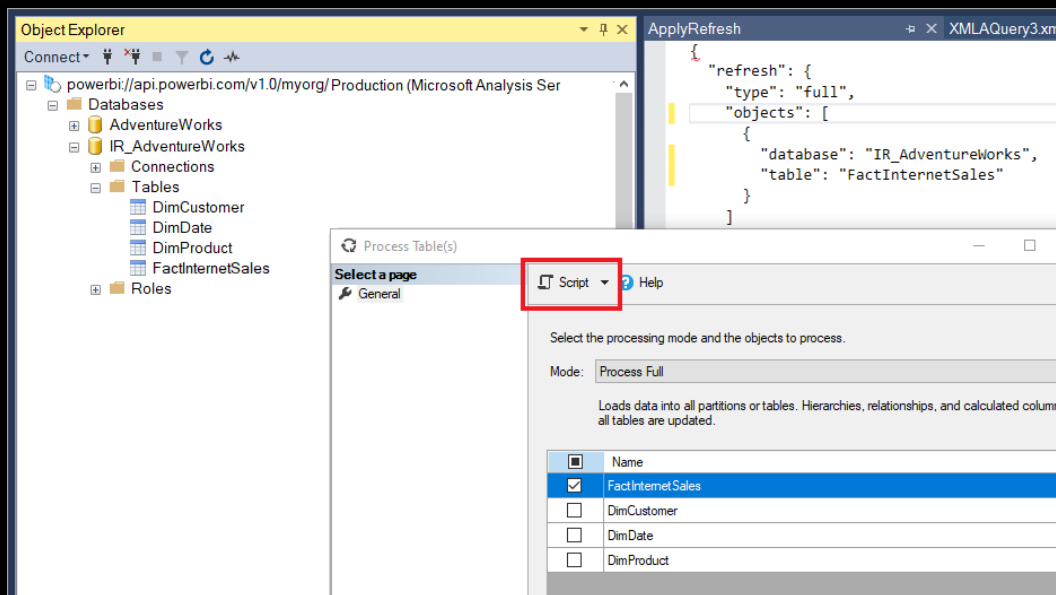
JSON

Copy

```
{
  "startTime": "2020-12-07T02:06:57.1838734Z",
  "endTime": "2020-12-07T02:07:00.4929675Z",
  "type": "Full",
  "status": "InProgress",
  "currentRefreshType": "Full",
  "objects": [
    {
      "table": "DimCustomer",
      "partition": "DimCustomer",
      "status": "InProgress"
    },
    {
      "table": "DimDate",
      "partition": "DimDate",
      "status": "InProgress"
    }
  ]
}
```


Is the API the only option?

- Execute operations over XMLA endpoints
- Using Tabular Model Scripting Language
- More granular control using SQL Service Management Studio



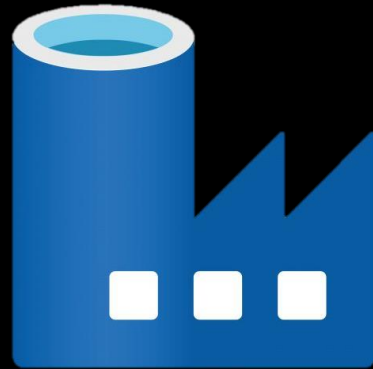


Binding refreshes to processes

Central E2E orchestration

- Combine pipelines from Data Platform with Power BI
- Lowest latency between source and report
- Incremental loading where possible
- Consider including backup operations for Power BI

What to use to trigger and bind together?





Demo –Refresh API
from a Pipeline

Wrap up

LET'S
RECAP...

Refreshes can be manual or **automated**

Incremental refresh makes your refresh **more efficient**

With **detect data changes**, further optimization can be done

Partitioning in incremental refresh should be **based on functional dates**

Lower latency by linking Power BI refreshes to other processes

Use the enhanced refresh API to **trigger individual objects**

Use the toolkit of your preference for **end-to-end orchestration**

Resources

Setting up scheduled refreshes

<https://learn.microsoft.com/en-us/power-bi/connect-data/refresh-scheduled-refresh>

Configure incremental refresh

<https://learn.microsoft.com/en-us/power-bi/connect-data/incremental-refresh-configure>

Enhanced refresh API

<https://learn.microsoft.com/en-us/power-bi/connect-data/asynchronous-refresh>

Semantic Model connectivity with XMLA endpoints

<https://learn.microsoft.com/en-us/power-bi/enterprise/service-premium-connect-tools>

Tabular Model Scripting Language (TMSL)

<https://learn.microsoft.com/en-us/analysis-services/tmsl/tabular-model-scripting-language-tmsl-reference?view=asallproducts-allversions>

These slides

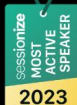
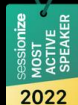
<https://github.com/marclelijveld/Slide-decks>

Thanks for attending!



Marc Lelijveld

Technical Evangelist | Architect
Macaw Netherlands



@MarcLelijveld



linkedin.com/in/MarcLelijveld



Data-Marc.com



https://evals.datagrillen.com/evals_vienna.aspx