Tabular Editor

Choosing the Right Storage Mode: Import, Direct Lake, DirectQuery & Hybrid



Data Saturday Holland 2026

Thank you sponsors!



























Meet the speakers



Greg Baldini
Core Developer

Builds high-performance data solutions at Tabular Editor. Loves tackling big, hairy data problems – especially optimizing models. Talk to him about anything.

linkedin.com/in/gregbaldini/



Just Blindbæk
Field CTO

20 years in semantic modeling. Customer- and partner-facing at Tabular Editor; previously a consultant and trainer. SSAS Maestro and Microsoft MVP, Data Platform.

linkedin.com/in/blindbaek/

Tabular Editor

Agenda

1. **Foundations:** The landscape of storage modes
2. **Deep dives:** Capabilities, limits and anti-patterns
3. **Mixing modes:** Composite, Hybrid & Proxy models
4. **Memory, refresh and cost implications**
5. **Decision framework**
6. **Wrap-up & Q&A**

Foundations

Foundations

or

VertiPaq internals: a full day session in 20 minutes

Tabular Editor

Considerations

Modes	Data freshness	Memory footprint	Refresh cost	Query performance	DAX feature support	License requirement
Import						
DirectQuery						
Direct Lake						
Mixing						

Tabular Editor

Everything is partitions

- Tables hold one or more partitions
- Storage mode is set on the partition
- Partitions: unit of management for data
    - PQ/M definition or legacy provider
    - Tables: aliases for a collection of same-schema partitions

Import

Modes	Data freshness	Memory footprint	Refresh cost	Query performance	DAX feature support	License requirement
Import	Refresh schedule	Model data + user queries + refresh	Power Query / M DAX calculated objects	++++	Everything	Free Desktop PBI Pro PBI PPU Fabric capacity Azure A SKU SSAS / AAS

- The default
- Full copy of data in VertiPaq: all data and relationship indices
- Orchestrate refreshes: scheduled refresh or API
- Flexible import logic: Power Query / M, all DAX calculated objects fully supported

Best when

- Stable historical data
- Need full DAX support
- You want optimization levers for size and query

Tabular Editor

DirectQuery

Modes	Data freshness	Memory footprint	Refresh cost	Query performance	DAX feature support	License requirement
DirectQuery	Live data at query time, per visual	User queries	n/a (all upstream)	Highly variable	Missing some DAX functions Calculated columns must be row-level	Depends on DQ source

- Usually* talking about DirectQuery to relational source: DAX->SQL translation
- Performance depends on optimization in source system
- Orchestrate data processing upstream

Best when

- Data latency matters most
- Complex security exists in source
- Overly strict data residency requirements

* Microsoft lumps many technologies under "DirectQuery"

Direct Lake

Modes	Data freshness	Memory footprint	Refresh cost	Query performance	DAX feature support	License requirement
Direct Lake	Reframing schedule	Model data + user queries	Minimal	Import equivalent at best	DAX calculated objects in preview	Fabric capacity

- Orchestrate data processing upstream
- Framing: point semantic model at an "as-of" transaction time in delta log
- Data served from VertiPaq
  - Transcode: read parquet from OneLake into VertiPaq in RAM
  - Inherits parquet row groups as segments
  - Relationship indices not cached

Best when

- Really clean data
- Amortize refresh cost across data access
- Robust upstream orchestration
- Acceptable to have slow time-to-first-byte

Considerations recap

Modes	Data freshness	Memory footprint	Refresh cost	Query performance	DAX feature support	License requirement
Import	Refresh schedule	Model data + user queries + refresh	Power Query / M DAX calculated objects	++++	Everything	Free Desktop PBI Pro PBI PPU Fabric capacity Azure A SKU SSAS / AAS
DirectQuery	Live data at query time, per visual	User queries	n/a (all upstream)	Highly variable	Missing some DAX functions Calculated columns must be row-level	Depends on DQ source
Direct Lake	Reframing schedule	Model data + user queries	Minimal	Import equivalent at best	DAX calculated objects in preview	Fabric capacity

