THE ORIGINAL MODEL

Before AI Models there was database Models and they are still just as important.

DATA DRIVEN INSIGHTS DRIVE GROWTH ESPECIALLY IN THE POST AI WORLD

Data Warehousing and Modelling

A CASE STUDY IN BUILDING SCALABLE, TRUSTWORTHY ANALYTICAL PLATFORMS

Author: Jayanth Kurup
Role: Database & Data Platform Consultant

Executive Summary

Data warehousing initiatives often begin with clear intent: consolidate data, enable reporting, and support better decision-making. Over time, however, many warehouses evolve into fragile systems that struggle to keep pace with business demands. Data pipelines break frequently, refresh cycles grow longer, and business intelligence teams spend more time working around data models than extracting insight from them. Metrics that should be foundational become sources of debate, eroding trust in the platform.

This whitepaper presents a practical case study in **data warehousing and modeling**, grounded in real-world consulting experience across SQL Server–based analytical platforms. It demonstrates how aligning data architecture, ETL design, and semantic modeling with actual business usage patterns leads to faster refresh cycles, more reliable metrics, and significantly lower long-term maintenance costs.

Rather than treating warehousing as a collection of tools, this paper emphasizes **systemic alignment**: how data is ingested, modeled, processed, and consumed must reflect both technical realities and business semantics. When these layers are designed cohesively, the warehouse becomes a stable analytical foundation rather than an operational burden.

1. The Customer Problem Space

1.1 A Changed ETL Landscape: From Packages to Platforms

The ETL landscape has undergone a significant shift over the past decade. Traditional data warehousing environments were built primarily around SSIS packages, tightly coupled to SQL Server and optimized for batch-oriented, relational data movement. These pipelines were often deployed as scheduled jobs, executed on fixed infrastructure, and designed around relatively stable source systems.

While this approach worked well in earlier generations of data warehouses, it no longer reflects how modern analytical platforms operate.

Today’s ETL environments are distributed, lake-centric, and heterogeneous by default.

From Monolithic Pipelines to Distributed Processing

In legacy architectures, ETL logic was encapsulated in SSIS packages that handled extraction, transformation, and loading end to end. These packages often combined business logic, orchestration, and error handling in a single artifact. As data volumes and source diversity increased, these designs became difficult to scale and maintain.

Modern platforms separate concerns more explicitly:

- Azure Data Factory and Dataflows handle orchestration and declarative transformations
- Data lakes act as persistent integration layers rather than transient staging areas
- PySpark and distributed compute enable large-scale transformations across semi-structured and unstructured data
- M / Power Query supports shaping and enrichment closer to the consumption layer

This shift replaces tightly bound pipelines with composable stages, each optimized for a specific role.

ETL Is No Longer Just Relational

SSIS was designed primarily for relational sources and targets. Today's data platforms must ingest:

- Application event streams
- API-based SaaS data
- Semi-structured formats such as JSON and Parquet
- Incremental and append-only datasets at scale

Data lakes provide the flexibility to store raw data in its native form, deferring schema enforcement until data is curated for analytical use. This decoupling fundamentally changes ETL design: transformation becomes iterative and layered rather than monolithic and destructive.

New Tools, New Failure Modes

The introduction of Azure Data Factory, PySpark, and lakehouse patterns solves many scalability problems—but it also introduces new risks when legacy ETL thinking is carried forward unchanged.

Common issues include:

- Recreating SSIS-style row-by-row logic in distributed engines
- Embedding business rules inconsistently across Spark jobs, Dataflows, and M queries
- Treating the data lake as an ungoverned dumping ground
- Losing visibility into lineage and failure points across loosely coupled stages

In other words, the tools have changed, but architectural discipline is still required.

SSIS Is Not Obsolete—But It Is No Longer Sufficient

SSIS remains effective for certain workloads: stable relational integrations, on-prem connectivity, and tightly controlled batch processes. The challenge arises when SSIS patterns are treated as the default approach for modern analytical platforms.

In a lake-centric architecture, ETL is no longer about "loading the warehouse once per day." It is about:

- Incremental ingestion
- Reprocessing at scale
- Supporting multiple downstream consumers
- Preserving raw data for future use cases

These requirements exceed what package-centric designs were meant to support.

ETL as an Ecosystem, Not a Tool Choice

The modern ETL landscape is best understood as an ecosystem rather than a single technology decision. Azure Data Factory, PySpark, M Query, and data lakes each serve a purpose—but only when their roles are clearly defined and aligned.

The fragility seen in many warehouses today is not caused by the new tools themselves, but by applying old mental models to new platforms. When ETL architecture evolves to reflect how data is now ingested, processed, and consumed, pipelines become more resilient, scalable, and easier to operate.

This shift sets the foundation for the rest of the warehouse design—and determines whether the platform becomes a durable asset or a growing operational liability.

