LTIMindtree

Point of View

Microsoft Fabric Warehouse vs. Lakehouse

Competing Architectures, One Unified Vision

Background

Navigating the shifting terrain of enterprise data architecture

Enterprise data architecture has always faced a tough balancing act: scale versus structure, flexibility versus governance. For years, the industry moved between two ends of this spectrum.

Data Warehouses offered structure, governance, and performance for well-defined, transactional data. They served Business Intelligence needs well—but at the cost of agility and scalability. Then came Data Lakes, built for scale and variety. They made it possible to ingest vast amounts of raw, unstructured data, but often at the expense of control, consistency, and performance.

What started as a solution turned into a dilemma. Warehouses couldn't keep up with volume and variety. Lakes couldn't deliver trust and reliability. Enterprises were forced into trade-offs—losing speed in one direction, or quality in another.

This fractured model called for something new. That's where the Lakehouse architecture entered the picture—merging the best of both worlds. Open Storage Formats like Delta Lake instituted transactional consistency, schema evolution, data versioning to stored files and offered a slew of unique optimization techniques coupled with strong interoperability with major compute engines like Spark. This bridged many key gaps in the Lake and the Warehouse model of data platform design. It was a significant leap but still left enterprises piecing together technologies.

Now, Microsoft Fabric steps in, not with another tool, but with a strategy. It builds on everything the industry has learned from Warehouses, Lakes, and Lakehouses. By natively integrating Delta Lake into One Lake and unifying your entire data estate—structured or unstructured, historical or real-time, Fabric doesn't ask you to choose between past investments and future innovation. It connects them.

This isn't just the next stage in data architecture. It's a reset. Fabric is shifting the focus away from managing silos toward unlocking value, so enterprises can spend less time stitching systems together and more time turning data into action. That shift is powered in part by the benefits of Data Lakehouse architecture, unified storage, open formats, and flexibility—without giving up on performance or governance.

Introduction

Chapter I: From dilemma to direction
What Microsoft Fabric really solves

For years, enterprise architecture swung between extremes, first championing rigidly governed Warehouses, then swinging to free-form Data Lakes. Each had strengths. Each exposed serious gaps. The question is no longer which architecture to choose, but how to unify the benefits of both without inheriting their flaws.

That's where Microsoft Fabric steps in—not as a compromise, but as a consolidation of decades of lessons. Rather than forcing you to pick sides, Fabric enables a "Lake-centric Warehouse" approach. You get the performance, governance, and structure of a Data Warehouse—alongside the agility, scalability, and benefits of a Data Lakehouse architecture.

And the best part? You don't have to restructure your entire data estate to adopt it. Fabric integrates with what you already have. With One Lake as its foundation and Delta Lake as its backbone, you're free to build a future-ready architecture that's tailored to your enterprise—whether your workloads lean towards BI, AI/ML, or real-time analytics.

In the next section, we'll show you exactly how Microsoft Fabric pulls off this unification—and why it's changing the data conversation for good.

Chapter II: The great unifier
How MS Fabric brings unity in diversity

At first glance, the choice between a Data Warehouse and a Lakehouse in Microsoft Fabric might feel like choosing sides, structured versus semi-structured, governed versus flexible, transactional versus analytical. But that binary thinking no longer applies. Fabric doesn't ask you to choose, it unifies both.

This unity isn't an afterthought; it's a deliberate architectural decision. Fabric is designed as a "Lake-centric Warehouse", allowing both the Warehouse and Lakehouse to operate within the same ecosystem. Instead of drawing hard lines, Fabric brings these architecture under one roof, giving businesses the flexibility to adapt based on their needs.

Here's how that balance plays out:

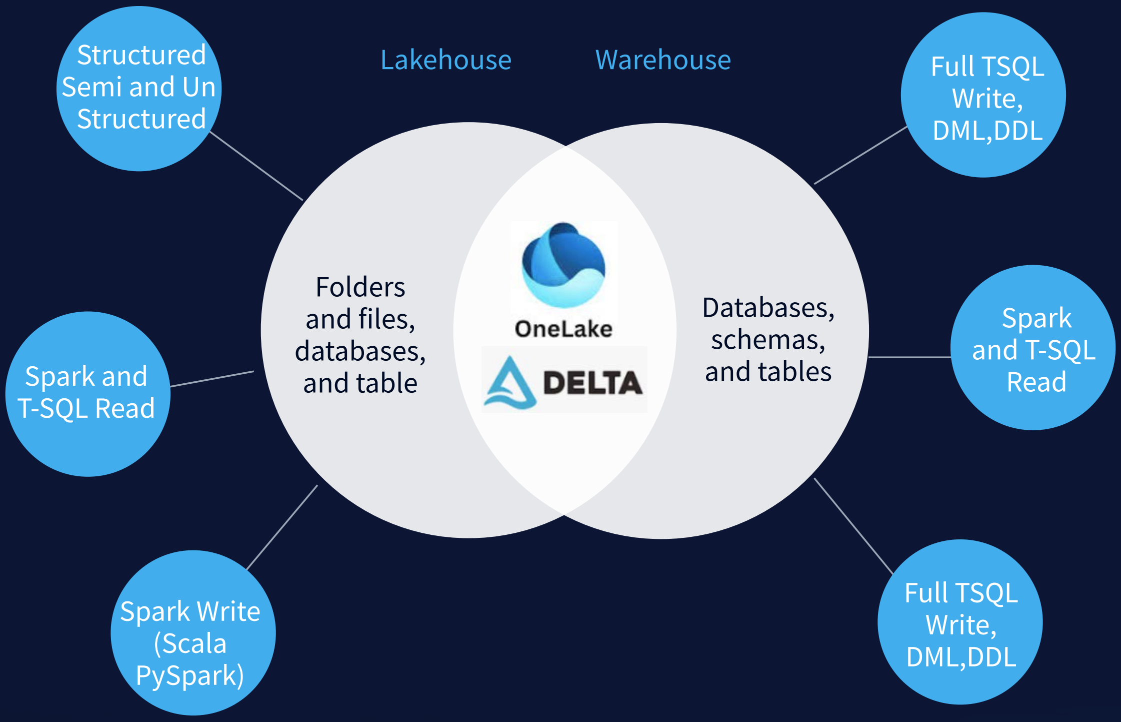
Fabric Data Warehouse

- End-to-end performance monitoring and built-in governance
- Full atomicity, consistency, isolation, and durability (ACID) compliance for transactional consistency
- Automatic performance optimization, no manual tuning required
- Data stored in One Lake using open Delta format
- Cross-database querying and semantic model support for analytics

Fabric Lakehouse

- Supports both structured and unstructured data at scale
- Auto-generated structured query language (SQL) analytics endpoint for querying Delta tables
- File-to-table automation for data engineers and scientists
- Direct Lake integration with Power BI for real-time analytics

Microsoft Fabric Eco System


Structured Semi and Un Structured
Lakehouse
Warehouse
Full TSQL Write, DML,DDL
Folders and files, databases, and table
OneLake
DELTA
Databases, schemas, and tables
Spark and T-SQL Read
Spark and T-SQL Read
Spark Write (Scala PySpark)
Full TSQL Write, DML,DDL

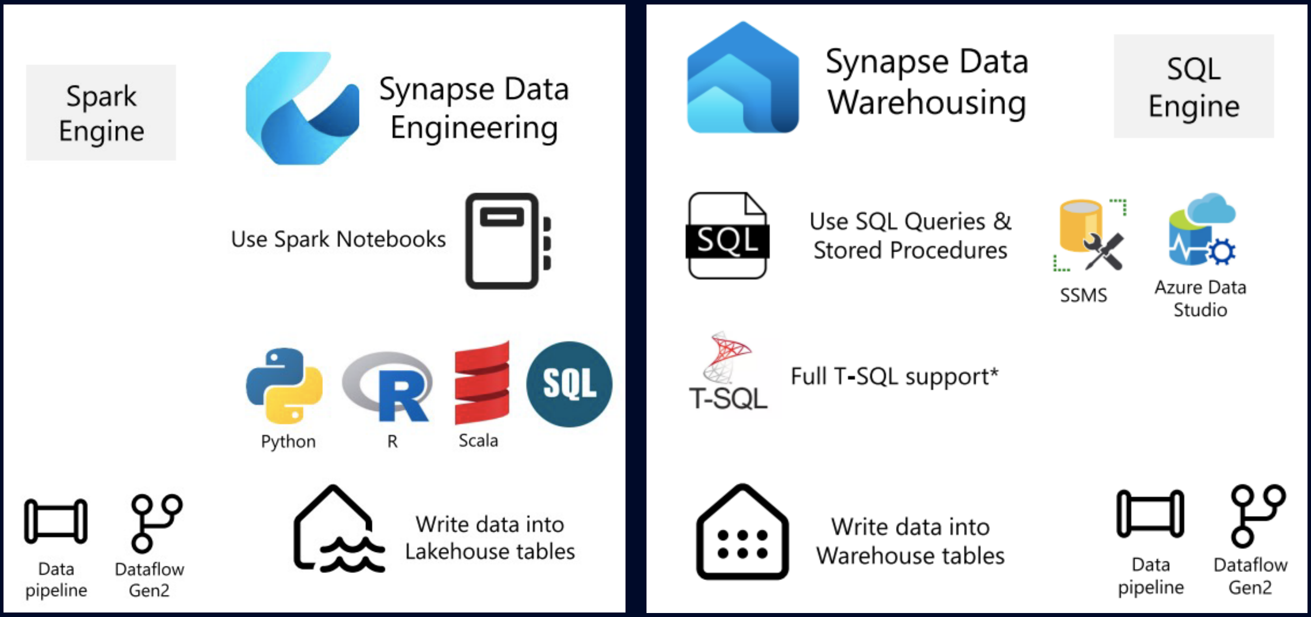


Fig 1: Delta unification of warehouse and lakehouse [2]

One Lake and Delta Unification

A paradigm shift

At the heart of Microsoft Fabric lies One Lake, a unified storage layer powered by Delta Lake. All experiences, whether in a Warehouse or a Lakehouse, generate and consume Delta tables, ensuring seamless interoperability.