Deep dives

VertiPaq

- Serves user queries for Import and Direct Lake
- Highly parallel, low-latency, high-concurrency, compressed, in-memory, column-store engine
- The storage engine (SE) portion in query plans
- Segments: unit of compression, unit of parallelism
- Compression
    - Value
    - Dictionary / hash
    - Run length encoding (RLE)

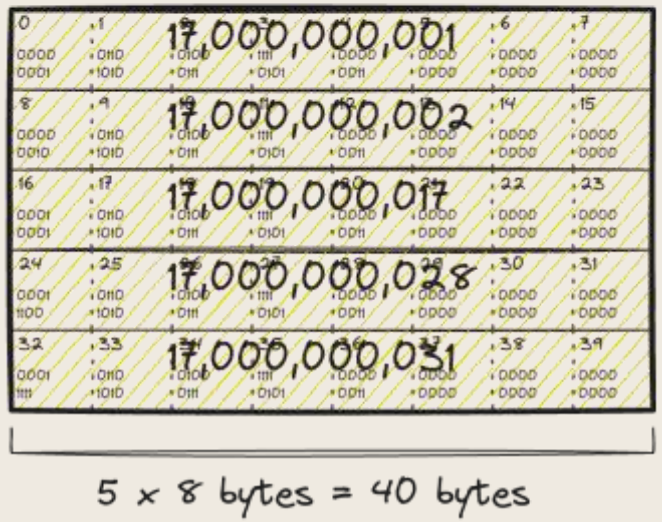
VertiPaq types

VertiPaq type	DAX / Tabular type
64-bit integer	- Whole number - Fixed decimal (currency)
64-bit floating point	- Decimal - Datetime: integer part as days since 1899-12-30; decimal part as fraction of day since 00:00:00 - Date: integer part as days since 1899-12-30 - Time: [0, 1) as fraction of day since 00:00:00
Bit	Logical / boolean
String	Text

Value encoding

raw

Number
17,000,000,001
17,000,000,002
17,000,000,017
17,000,000,028
17,000,000,031



Some questions:

What is the same?
What is different?

How big is the common part?
How big is the varying part?

Tabular Editor

Value encoding

raw

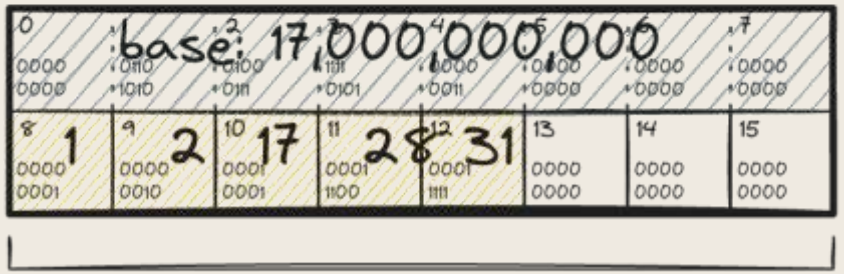
Number
17,000,000,001
17,000,000,002
17,000,000,017
17,000,000,028
17,000,000,031



5 x 8 bytes = 40 bytes

value encoded

Offset
1
2
17
28
31



8 + 5 x 1 = 13 bytes

Tabular Editor

Dictionary encoding

raw

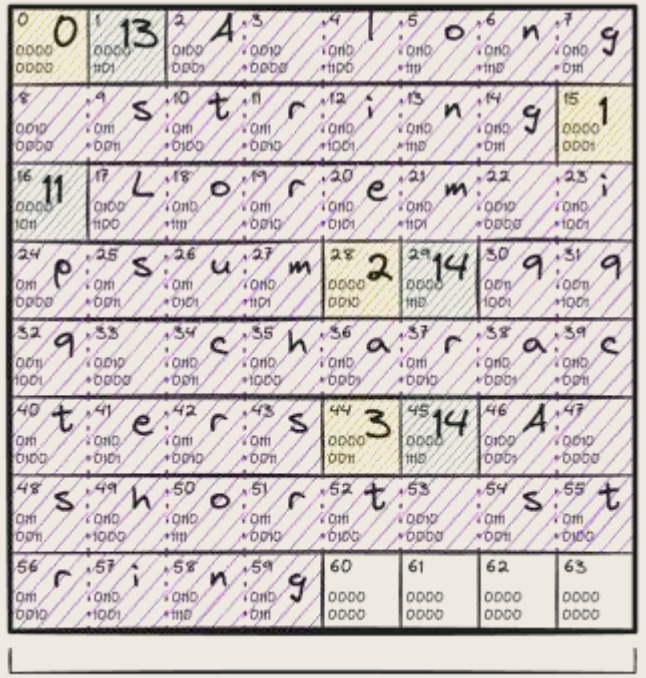
String
A long string
Lorem ipsum
Lorem ipsum
999 characters
A short string
999 characters



