ProtoComet

INDEPENDENT TECHNOLOGY CONSULTING

CASE STUDY

Data Platform Modernization: Azure Synapse to Microsoft Fabric

A phased migration and re-architecture into a governed, Medallion-structured Lakehouse platform. Built for trust today, and self-service tomorrow.

Yogesh Verma · Founder & Principal Consultant, ProtoComet · 2026

Contents

1. Executive Summary

The organization's data platform grew up around Azure Synapse Analytics and an Azure MySQL Flexible Server datamart, serving as the backbone for reporting from Salesforce CRM, Shopify, internal APIs, and file-based sources into Power BI. Over several years of iterative build-out, the platform accumulated the design debt common to fast-moving analytics estates: business logic buried in database views, long monolithic pipelines, and no formal separation between raw, cleansed, and consumption-ready data.

ProtoComet partnered with the organization's data engineering team to migrate this estate to Microsoft Fabric and, in the same motion, rebuild it as a governed, enterprise-grade data platform. The engagement adopts the **Medallion Architecture** (Bronze → Silver → Gold) as its central design pattern, with a Fabric Lakehouse backed by OneLake replacing the RDBMS-centric datamart entirely. Delivery is sequenced across two migration iterations; a low-risk lift-and-shift followed by a structural refactor, so that reporting continuity is never put at risk while the underlying architecture is rebuilt.

This document covers Phase 0 (the **Synapse-to-Fabric migration**) and Phase 1 (building the **well-architected, certified data platform on Fabric**). Self-service analytics — where business users build their own reports directly on certified models without going through the data team — is Phase 2. It is deliberately treated here as a forward-looking roadmap item, scoped and delivered as a separate engagement once Phase 1 has landed. The platform is nonetheless being engineered from day one to make that next step straightforward: certified semantic models, access boundaries, and governed discoverability are foundational to Phase 1, not retrofitted later.

> **About Client**
> The client is a century-and-a-half-old professional membership association representing the architecture and design profession, serving a member base of more than 100,000 practitioners across the country and internationally. Alongside advocacy, credentialing, and continuing-education programs, the organization runs a multi-channel operation that spans member relationship management, e-commerce for publications and contract documents, and a steady stream of engagement data from its chapters and programs. That combination — a large, distributed membership base, commerce operations, and multiple source systems feeding a single view of the member and customer relationship — is what made the existing Synapse-and-MySQL platform increasingly difficult to scale, and what the Fabric-native redesign was built to support going forward.

1.1 Engagement at a Glance

Dimension	Detail
From	Azure Synapse Analytics + Azure MySQL Flexible Server (datamart)
To	Microsoft Fabric — OneLake, Lakehouse, Direct Lake semantic models
Architecture pattern	Medallion Architecture (Bronze → Silver → Gold)
Delivery approach	Two-iteration migration: Lift-and-Shift, then Refactor-and-Optimize

Dimension	Detail
Sources in scope	Salesforce CRM, Shopify, internal database, third party APIs, GA4, Azure Blob Storage (Excel drops)
Consumption today	Power BI, connected directly to the MySQL datamart schema. Manual lookup
Consumption target	Power BI, Excel, and Fabric Agents on certified Direct Lake semantic models
Self-service enablement	Part of Roadmap in Phase 2, scoped as a separate follow-on engagement

2. Current State Assessment

2.1 Current Architecture Overview

The existing data platform is built on Azure Synapse Analytics with Azure MySQL Flexible Server as the primary data store. The architecture handles data ingestion from multiple systems, applies transformations via pipelines and Python notebooks, and surfaces data to Power BI through a datamart schema in MySQL.

2.2 Current Architecture Components

Component	Technology	Role
Orchestration & Ingestion	Azure Synapse Pipelines	Schedule-triggered and storage-event-triggered pipelines fetching from Salesforce, Shopify, Azure Storage files, and other internal sources
Transformation	Synapse Notebooks (Python)	Data processing, business logic, cleaning and enrichment applied within notebook compute
Target Data Store	Azure MySQL Managed Database	All processed data lands here; one database staging raw ingested data, another is the DataMart, holding ready-to-consume data
Database Objects	MySQL Views	Heavy use of views and SQL scripts embedded in the database layer
Orchestration Pattern	Master + Child Pipelines	A master pipeline triggers multiple smaller child pipelines per source domain
Visualization	Power BI	Connects directly to MySQL datamart schema tables and views
File-Based Ingestion	Azure Blob Storage + Event Trigger	Excel files dropped in storage trigger dedicated pipelines