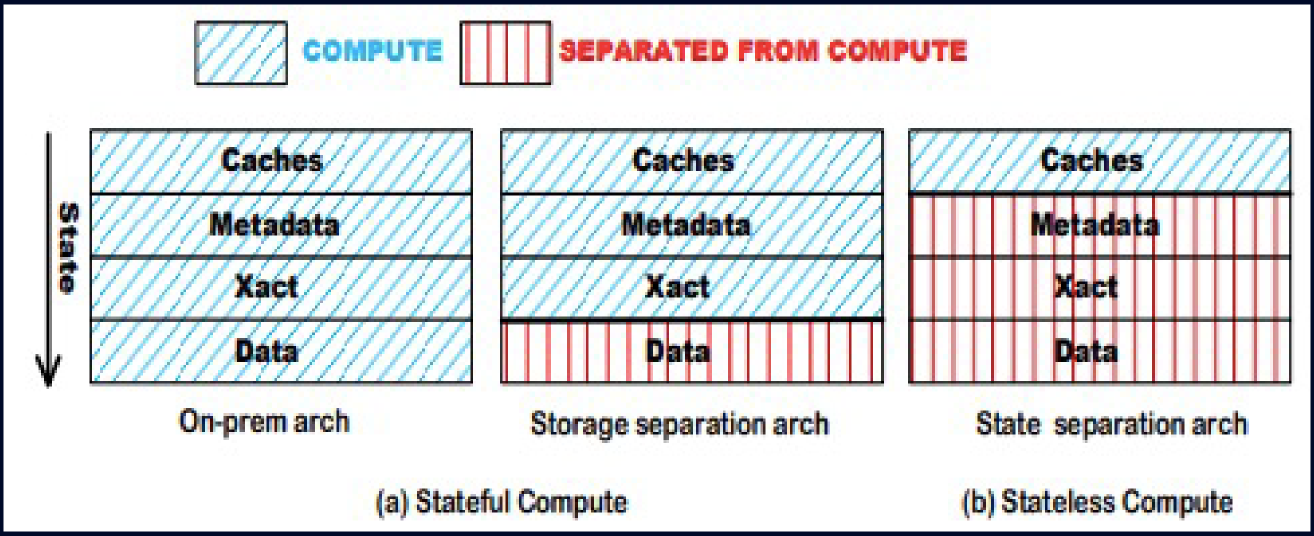
The role of Polaris engine

Sitting behind this unified experience is Polaris, the query engine powering Fabric's SQL experiences. Unlike legacy engines, Polaris is built cloud-first, optimizing for both performance and flexibility.

What sets it apart:

- Decouples compute from state, ensuring cloud-native execution
- Uses "data cells" for efficient parallel query execution
- Enables fine-grained task-level optimizations via a DAG-based query model

Polaris helps Fabric deliver warehouse-grade performance across both the traditional Warehouse and modern Lakehouse endpoints.



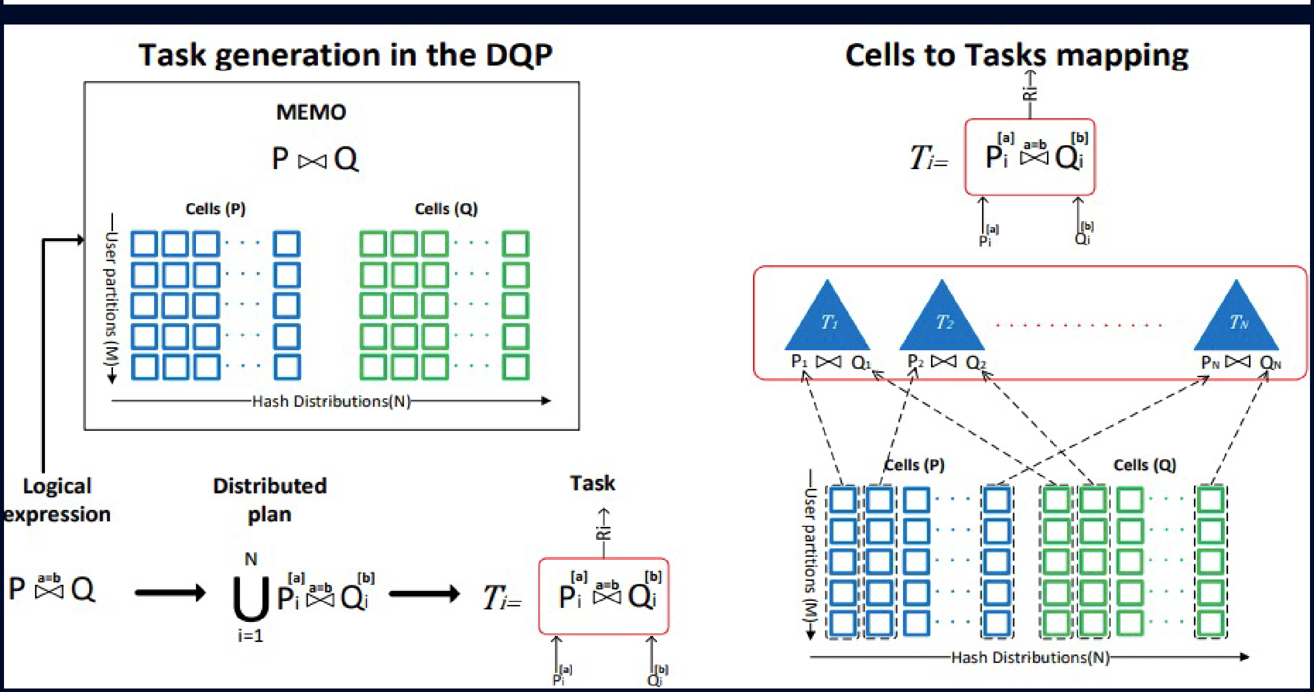


Fig 2: Polaris engine behind the scenes [3]

Delta Lake interoperability: No more lock-in

Fabric's choice of Delta Lake as its storage standard is strategic—not just for performance, but for openness. Delta is open-source and widely supported, allowing tools like Spark, Presto, Trino, and more to read and write data without vendor constraints.