83 bytes

dictionary

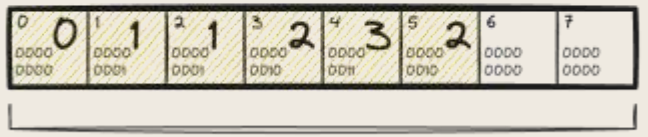
DictKey	String
0	A long string
1	Lorem ipsum
2	999 characters
3	A short string



60 bytes

encoded column

DictKey
0
1
1
2
3
2



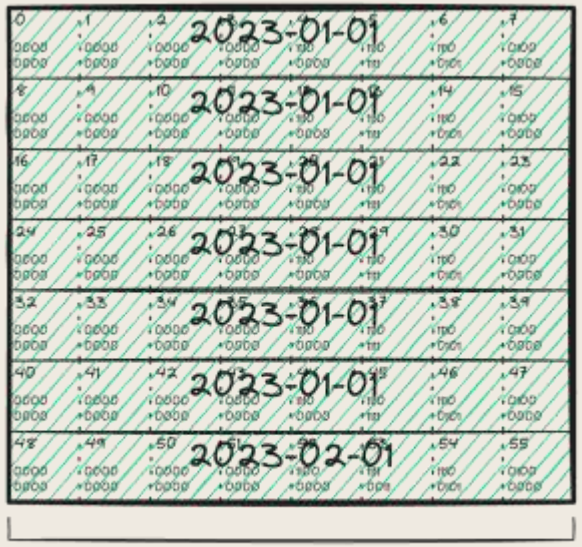
6 bytes

Tabular Editor

Run-length encoding

raw

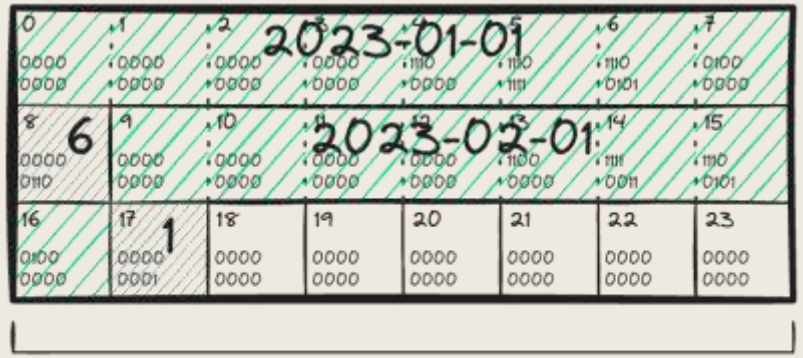
Date
2023-01-01
2023-01-01
2023-01-01
2023-01-01
2023-01-01
2023-01-01
2023-02-01