2.3 Identified Shortcomings

The following table documents the specific shortcomings of the current architecture and maps each to the improvement delivered by the proposed Fabric-native design:

#	Shortcoming	Consequence Today	Fabric-Native Improvement
1	RDBMS (MySQL) as a data engineering target	MySQL is optimized for OLTP workloads, not analytical query patterns. Columnar scans, large aggregations, and parallel reads are significantly slower than a purpose-built analytical store; storage costs are also higher at scale.	Replaced by Fabric Lakehouse (Delta Parquet on OneLake), a columnar, cloud-native analytical store built for exactly this workload.
2	Business logic embedded in MySQL views and scripts	Logic embedded in database objects is an anti-pattern — invisible to version control, difficult to test, and tightly coupled to the MySQL engine.	All transformation logic moves to Fabric Notebooks (PySpark / Spark SQL), versioned in Git, testable, and engine-agnostic.
3	No formal data architecture or layering	Absence of a layered architecture (staging → curated → consumption) means raw data, intermediate states, and consumption-ready data are commingled.	Medallion Architecture (Bronze → Silver → Gold) enforces explicit layering, each with a defined contract, owner, and quality bar.
4	Master-pipeline-drives-many-children pattern	A master pipeline that serially fans out to many long-running children creates bottlenecks, makes failure diagnosis difficult, and increases maintenance overhead.	Pipelines are refactored around use-case-aligned, independently deployable sets. Fabric Data Factory supports parallel, dependency-aware orchestration natively.
5	Lengthy monolithic pipelines per source	Single pipelines that connect, extract, transform, and load in one long sequence are fragile and hard to reuse across sources.	Ingestion pipelines (Bronze) are decoupled from transformation notebooks (Silver/Gold); each stage can be retried, monitored, and developed independently.
6	No data lineage or discoverability	There is no catalogue, endorsement, or discoverability layer. Consumers must already know which table to use, with no signal about quality or trustworthiness.	Microsoft Purview provides lineage, cataloguing, and sensitivity labelling. Certified semantic models carry endorsement metadata visible to every consumer.

3. Business Impact & Transformation Outcomes

Taken individually, the shortcomings above are technical. Taken together, they roll up into five outcome areas that matter to the business: performance, delivery speed, governability, trust, and resilience. The table below reframes the transformation in those terms: what today's constraint costs the organization, and what the Fabric-native redesign changes about it.

Outcome Area	Current State Constraint	Target State on Fabric	Business Value
Analytical Performance & Scalability	MySQL's OLTP engine strains under columnar scans and large aggregations as data volume grows; storage costs climb with scale.	Delta Parquet on OneLake — a columnar, cloud-native analytical store purpose-built for this workload profile.	Faster dashboards and headroom to grow data volume and history without another re-platforming exercise.
Engineering Velocity & Maintainability	Logic buried in MySQL views/procedures; monolithic, tightly-chained pipelines are hard to test, extend, or safely change.	Logic lives in version-controlled PySpark/Spark SQL notebooks; ingestion and transformation are decoupled, independently deployable stages.	Faster, safer delivery of new data sources and metrics, with fewer late-night pipeline incidents.
Architectural Integrity & Governance	No formal layering — raw, intermediate, and consumption-ready data commingle in a single schema.	Medallion Architecture enforces explicit Bronze / Silver / Gold contracts, each with a defined owner and quality bar.	A predictable, auditable data flow that new team members and auditors can reason about without tribal knowledge.
Data Trust & Discoverability	No catalogue or endorsement signal; users guess which of several similarly-named tables is authoritative.	Microsoft Purview lineage and cataloguing, paired with certified Fabric semantic models carrying a visible endorsement badge.	Business users trust the numbers in front of them, and the data team stops fielding “which table do I use” questions.
Operational Resilience	A master pipeline serially fans out to many long-running children; one failure is hard to isolate.	Fabric Data Factory parallel, dependency-aware orchestration, gated by a pipeline control table between stages.	Failures are isolated and diagnosable; reprocessing one domain no longer requires rerunning everything.