This means your data remains portable, accessible, and shareable—within your enterprise or across clouds using Delta Sharing. It's another step in breaking down silos, without compromising on reliability or control.

Chapter III: Means to and end
Understanding Fabric's endpoints and optimization strategy

Fabric endpoints: Tailored experiences for every need

Microsoft Fabric isn't just about architectural convergence—it’s also about experience convergence. That’s where Fabric’s three key endpoints come in. Each is designed for a specific type of user and workload, ensuring the right interface for the right job.

Lakehouse Endpoint (for Spark and file-based analytics)

- Best for: Data engineers, data scientists, and AI/ML workloads
- Core functionality: Connects directly to One Lake for file-based processing using Spark, PySpark, and SQL
- Strengths: Supports structured and unstructured data, schema-on-read flexibility, and advanced transformations in notebooks
- Limitations: While you can write SQL queries, full T-SQL capabilities and transactional support are limited

SQL Analytics Endpoint (for SQL querying on Lakehouse Data)

- Best for: Analysts and business intelligence users needing SQL-based access to Delta tables
- Core functionality: Read-only SQL querying on Delta tables within the Lakehouse
- Strengths: Enables relational-style querying, integrates seamlessly with Power BI, and supports security policies for structured governance
- Limitations: No data manipulation language (DML) operations (insert/update/delete), no direct file access—queries limited to Delta tables

Data Warehouse Endpoint (for full T-SQL and transactional workloads)

- Best for: SQL developers and enterprise BI teams needing full data warehousing capabilities
- Core functionality: Complete T-SQL experience with multi-table transactions, DML support, and advanced schema management
- Strengths: Best for highly structured, governed, and transactional workloads, enabling cross-database ingestion and fine-grained access control
- Limitations: While built on Delta Lake, it is not optimized for Spark-based file processing

A quick clarification before we go deeper

Before diving into comparisons, it’s important to clarify something:

The SQL Analytics Endpoint does not equate to a full-fledged Data Warehouse experience. It delivers a “warehouse-like” read-only experience on top of Delta tables—but it’s not a substitute for full transactional capabilities. Just because a feature isn’t supported by the SQL Endpoint doesn’t mean the Lakehouse lacks it, it might simply be implemented through other interfaces like Spark notebooks or pipelines.

Fabric endpoints
Capability differences deep dive

Capabilities	SQL Endpoint only	Lakehouse Mode	Data Warehouse
Create/alter/ insert/update/ delete by t-sql	Not Supported (with some exceptions on ALTER being used for constraints and mask functions)	Supported by Spark SQL, PySpark or any other Lakehouse endpoint supported Languages	Supported
Primary, Foreign, Unique Key constraint (Not Enforced)	Limited Support	Not Supported by Spark	Add nullable columns of supported column data types. add or drop primary key, unique, and foreign_key column constraints, but only if the not enforced option has been specified
Column-level security at the data level	Supported only by GRANT T-SQL, for existing tables	Not Supported by Spark	Supported
Row Level Security at the data level	SQL analytics endpoint use existing tables. In the SQL analytics endpoint, you cannot create table, but you can create schema, create function, and create security policy	Not Supported by Spark	Supported
Dynamic Data Masking	Dynamic data masking works on SQL analytics endpoint. you can add masks to existing columns using alter table … alter column (exception)	Not Supported by Spark	Supported
SQL Granular Permission	Object-level-security can be managed using GRANT, REVOKE, and DENY T-SQL syntax. Users can be assigned to SQL roles, both custom and built-in database roles	Not Supported by Spark	Object-level-security can be managed using GRANT, REVOKE, and DENY T-SQL. Users can be assigned to SQL roles, both custom and built-in database roles
Time Travel	Not Supported	Not Supported by Spark	Supported
Restore in Place	Not Supported	Not Supported by Spark	Supports System defined restore points, user defined restore points and restore point retentions
Warehouse specific Monitoring	Supported by Metrics App for query activity, query insights and Dynamic Management Views	Not Supported by Spark	Supported by Metrics App for query activity, query insights and Dynamic Management Views
Collation Support	Not Supported	Not Supported by Spark	Currently, the only method available for creating a case-insensitive data warehouse is via REST API. All Fabric warehouses by default are configured with case-sensitive (CS) collation Latin1_General_100_ BIN2_UTF8. You can also create warehouses with case-insensitive (CI) collation – Latin1_ General_100_CI_ AS_KS_WS_SC_UTF8

fig 3: Capability comparisons across endpoints [4]

This above image highlights the key distinctions between Fabric's three endpoints. It helps teams align the right tool with the right workload and avoid common misinterpretations about what each experience is designed to do.