7 x 8 bytes = 56 bytes

run-length encoded

Date	Count
2023-01-01	6
2023-02-01	1



2 x (8 + 1) = 18 bytes

Tabular Editor

Direct Lake

- Loads data into VertiPaq engine in-memory

Direct Lake

– Loads data into VertiPaq engine in-memory

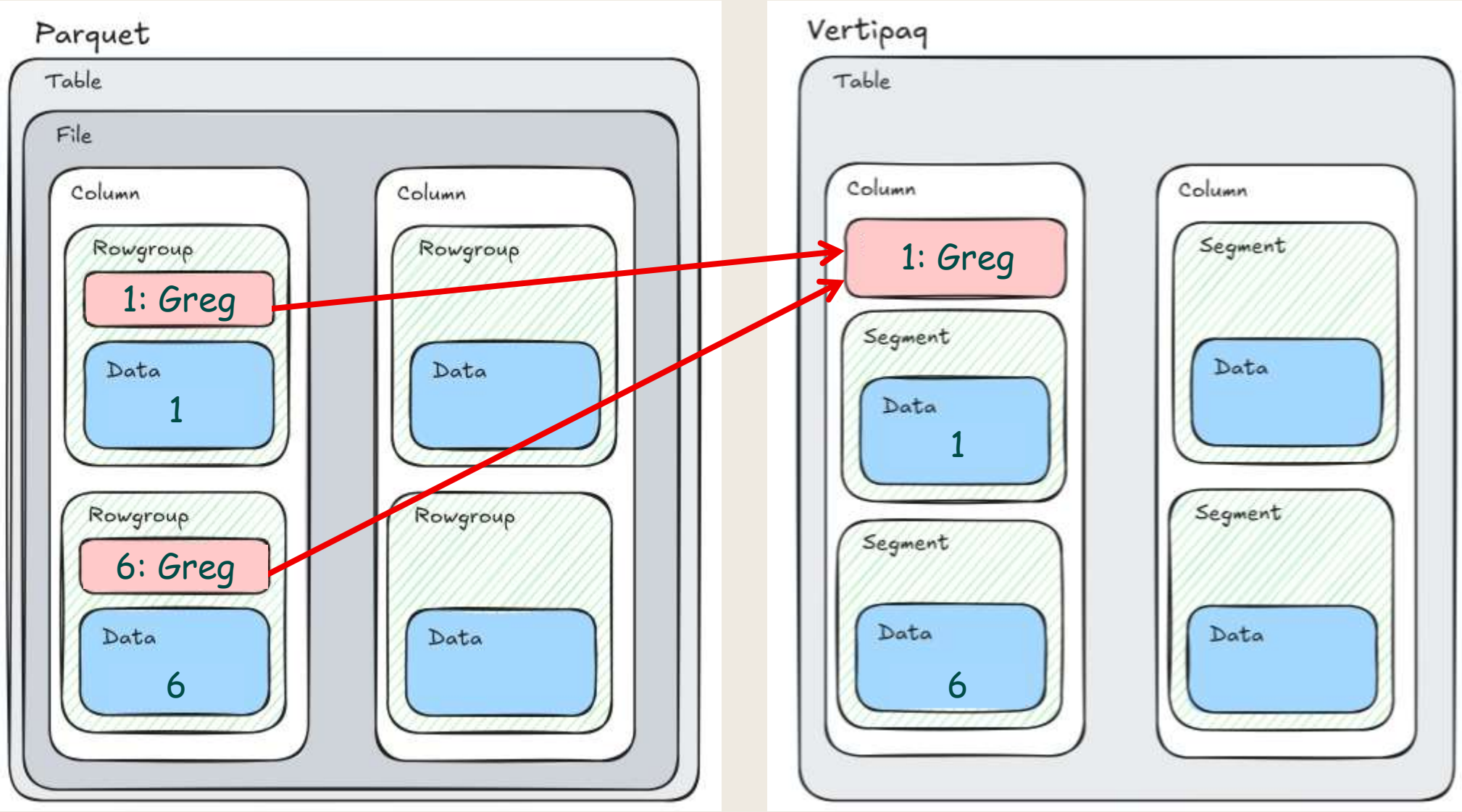
Direct Lake

- Loads data into VertiPaq engine in-memory

Direct Lake

–Loads data into VertiPaq engine in-memory

Direct Lake - Transcoding



Tabular Editor

Direct Lake

- Framing vs. transcoding
  - framing updates the Delta log reference (fast, metadata-only)
  - transcoding loads column segments into VertiPaq on first query for that column
- Over OneLake vs. over SQL Endpoint
  - SQL Endpoint can fall back to DirectQuery (unsupported functions, row limits, relationship types)
  - OneLake variant does not fall back: hard limits instead
- Appropriate for: large volumes in Fabric, many infrequently-accessed columns, enterprise distribution.
- Inappropriate for: non-technical teams, early-stage implementations, scenarios where calculated columns are needed without an upstream Delta table change.