These are directional, architecture-grounded improvements rather than guaranteed figures — the actual performance and cost delta will be confirmed against production workloads during Iteration 2 validation. The point of the redesign is that every one of these constraints is structural today, and every one of them is removed by moving to a layered, Fabric-native platform.

4. Migration Strategy & Approach

4.1 Guiding Principle: Manual Recreation Over Automated Migration

The migration of Azure Synapse workloads to Microsoft Fabric is executed through deliberate manual recreation of all pipelines and notebooks in Fabric-native tooling, rather than relying on Microsoft's built-in Migration Assistant. Where applicable, native Fabric connectors are used in preference to custom-coded integrations, and Dataflow Gen2 replaces simpler transformation steps previously handled in notebooks. This is a deliberate choice to use the migration as an opportunity to clean up and properly structure workloads in Fabric from the ground up, rather than carry the existing debt forward.

Why manual recreation was chosen over the Migration Assistant:

- **Migration Assistant is still in preview:** Microsoft's built-in pipeline migration experience is not yet production-hardened, introducing avoidable risk.
- **Partial automation at best:** Pipelines are categorised as Ready, Needs Review, Coming Soon, or Unsupported; anything outside “Ready” still requires manual intervention.
- **Known blockers the tool cannot resolve:** Certain artefacts have no direct Fabric equivalent, and triggers are intentionally excluded from automated migration.
- **Connection mapping is manual regardless:** Even successfully migrated pipelines require every Synapse linked service to be manually mapped to a Fabric connection before use.
- **It's copy-and-adapt, not a direct move:** Microsoft's own guidance describes Synapse-to-Fabric migration as requiring validated runtime behaviour and realigned security patterns no matter which tooling is used.
- **Full control and native connectors:** Manual recreation lets the team restructure pipelines and adopt Fabric-native connectors (Salesforce, Shopify, Azure Blob Storage) and Dataflow Gen2 in place of custom Python REST calls and legacy Dataflow Gen1 components.

4.2 Two-Iteration Delivery Model

Migration is structured in two iterations to manage risk and preserve existing Power BI report continuity:

Iteration	Objective	Data Target	Power BI Impact
1 — Lift and Shift	Synapse pipelines and notebooks are recreated in Fabric as-is, with behaviour unchanged.	MySQL remains the target database.	None — reports are untouched. Risk is minimised by keeping the consumption layer static while the engine underneath changes.
2 — Fabric Native Data Platform	MySQL is decommissioned; all data lands natively in Fabric Lakehouse following the Medallion Architecture.	Fabric Lakehouse (OneLake), replacing MySQL entirely.	Reports are re-pointed to certified semantic models, with output validated for parity before cutover.

5. Target Architecture: A State-of-the-Art Data Platform on Fabric

5.1 Core Storage: Fabric Lakehouse

The Fabric Lakehouse is the primary data store — the optimal choice for this data engineering workload profile. It is backed by OneLake, a single, tenant-wide Delta Lake storage abstraction, and natively supports the Delta format (ACID transactions, time travel, schema enforcement) on top of Parquet columnar storage.

Other Fabric stores can be brought in as future use cases demand. For example, Fabric Eventhouse (KQL) suits real-time and streaming data, time-series, and high-frequency operational telemetry; Cosmos DB suits high-performance application-layer databases etc. and can be introduced as the platform's natural extension points.

5.2 Medallion Architecture Design

The Medallion pattern defines three explicit data layers within the Fabric Lakehouse. Each layer has its own data contract, quality standard, and set of permitted transformations. The diagram below shows how source systems flow through Bronze, Silver, and Gold into governed consumption.