Analytics optimization in Lakehouse vs Warehouse

Strategic differentials in data loading

Data Ingestion Strategy Warehouse	
Use Case/ Scenario	**Method/Strategy**
• Need code-rich data ingestion operations. • Looking for highest data ingestion throughput possible. • When you need to add data ingestion as part of a Transact-SQL logic.	Use the COPY (Transact-SQL) statement
• Need code-free or low-code, robust data ingestion workflows. • Need recurrent run repeatedly, at a schedule. • Transfers that involves large volumes of data.	Data pipelines
• Code-free experience that allow custom transformations to source data before it's ingested. • These transformations include (but aren't limited to) changing data types, adding or removing columns, or using functions to produce calculated columns	Data Flows

Data Ingestion Strategy Warehouse	
Use Case/ Scenario	**Method/Strategy**
• Small file upload from local machine	Use Local file upload
• Small data or specific connector	Use Dataflows
• Large data source	Use Copy tool in pipelines
• Complex data transformations	Use Notebook code
• Streaming data	Use Event stream to stream data into Event house; enable OneLake availability and create a shortcut from Lakehouse
• Time-series data	Get data from Event house

Fig 4: Data loading strategy [4]

Warehouse-specific best practices for performance optimization

Capabilities	Optimization
Cold run (cold cache) performance	**Caching remains consistently active and operates seamlessly in the background** • The first 1-3 executions of a query perform noticeably slower than subsequent executions • If the first run's performance is crucial, try manually creating statistics • if the first run's performance is not critical, you can rely on automatic statistics that will be generated in the first query
Statistics	The Warehouse uses a query engine to create an execution plan for a given SQL query. When you submit a query, the query optimizer tries to enumerate all possible plans and choose the most efficient candidate. **User-defined statistics** • The user manually uses data definition language (DDL) syntax to create, update, and drop statistics as needed **Automatic statistics** • Engine automatically creates and maintains statistics at query time
Group insert statements into batches (avoid trickle inserts)	A one-time load to a small table with an INSERT statement,, might be the best approach depending on your needs. However, if you need to load thousands or millions of rows throughout the day, singleton INSERTS aren't optimal
Minimize transaction sizes	INSERT, UPDATE, and DELETE statements run in a transaction. When they fail, they must be rolled back. To reduce the potential for a long rollback, minimize transaction sizes whenever possible. • Minimizing transaction sizes can be done by dividing INSERT, UPDATE, and DELETE statements into parts • Consider using CTAS (Transact-SQL) to write the data you want to keep in a table rather than using DELETE
Utilize star schema data design	A star schema organizes data into fact tables and dimension tables. It facilitates analytical processing by denormalizing the data from highly normalized OLTP systems

Capabilities	Optimization
Reduce query result set sizes	Reducing query result set sizes helps you avoid client-side issues caused by large query results. • The SQL Query editor results sets are limited to the first 10,000 rows to avoid these issues in this browser-based UI. • If you need to return more than 10,000 rows, use SQL Server Management Studio (SSMS) or Azure Data Studio
Choose the best data type for performance	When defining your tables, use the smallest data type that supports your data as doing so will improve query performance. • This recommendation is important for CHAR and VARCHAR columns. • If the longest value in a column is 25 characters , then define your column as VARCHAR(25) • Use integer-based data types if possible. SORT, JOIN, and GROUP BY operations complete faster on integers than on character data
Data Compaction	Data compaction consolidates smaller Parquet files into fewer, larger files, which optimizes read operations. • The data compaction process is seamlessly integrated into the warehouse. As queries are executed, the system identifies tables that could benefit from compaction and performs necessary evaluations

Fig 5: Warehouse optimization[4]

SQL Endpoint-specific best practices for performance optimization

Key Performance Area	Optimization
Automatic metadata discovery Latency	• If you are observing increased latency for changes to sync between lake houses and SQL analytics endpoint, it could be due to large number of lake houses in one workspace. • In such a scenario, consider migrating each Lakehouse to a separate workspace as this allows automatic metadata discovery to scale
Lakehouse table maintenance operations	Parquet files are immutable by design. When there's an update or a delete operation,. • A Delta table will add new parquet files with the changeset, increasing the number of files over time, depending on frequency of updates and deletes. • If there's no maintenance scheduled, eventually, this pattern creates a read overhead. More discussed in later.
Committed Change latency in Lakehouse SQL Endpoint	We recommend initiating an on-demand metadata sync, triggered from the SQL query editor Refresh ribbon option. This option forces an on-demand metadata sync
Partition size considerations	The choice of partition column for a delta table in a Lakehouse also affects the time it takes to sync changes to SQL analytics endpoint • A column with high cardinality (mostly or entirely made of unique values) results in Many partitions. • A large number of partitions negatively impacts performance of the metadata discovery scan for changes. • If the cardinality of a column is high, choose another column for partitioning • The size of each partition can also affect performance. Our recommendation is to use a column that would result in a partition of at least (or close to) 1 GB

Fig 6: SQL endpoint optimization[4]

LTIMindtree

Lakehouse Delta-specific optimizations

Method	Key Function
Optimize	Consolidates multiple small Parquet files into large file. Big Data processing engines, and all Fabric engines, benefit from having larger files sizes. Having files of size above 128 MB, and optimally close to 1 GB
V-Order	Applies optimized sorting, encoding, and compression to Delta parquet files to enable fast read operations across all the Fabric engine
Vacuum	Removes old files no longer referenced by a Delta table log. Files need to be older than the retention threshold, and the default file retention threshold is seven days. All the delta tables in OneLake have the same retention period.