Mixing modes

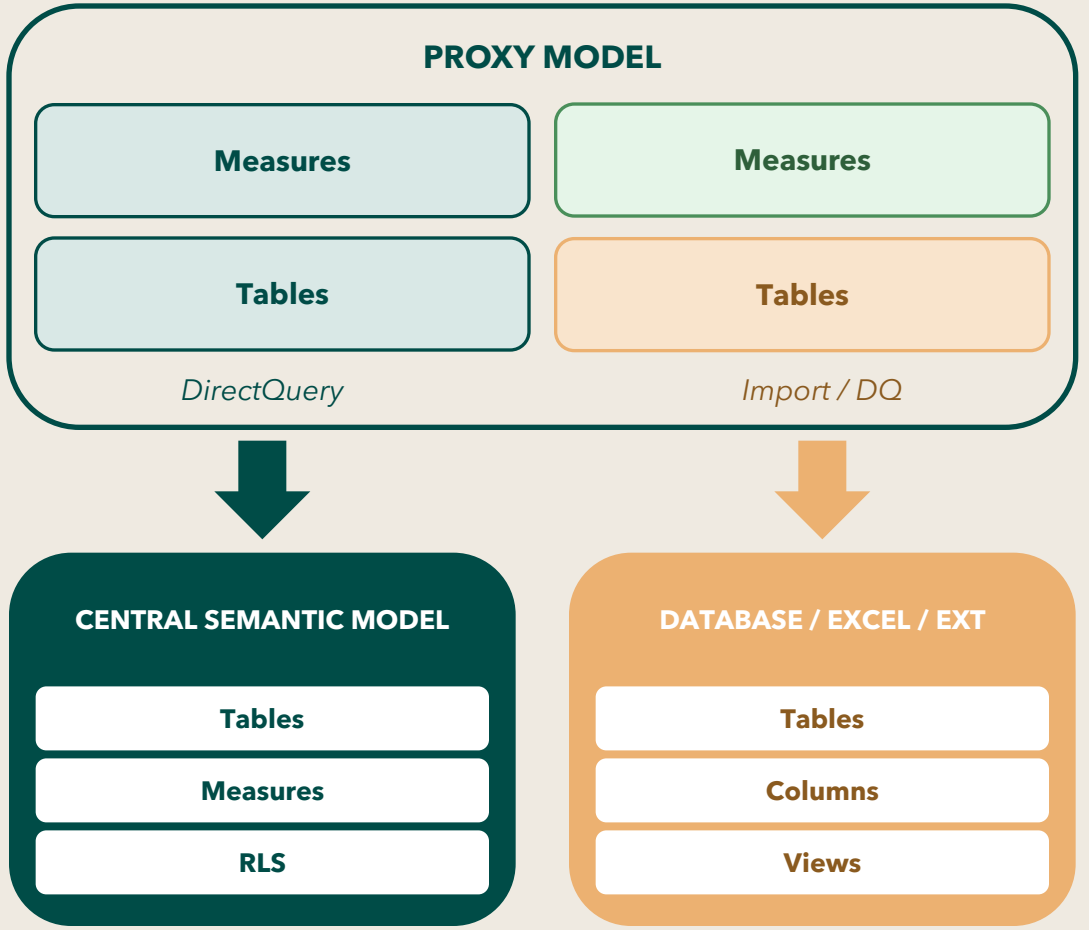
Composite / mixed-mode / dual-mode models



- Any combination of storage modes in one semantic model.
- Important clarification: composite ≠ only Import + DQ.
- It includes Direct Lake + Import (now in public preview – regular relationships, no limited relationship penalty), DQ + Import, DQ + Direct Lake, and more.
- **Import + DQ – user-defined aggregations:** keep a small pre-aggregated Import table for fast queries and push high-cardinality detail (transaction IDs, row-level attributes) to a hidden DQ table. Dimensions in Dual mode serve both sides without limited relationships.
- **Direct Lake + Import – dimensions in Import:** unlocks Calculated Columns and Hierarchies on dimensions, which Direct Lake does not yet support – while fact tables stay in Direct Lake.