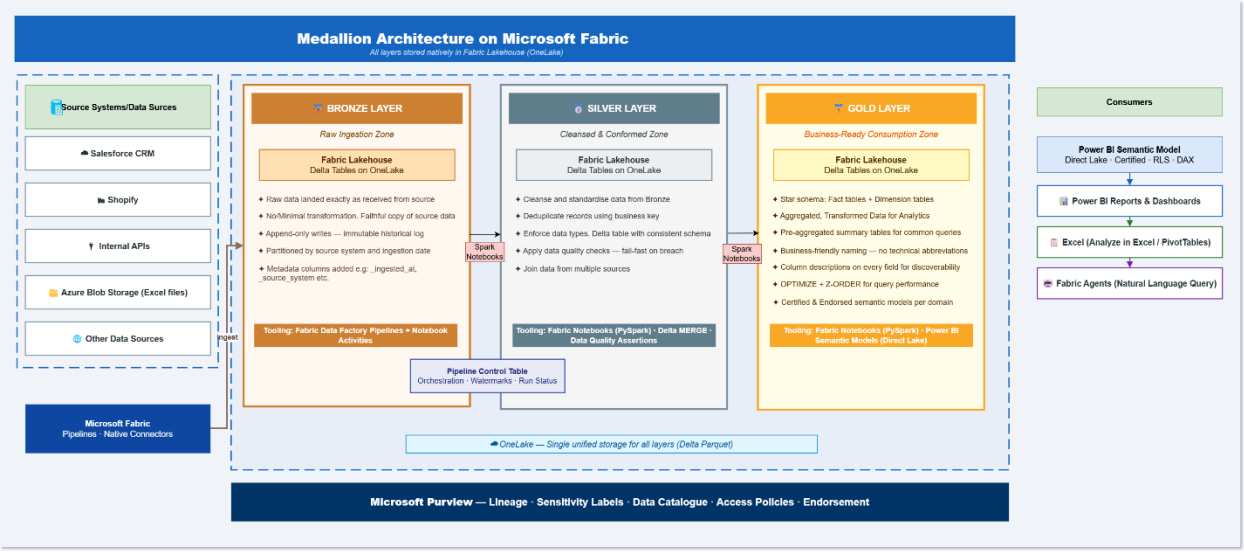

Figure 1 — Medallion Architecture on Microsoft Fabric: Bronze, Silver, and Gold layers landing natively in OneLake, feeding certified Direct Lake semantic models.

Layer	Purpose	Key Pointers	Tooling
BRONZE *Raw Ingestion Zone*	Land data exactly as received from source systems. No transformation. Immutable once written.	Schema-on-read; append-only where possible; ingestion timestamp and source metadata added at landing; partitioned by source and ingestion date.	Fabric Data Factory Pipelines (REST connectors), Lakehouse file API for Excel drops, notebooks for structured landing.
SILVER *Cleansed & Conformed Zone*	Clean, validate, deduplicate, and conform data into typed, consistent structures — the isolation buffer between source volatility and consumption.	Strongly typed columns, schema-on-write; deduplication, null handling, referential checks; conformed dimensions across sources; still entity-level, not metric-level.	Fabric Notebooks (PySpark), Delta MERGE for upserts, data-quality assertions that fail fast on threshold breach.
GOLD *Business-Ready Consumption Zone*	Business-aligned, metric-ready data modelled for consumption — what Power BI semantic models, analysts, and (in Phase 2) self-service users build from.	Star-schema modelled fact and dimension tables; business-friendly naming with a description on every column; pre-aggregated summary tables where query patterns demand it; org-standard date dimension.	Fabric Notebooks (PySpark), Delta tables with OPTIMIZE + Z-ORDER, certified Power BI semantic models via Direct Lake.

5.3 Certified Semantic Models. Why They Matter?

A Power BI semantic model (formerly a dataset) is a governed, reusable data model that sits between Gold Lakehouse tables and any consumer — a Power BI report, an Excel PivotTable, or a Fabric agent. In Direct Lake mode, the semantic model reads directly from Delta tables in the Lakehouse without importing or caching data, giving near-real-time freshness without import refresh overhead.

A certified semantic model is one formally endorsed via the Fabric endorsement workflow or by a data steward or domain owner through Purview, signalling to consumers that it is the authoritative, quality-assured version for that domain.