Fig 7: Delta optimization[4]

Important to know

- V-Order is enabled by default in Microsoft Fabric and in Apache Spark
- V-Order in Lakehouse can be controlled at Session Level, Table Level and while writing data frames
- When ZORDER and VORDER are used together, Apache Spark performs bin-compaction, ZORDER, VORDER sequentially
- In warehouse, the effect of V-Order on performance can vary depending on your table schemas, data volumes, query, and ingestion patterns
- V Order in Warehouse can be disabled once but irreversibly
- Disabling V-Order can be only useful for write-intensive warehouses

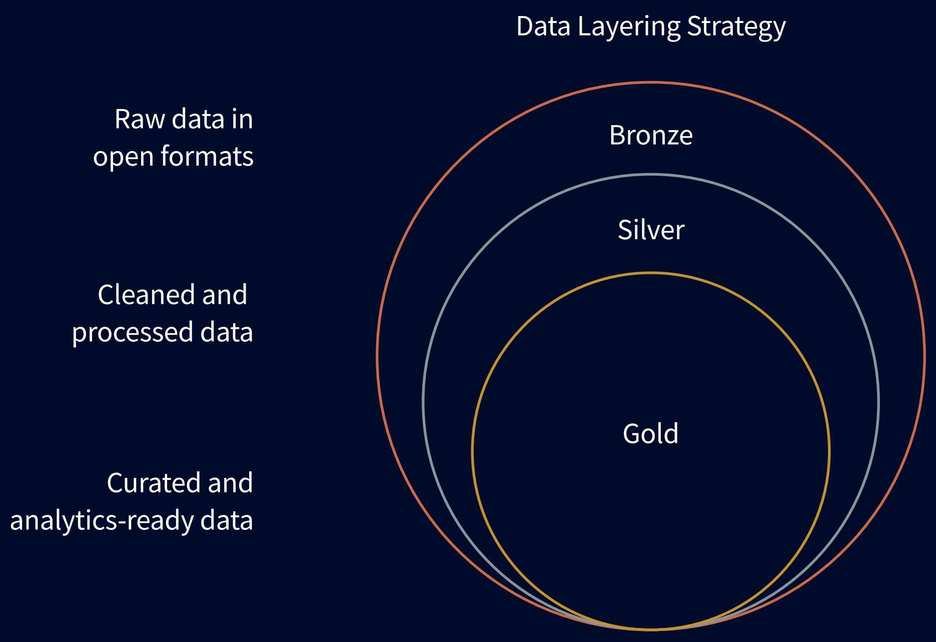
Chapter IV: Warehouse vs. Lakehouse finale Empowered Decisions

Final decision

After examining the nuances between Warehouse and Lakehouse, their endpoint capabilities, and performance optimizations, one insight becomes unmistakably clear: there's no universally perfect choice. The right solution hinges on your organisation's business objectives, the nature of your data, and the skill sets of your team.

This leads us to a critical architectural consideration—the Medallion Architecture. This widely adopted framework brings harmony between Lakehouse and Warehouse use cases by structuring data into layered zones. Within Microsoft Fabric, this approach ensures clarity, scalability, and performance across your analytics pipeline.

Choosing the right layer and storage



Key considerations

- Leverage shortcuts in Fabric to prevent redundant data duplication between Lakehouse and Warehouse.
- Enable Power BI Direct Lake mode to query data in real time, minimising latency and avoiding data movement.
- Embrace hybrid approaches by using the Lakehouse for raw ingestion and exploration, while relying on the Warehouse for structured, governed analytics.

Gold layer decision
Warehouse vs. Lakehouse

Criteria	Warehouse (Gold Layer)	Lakehouse (Gold Layer)
Best for	Structured, governed BI reporting	AI/ML, real-time and big data analytics
Data structure	Highly structured, relational	Structured and semi-structured
Querying and performance	Optimized for T-SQL and indexing	Optimized for Spark-based and SQL queries
Data modelling	Star schema, dimensional modelling	Flexible schema evolution
ETL/ELT workloads	Transactional ELT with DML operations	Batch and streaming ingestion
Integration with Power BI	Query mode (structured reports)	Direct Lake Mode (real-time insights)
Governance and security	Fine-grained access control	Lake-based ACLs, object-level security

Reference architecture patterns

Below are two visual patterns that represent the architectural placement of the Gold layer, depending on whether you lean toward the Warehouse or the Lakehouse:

Pattern 1: Gold layer in Warehouse

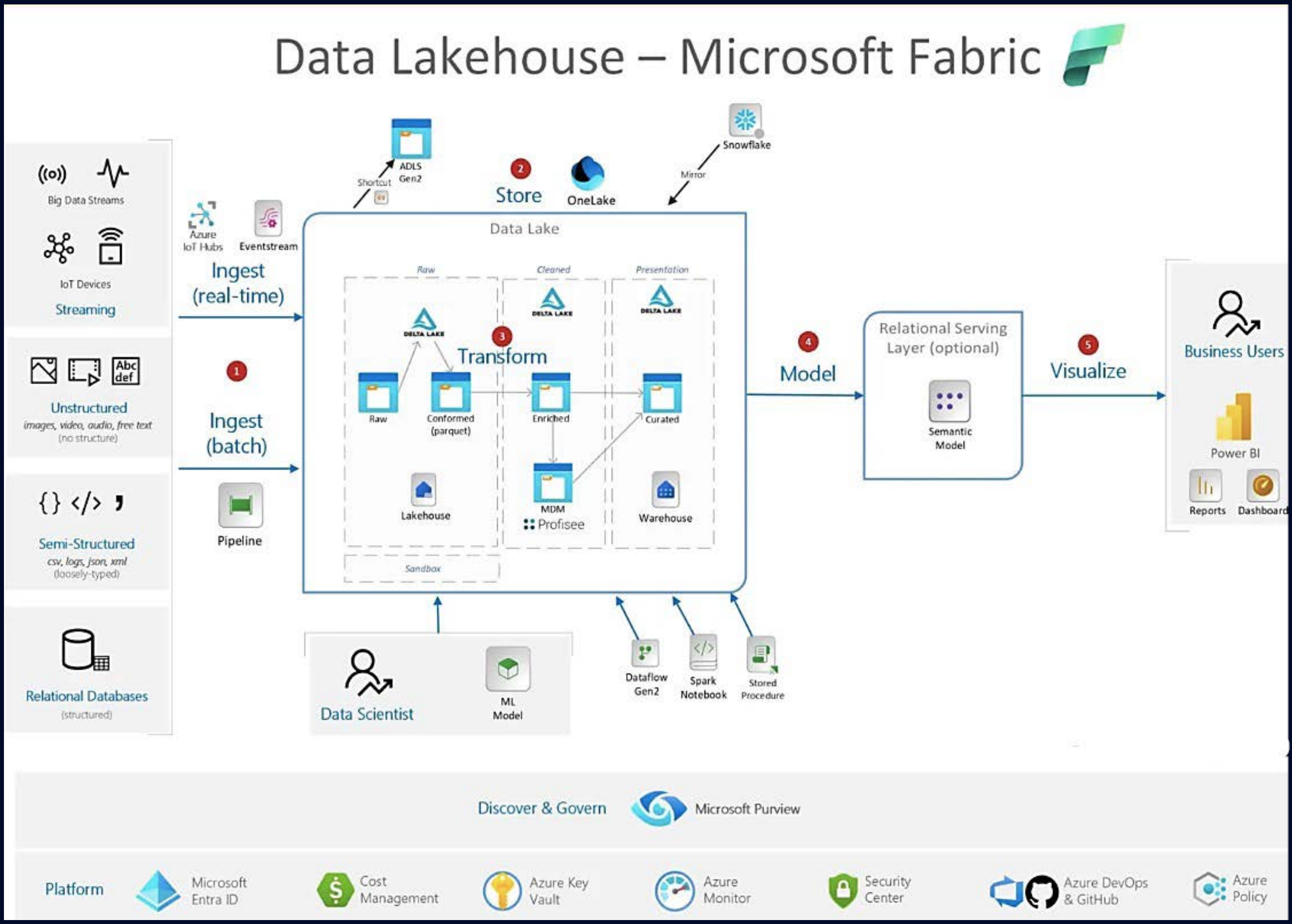


Fig 8: Gold layer in warehouse[5]