Proxy model pattern

- A DirectQuery connection to another published semantic model – extends a central model without duplicating data or logic. Inherits upstream RLS; performance is bounded by upstream.
- **Sweet spot:** central model covers 80–95%, extended by max ~20% (e.g. product names → families).
- **Watch out:** using it to stitch many models together – high-cardinality relationships will hurt performance.



Hybrid tables



- Partitioned tables spanning multiple storage modes.
- **Classic pattern:** Import for history + DirectQuery for the current hot partition (near-real-time tail without querying the whole table).
- **Reverse pattern:** Import for today (if freshness of the current day is paramount and data is small) + DirectQuery for deep history (rarely queried, no memory cost).
- **Setup:** Power BI Desktop's incremental refresh UI builds the classic pattern automatically. Custom partitions (like the reverse pattern) require Tabular Editor via the XMLA endpoint.
- **Licensing:** incremental refresh alone runs on Pro. Adding a DirectQuery partition (i.e. making it hybrid) needs PPU / Premium / Fabric – that's where the read/write XMLA endpoint lives.

Composite model anti-patterns

⚠ Composite as primary architecture

Mixing modes as a default instead of a targeted fix.

INSTEAD

Start single-mode. Reach for composite only when a specific problem demands it.

⚠ DQ to avoid duplicating dimensions

Composite-model complexity to save a 50 MB dim table.

INSTEAD

Copy the dim into Import. Duplication is cheap; complexity isn't.

⚠ Proxy chain abuse

Model A → Model B → Model C. Every hop adds latency and a failure point.

INSTEAD

Keep the chain one hop deep. More is a dependency tree, not an architecture.

⚠ Slow DQ source buried in composite

Silent IR failures mean you don't know what's Import vs DQ until users complain.

INSTEAD

Profile the DQ source before you compose. If it's slow, fix it upstream or import a subset.

Memory, refresh & cost implications

Memory is a hard ceiling

Memory vs. compute

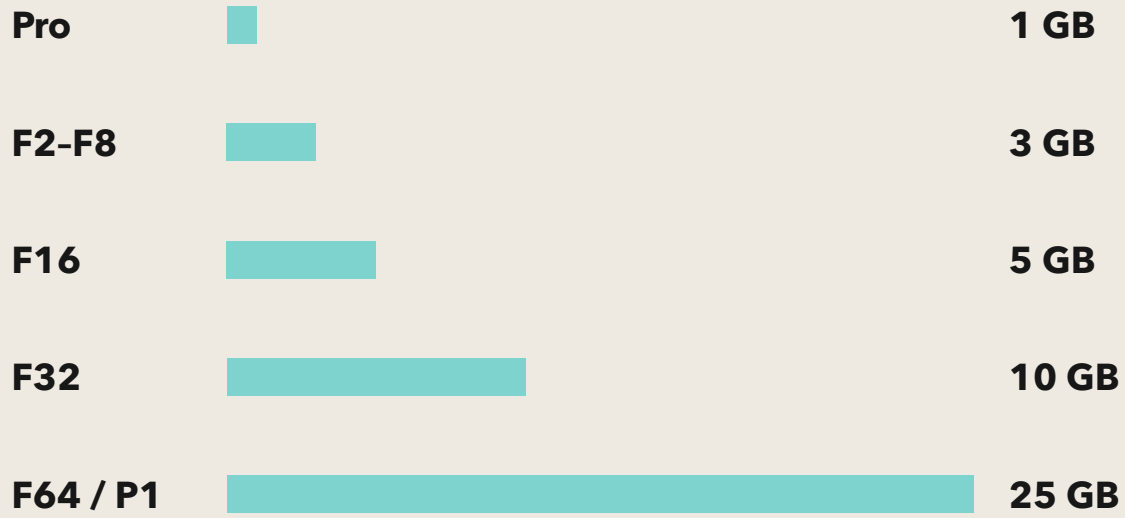
Compute is flexible. Memory is not.