5.4 End-to-End Data Flow

The following describes the end-to-end data flow from source systems to the consumption-ready Gold layer:

Stage	Step	Fabric Component	Notes
Ingestion	Source systems (Salesforce, Shopify, internal APIs) → Bronze	Data Factory Pipeline + REST/HTTP connectors	Schedule-triggered. Append-only write. Metadata columns: _ingested_at, _source_system, _batch_id.
Ingestion	Excel files (Azure Storage) → Bronze	Data Factory Pipeline + Blob event trigger	Event-triggered on blob creation. Notebook validates schema before landing; rejects logged to a quarantine table.
Transformation	Bronze → Silver: cleanse, deduplicate, type-cast, conform	Fabric Notebook (PySpark)	Delta MERGE for idempotent upserts. Data-quality assertions fail loudly on threshold breach.
Transformation	Silver → Gold: star-schema modelling, metric derivation	Fabric Notebook (PySpark)	Fact/dimension tables written as Delta. Z-ORDER on high-cardinality filter columns.
Semantic Layer	Gold Delta tables → Direct Lake Semantic Model	Power BI Semantic Model (Direct Lake)	One model per business domain. Certified and endorsed; relationships, measures, and friendly naming defined here.
Orchestration	Pipeline dependency management and scheduling	Fabric Data Factory + Notebook Activities	Bronze triggers Silver on success; Silver triggers Gold on success. Failures alert via monitoring + email.

Stage	Step	Fabric Component	Notes
Governance	Sensitivity labelling, lineage, access control	Microsoft Purview + OneLake Security	Sensitivity labels on Gold tables/models. RLS at the semantic model layer. OneLake roles control Lakehouse access.

6. Governance, Trust & Discoverability

6.1 The Discoverability Problem, Solved

Today, finding the right table looks like this:

- Ask someone on the data team “which table has order data?”
- Get told “it's in the datamart schema, probably vw_orders_summary or maybe tbl_ord_final. Not sure which is latest.”
- Connect to MySQL, browse tables, find five tables with similar names.
- No way to know which is authoritative, which is stale, or what the columns mean.
- Write a query, get a number, and have no idea if it's trustworthy.

That is a discoverability problem, not a technical one, and it is exactly what Fabric's endorsement system is designed to solve.

When the data team builds a semantic model on the Gold layer and is confident it's production-ready, they endorse it. There are two levels:

- **Promoted:** Ready to use, and better than an unendorsed model.
- **Certified:** Formally reviewed and approved by a data steward; the authoritative source for that domain.

From the Power BI user's side, opening Power BI to build a new report surfaces these endorsed datasets directly. They don't need to know table names or which workspace something lives in. The certified badge tells them this is the one to build on.

6.2 Workspace & Lakehouse Organization

All three medallion layers are hosted within a single Fabric workspace per environment, each as a dedicated Lakehouse item, with a separate workspace for every environment (dev / test / prod), prefixed for easy identification. This keeps the platform footprint simple and operationally manageable while remaining fully compatible with splitting into per-layer workspaces later if scale or governance requirements demand it.

```
workspace-prod
  |--- lakehouse_bronze
  |--- lakehouse_silver
  |--- lakehouse_gold
```

Why this approach works well:

- **Clear separation of concerns:** Each layer is a distinct Lakehouse item with its own SQL analytics endpoint and storage namespace, obvious without relying on naming prefixes alone.
- **Structural safety, not convention:** A notebook writing to Bronze cannot accidentally write to Gold; the Gold SQL endpoint only ever shows Gold tables; a Direct Lake semantic model bound to lakehouse_gold cannot reference a Silver table even by accident.
- **Simplified operations:** One workspace means one place to monitor pipeline runs, notebook executions, capacity, and lineage while per-Lakehouse monitoring still shows storage and query volume independently.
- **Scalable governance:** Workspace-level roles (Admin, Member, Contributor, Viewer) provide the first layer of access control; fine-grained, table- and column-level access is governed through Purview's OneLake data access policies, enforced at the storage layer regardless of which tool — Power BI, Excel, SQL endpoint, or notebook — a user connects through.

7. Key Design Decisions

A handful of architectural questions came up repeatedly during design. They are captured here in condensed form as a record of the reasoning behind the platform, not just its shape.

7.1 Data Modeling — Star Schema

The Gold layer is modelled using a star schema — the standard technique for data warehouses and BI tools. A single central fact table, containing quantitative business events, is surrounded by dimension tables containing descriptive context.