Pattern 1: Gold layer in Lakehouse

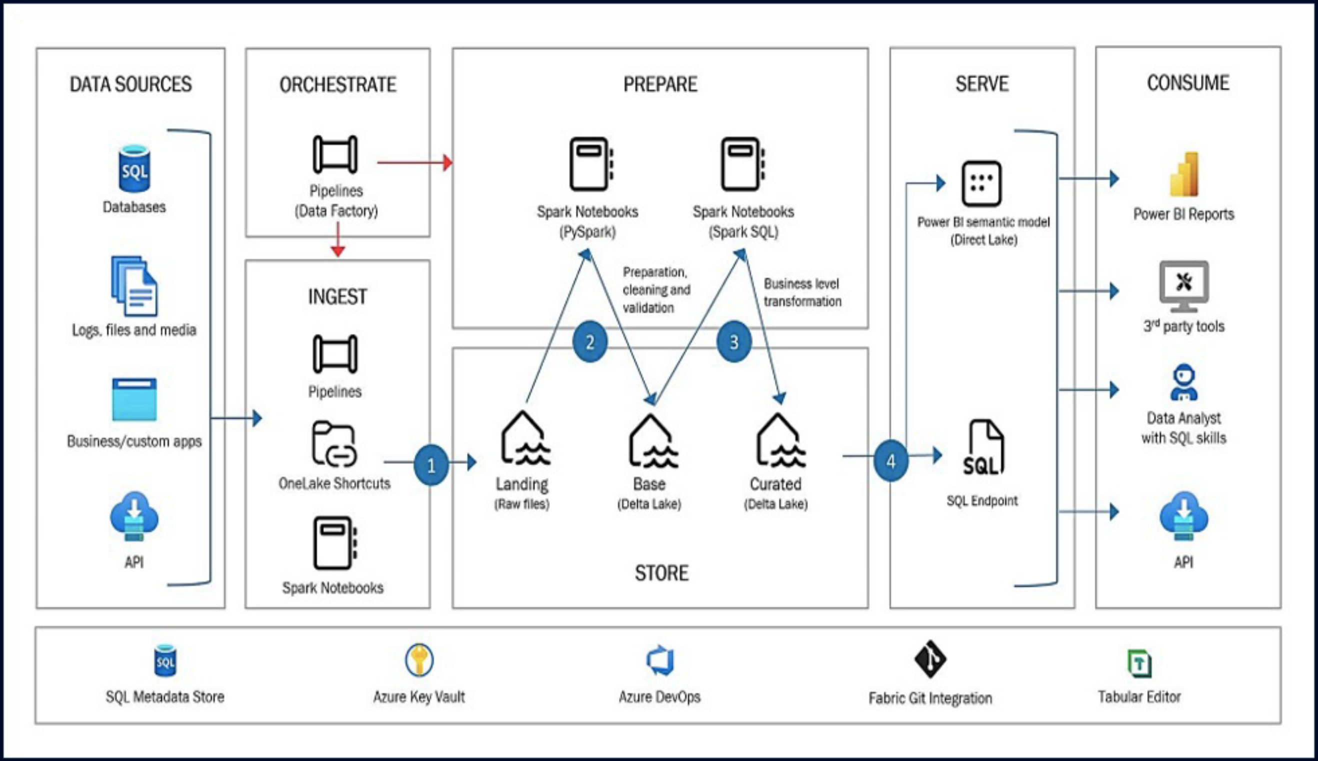


Fig 9: Gold layer in lakehouse[6]

Scenario-based decisions

Not every organisation has the same analytical priorities. The guidance below outlines when each option makes sense based on your team's skills and your data landscape.

Choose Fabric Data Warehouse if...

- Your team is SQL-centric: T-SQL is at the heart of your reporting and transformation workloads
- You work with structured transactional systems: Sources like enterprise resource planning (ERP) and customer relationship management (CRM) require multi-table transactions and indexing
- You're migrating from a legacy warehouse: Moving from SQL Server, Synapse, or similar platforms becomes smoother with Warehouse
- Self-service analytics is SQL-based: Your analysts depend on familiar querying and structured exploration
- You need robust recovery options: Warehouse supports restore points and backup strategies for governed recovery

Choose Fabric Lakehouse if...

- Your data comes in varied formats: JSON, Avro, ORC, or even images and raw logs are handled with ease
- Your use cases include AI, ML, or Spark transformations: Schema evolution and open formats make Lakehouse ideal
- You're working with streaming pipelines: Tools like Kafka or Event Hubs fit naturally into a Lakehouse setup
- You're migrating from cloud data lakes: Delta Lake ensures compatibility with big data ecosystems like Databricks or Hadoop
- Your team prefers flexible data models: Schema-on-read allows agility without rigid upfront modelling
- You need federated access and cross-workspace queries: Shortcuts and virtualisation in One Lake power seamless integrations

Conclusion

Microsoft Fabric isn't just another data platform—it's a unified ecosystem that brings together the strengths of both Data Warehouses and Lakehouses. Rather than replacing existing paradigms, Fabric modernizes and integrates them, offering flexibility without compromise. The decision to adopt Warehouse or Lakehouse isn't binary; it's architectural. The most effective strategies are shaped not by trends, but by purpose—by understanding your organisation's data needs, scalability goals, and operational context.

Ultimately, the architecture you choose should reflect your team's strengths. SQL-heavy teams will find familiarity and control in the Warehouse, enabling governed analytics and structured reporting. On the other hand, data engineers and AI/ML practitioners can harness the agility and scalability of the Lakehouse for streaming, real-time, and large-scale analytics. The key lies in striking the right balance, leveraging each component where it fits best and allowing Fabric to unify these layers into a coherent, purpose-driven solution.

References

[1]*Delta Lake: Up and Running: Modern Data Lakehouse Architectures with Delta Lake,* O'Reilly Publication of Book: https://www.oreilly.com/library/view/delta-lake-up/9781098139711/ch01.html

[2]*Microsoft Fabric: Lakehouse vs Warehouse, James Serra:* https://www.youtube.com/watch?v=34sI2e30JUM

[3]*POLARIS: The Distributed SQL Engine in Azure Synapse, Whitepaper:* https://www.vldb.org/pvldb/vol13/p3204-saborit.pdf

[4]*Microsoft Fabric decision guide: Choose between Warehouse and Lakehouse,* WilliamDAssafMSFT, ttorble, January 26, 2025: https://learn.microsoft.com/en-us/fabric/fundamentals/decision-guide-lakehouse-warehouse

[5]*Microsoft Fabric reference architecture,* James Serra, August 12, 2024: https://www.jamesserra.com/archive/2024/08/microsoft-fabric-reference-architecture/

[6]*From Ingest to Insights: Building robust Data Lakehouses with Microsoft Fabric,* Just Blindbæk, Twoday, November 14, 2023: https://www.twoday.dk/en/blog/from-ingest-to-insights-building-robust-data-lakehouses-with-microsoft-fabric-tag-da

About the author

Suman

Solution Architect, Data Architecture and Strategy, LTIMindtree

Suman is a Cloud Solution Architect specializing in Azure, with over 15 years of experience in Data and Analytics. He focuses on designing cloud-based architectures that support complex business goals, drive organizational modernization, and enable strategic roadmaps for building future-ready data landscapes. His work spans multiple domains, helping clients align their data platforms with evolving needs.