Fabric smooths and bursts compute. Memory is a per-model ceiling set by your SKU — exceed it, and the refresh fails outright.

"Resource Governance: This operation was cancelled because there wasn't enough memory to finish running it."

Memory limits by SKU

Hard ceilings — not guidelines:



Tabular Editor

Storage mode shifts cost – it doesn't eliminate it



Import

Full model in VertiPaq
Largest memory footprint. Loaded at refresh and held until the next one.



Direct Lake

Columns on demand
Lower footprint at rest – but the same SKU memory ceiling applies once segments transcode.



DirectQuery

No VertiPaq footprint
Memory pressure disappears – but every user query now pays at the source system.

The "two copies" problem during refresh
Full Import refresh briefly holds two copies of the model in memory – the live one serving queries and the new one being processed. A model that fits at rest may still breach limits during refresh. Incremental refresh and partition strategies mitigate this.

Optimization levers – start with the diagnostic

Most apply regardless of mode. All of them beat switching modes as the first move.

- **Remove unused columns** – biggest single win, usually.
- **Reduce cardinality** of high-cardinality string columns.
- **Use aggregations** to offload large Import fact-tables to DirectQuery.
- **Incremental refresh** – only the current partition is replaced; lowers the "two copies" peak.
- **Split large models** – one giant model might be better as two purpose-built ones.

Decision framework

The default rule

STEP 1

START

1

…with Import.

It's the default for a reason – simplest to build, fastest to query, fewest footguns.

STEP 2

OPTIMIZE

2

…before you abandon it.

Incremental refresh. Partitioning. Column pruning. Aggregations. In that order.

STEP 3

ONLY THEN LEAVE

3

…with a measurable reason.

A clear, named problem that Import optimization cannot solve. Not a preference.

> **Import is the answer until something measurable tells you otherwise.**

Slow refresh is not a reason to switch modes – it's a reason to optimize. Preference is not a measurable problem.

Problem → Solution

IF YOU HAVE THIS PROBLEM		REACH FOR THIS SOLUTION
1 Data must be fresher than scheduled refresh allows	→	**Hybrid tables** Hot DQ partition for live data, Import for the rest
2 Dataset too large, or detail columns rarely queried	→	**Import + DQ on detail** Keep aggregates in memory, defer wide tables to source
3 Data already in OneLake as Delta tables	→	**Direct Lake over OneLake** Skip the Import refresh entirely – load columns on demand
4 Extend a central model without modifying it	→	**Proxy (Composite) model** DQ over the enterprise model, add team-specific logic on top
5 Source RLS too complex to duplicate in the model	→	**DirectQuery** Push security to the source – accept the query-time cost

Wrap-up & Q&A

Key takeaways

01

Import is the explicit default.

Every other mode needs a specific, evidence-based justification.

02

Direct Lake isn't magic.

It's VertiPaq with on-demand loading from Delta files – Fabric-only. Understand framing vs. transcoding.

03

DirectQuery is a targeted tool, not a default.

Expensive at scale on pay-as-you-go sources – measure query fan-out before committing.

04

Memory is a hard ceiling.

Plan your budget at 2× model size – Import refresh holds two copies in memory.

05

Composite and Hybrid solve specific problems.

The decision flowchart is a starting point, not a substitute for testing.

Resources

- Kurt Buhler's two-part storage mode series (Tabular Editor blog)
- Storage mode in Power BI semantic models (Microsoft docs)
- Direct Lake overview (Microsoft docs)
- Composite models (SQLBI whitepaper)
- Semantic Model Memory Errors series (Chris Webb blog)
- The three-part memory optimization series (Tabular Editor blog)



Download slides

Loved it? Learned something? Tell us! Share your feedback in just 1 minute



Rate This session

Choosing the Right Storage Mode: Import, Direct Lake, DirectQuery & Hybrid



Tabular Editor



Thank you for your time!

linkedin.com/in/gregbaldini/

linkedin.com/in/blindbaek/

Thank you to the sponsors!

