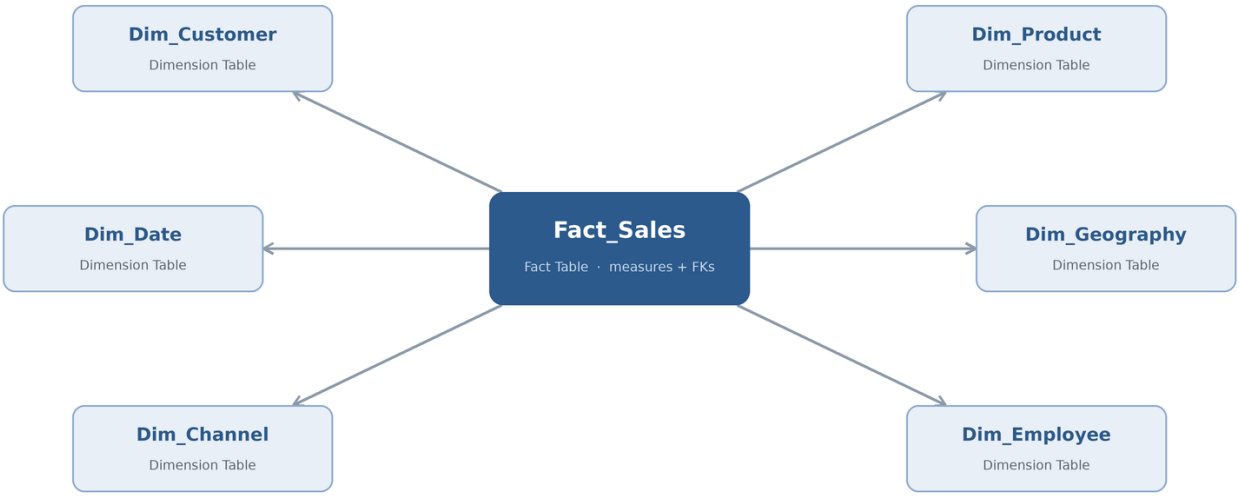

Figure 2 — Star schema: a central fact table (transactional events, measures) surrounded by descriptive dimension tables, one join away.

- **Fact table —** transactional events (an order placed, a payment made); numeric measures and foreign keys to dimensions; wide in row count, narrow in columns.
- **Dimension tables —** the descriptive context — who, what, when, where; smaller row counts, wider with descriptive attributes.

Where dimension entities need further normalisation, a Snowflake schema — an extension of the star schema in which dimension tables are broken into sub-dimensions — can be used selectively. In the example below, Customer and Product are normalised one level further (region/city, category/brand), while Date stays flat because it gains nothing from further splitting.

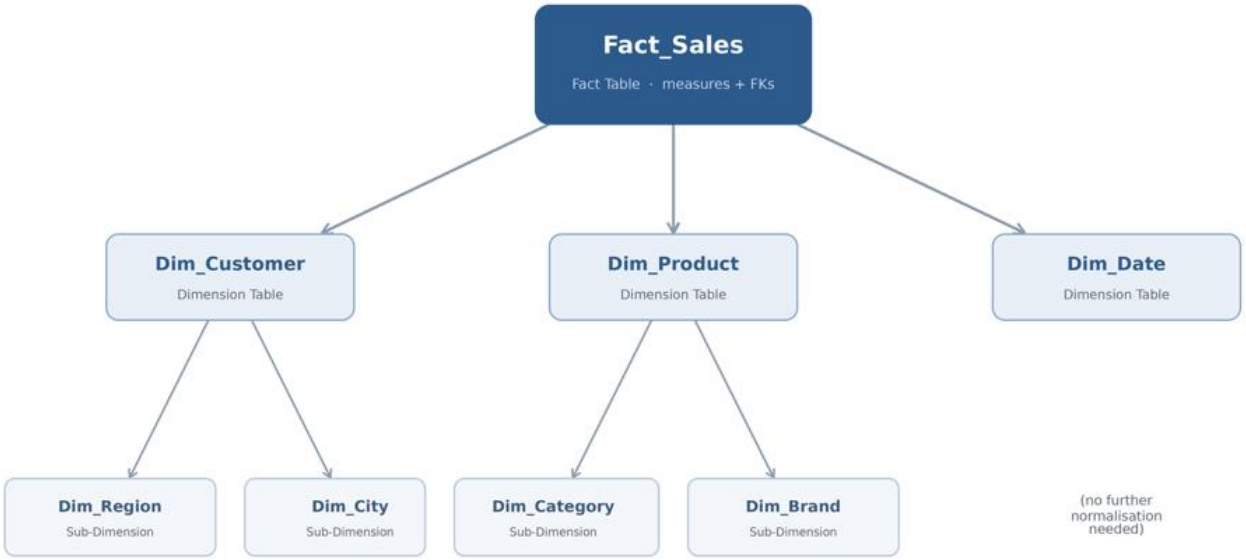

Figure 3 — Snowflake schema: selected dimension tables are normalised into sub-dimensions, trading some query simplicity for reduced redundancy.

> Star remains the default for the Gold layer as it is simpler to query and faster for Direct Lake and Power BI to consume. **Snowflake schema** is reserved for the specific dimensions where normalisation meaningfully reduces redundancy, not applied wholesale.

7.2 Efficient Source Extraction — Full vs. Incremental Copy

Fabric simplifies data movement with native support for bulk copy, incremental copy, and change data capture (CDC) replication, copying only new or changed data since the last run to save time and resources with minimal manual configuration.

Pattern	How it works	When to use it
Full copy	Every run copies all data from source to destination.	Reference and lookup tables, or any small table where a full pull every run is cheap.
Incremental copy	First run copies everything; subsequent runs move only new or changed data, tracked via watermark (rowversion, datetime, integer) or CDC.	Default pattern for transactional, high-volume tables.
Periodic full snapshot	A full pull and reconciliation on a longer cadence (weekly/monthly), layered on top of incremental copy.	Catching drift or late-arriving corrections that incremental watermarks can miss.

7.3 Decoupling Bronze and Silver

Bronze ingestion and Silver transformation are separate concerns. Bronze just lands raw data; whereas the silver transformation is only one of potentially several consumers of it. Tightly chaining them so Silver is always triggered directly by Bronze is an anti-pattern. Bronze and Silver pipelines are independently deployable and independently re-runnable. Silver does not start blindly on a schedule; it starts in response to a signal that Bronze completed successfully, written to a pipeline control table in the Lakehouse.

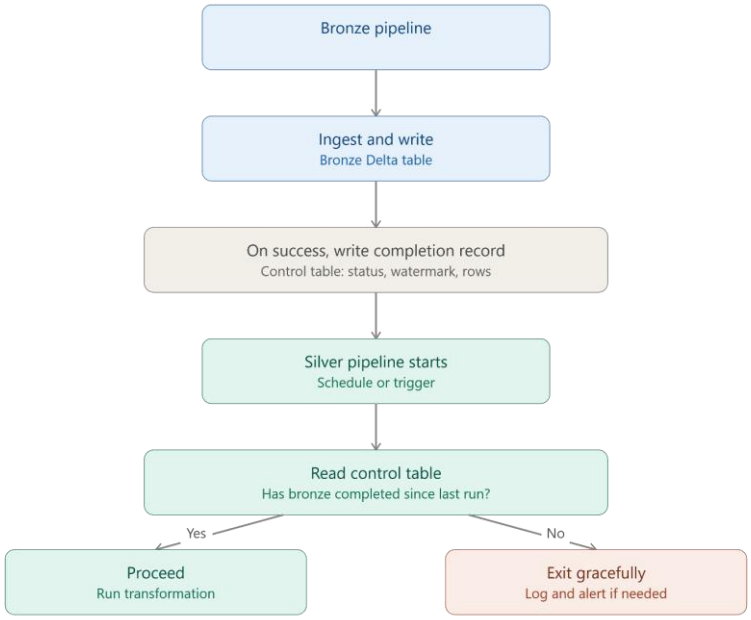

Figure 4 — Bronze and Silver are decoupled; a pipeline control table gates Silver's start on Bronze's confirmed completion.

7.4 Semantic Model Design Principles

The right boundary for a domain model is one certified semantic model per business subject area that a team owns end-to-end, not per source system, not per table, per business domain.

In the model	Not in the model
The fact table(s) for that domain	Raw or intermediate columns from Silver that business users don't need
All dimension tables those facts join to, with relationships defined explicitly	Technical key columns
DAX measures — the pre-defined, certified metric calculations	System metadata columns
Display folders organising fields into logical groups	—
Field descriptions — what each field means, includes, and excludes	—
RLS roles, as required	—

8. Delivery Roadmap

Delivery is sequenced across three phases. The first two are the subject of this document; the third — self-service enablement is deliberately kept out of scope here and shown only as the platform's next horizon.

Phase	Status	Goal	Definition of Done
Phase 0	**In Progress**	Move all Synapse workloads to Fabric; decommission Synapse instances.	Fabric pipelines and notebooks reproduce Synapse behaviour exactly; Power BI reports are unaffected.
Phase 1	**Next**	Raw/source data → clean, governed, well-modelled data fit for someone else to explore with little help.	A business user can open Power BI, find a certified semantic model, and the fields make sense without needing to know how the sausage was made.
Phase 2	**Roadmap — Future Phase**	Certified models → user-built views, exploration, and answers to their own questions, without the data team pre-building reports.	Users are answering their own questions and building their own views; the data team is no longer in the report-request queue for routine things.

8.1 Phase 0 — Synapse to Fabric Migration (Lift and Shift)

Goal: recreate all Synapse workloads in Fabric without changing behaviour or breaking Power BI reports. MySQL remains the data sink; risk is minimised by keeping the consumption layer untouched.

Task	Description
Create Fabric workspaces	Set up dev/test/prod workspaces with correct roles and Git integration.
Recreate Synapse pipelines in Fabric	Map each Synapse pipeline to an equivalent Fabric Data Factory pipeline, validating connectors against Fabric's native library and preferring native over custom where available.
Migrate Synapse notebooks	Python notebooks are largely portable. Validate PySpark API compatibility; adjust storage connection references to Fabric-managed connections.
Testing and data validation	Compare output data, row counts, and key metric values on copy tables before switching to the main tables.
Decommission Synapse	Disable Synapse pipelines; monitor Fabric for several weeks before formally decommissioning Synapse resources.

8.2 Phase 1 — Well-Architected Data Platform on Fabric

Goal: decommission MySQL as a data engineering target, implement the Medallion Architecture in Fabric Lakehouse, migrate business logic from MySQL views/procedures to Fabric notebooks, and build certified Gold-layer semantic models.

Task	Detail	Dependency
Implement Bronze Lakehouse layer	Create Lakehouse, define Bronze schemas, update ingestion pipelines to write to Lakehouse. Append-only, raw, partitioned by date.	Iteration 1 complete
Implement Silver transformation notebooks	PySpark notebooks read Bronze Delta tables, apply cleansing/deduplication/typing, and write to Silver via MERGE.	Bronze layer stable
Migrate MySQL views and SQL logic	Catalogue all MySQL views and stored procedures; rewrite as PySpark/Spark SQL; validate parity against MySQL results.	Silver layer stable
Implement Gold star-schema tables	Design and build fact/dimension tables per domain; optimize with OPTIMIZE + Z-ORDER; validate with stakeholders.	Silver stable + business sign-off
Build and certify semantic models	Create Direct Lake models on Gold tables; define relationships, DAX measures, field descriptions; endorse as Certified; configure RLS.	Gold layer stable
Migrate Power BI data source	Switch report data sources from MySQL to certified semantic models; validate output parity. Final MySQL dependency removed.	Semantic models certified
Decommission MySQL	Once Power BI is confirmed stable on Fabric semantic models, decommission Azure MySQL (or repurpose for non-BI applications).	Power BI migration validated

8.3 Looking Ahead: Phase 2 — Self-Service Enablement

> **ROADMAP — FUTURE PHASE**
> Self-service consumption is intentionally out of scope for this engagement. Once Phase 1 has delivered a certified, well-modelled Gold layer, Phase 2 will define how business users build their own views and answers directly on it, governed by the access boundaries and endorsement model already established in Phase 1. It will be covered in a separate technical document.

9. Closing

This engagement takes the organization's data platform from an RDBMS-centric estate carrying years of accumulated design debt to a governed, Fabric-native Lakehouse built on the Medallion Architecture. Phase 0 removes migration risk by moving workloads to Fabric without disturbing reporting; Phase 1 uses that foundation to rebuild the platform properly — layered, version-controlled, certified, and discoverable by design.

The result is a platform that is trustworthy today and ready for what comes next: when the organization is ready to open certified models to self-service exploration, the groundwork — governed access, endorsed semantic models, clean domain boundaries — will already be in place.