1.2 BI Teams Fighting the Model Instead of Using It

Another recurring issue is the growing disconnect between data models and their consumers. Business intelligence teams often find themselves compensating for modeling deficiencies by embedding complex logic into reports, dashboards, or DAX measures. Over time, this leads to duplicated logic, inconsistent calculations, and performance degradation at the reporting layer.

When the semantic model does not reflect how the business actually thinks about its data, BI teams are forced to translate concepts repeatedly. Instead of focusing on analysis and insight, effort is spent reconciling definitions, rewriting measures, and explaining discrepancies between reports that ostensibly draw from the same data source.

This situation is both inefficient and risky. As logic proliferates across reports, maintaining consistency becomes increasingly difficult, and trust in the platform begins to erode.

1.3 Misaligned Dimensional Design and Business Semantics

At the heart of many warehouse challenges lies a misalignment between dimensional models and business definitions. Dimensions are created based on source system structures rather than business entities, hierarchies are incomplete or unintuitive, and measures lack clear ownership or meaning.

As a result, simple questions such as “What is revenue?” or “Who is the customer?” can yield different answers depending on the report or team involved. These discrepancies often lead to prolonged disputes during reviews and decision-making meetings, undermining confidence in the data warehouse as a single source of truth.

---

2. Consulting Approach: Warehouse Design as a System

The approach taken in this case study treated the data warehouse as an integrated system rather than a collection of independent components. ETL, storage, modelling, and consumption were evaluated together, with the explicit goal of aligning data structure, performance characteristics, and business meaning.

Three guiding principles shaped the engagement:

1. Design for analytical workload behaviour, not transactional convenience
2. Encode business semantics into the model, not into reports
3. Optimize for repeatability, recoverability, and scale

This mindset shifted the focus from short-term fixes to long-term stability.

---

3. Dimensional Modelling and Semantic Layer Design

3.1 Modelling for How the Business Thinks

Dimensional modelling was approached not as a technical exercise, but as a translation of business concepts into data structures. Fact tables were designed around measurable business events, while dimensions represented entities and attributes that business users naturally reference when asking questions.

Hierarchies were explicitly defined to reflect real analytical paths rather than implicit relationships inferred by tools. Measures were carefully scoped and documented, ensuring that each metric had a single, well-understood definition.

By aligning the model with business language, the semantic layer became intuitive for consumers, reducing the need for ad-hoc calculations and report-level logic.

3.2 Structuring the Semantic Layer for Stability

The semantic layer, implemented using SSAS or equivalent modelling constructs, was treated as a first-class architectural component. Processing strategies, aggregation behaviour, and dependency management were explicitly designed to support predictable refresh cycles and consistent query performance.

Rather than exposing raw tables to BI tools, the semantic layer acted as a controlled contract between data engineering and analytics teams. This separation reduced downstream breakage when source systems changed and allowed modelling decisions to be governed centrally.

3.3-Dimensional Modelling Is Not OLTP Design—and Mixing the Two Does Not Scale

One of the most common architectural mistakes in data warehousing is treating dimensional models as an extension of traditional database design. Teams attempt to reuse transactional tables, lightly denormalized schemas, or "hybrid" structures that are expected to serve both operational and analytical needs.

While this approach can appear efficient in the short term, it does not scale—either technically or organizationally.

Different Goals, Different Design Principles

Traditional database design, particularly in OLTP systems, is optimized for:

- Data integrity and consistency
- Efficient writes and updates
- Minimizing redundancy through normalization
- Supporting transactional workflows

Dimensional modelling, by contrast, is optimized for:

- Read-heavy analytical workloads
- Simple, predictable query patterns
- Business-oriented navigation paths
- Performance at scale through denormalization

These goals are not compatible within the same table structures.

OLTP tables are designed to *process events*. Dimensional models are designed to *analyze events*. Conflating the two leads to compromises that satisfy neither purpose well.

Why "Hybrid" Tables Become a Liability

Attempts to combine transactional and analytical requirements often result in tables that:

- Contain both normalized and denormalized attributes
- Mix surrogate keys with natural keys inconsistently
- Encode business logic implicitly rather than explicitly
- Grow wide and sparse as new reporting needs emerge

Initially, these designs reduce duplication. Over time, they increase complexity dramatically. Queries become harder to reason about, indexing strategies conflict, and performance degrades as access patterns diversify.

Most critically, every new requirement forces another compromise, accelerating architectural entropy.

The Cost of Forcing Analytics into OLTP Shapes

When dimensional principles are ignored or partially applied, downstream systems compensate:

- BI tools embed increasingly complex logic
- Measures duplicate transformations already present elsewhere
- Performance tuning shifts to reports rather than the model
- Definitions drift as different teams interpret the same structures differently

What appears to be a modelling shortcut becomes an ongoing operational burden.

Dimensional Models as Analytical Contracts

Dimensional models succeed because they act as **explicit analytical contracts**. Facts represent measurable business events. Dimensions provide stable, business-friendly context. Hierarchies are intentional. Grain is clearly defined.

This clarity enables:

- Predictable query performance
- Reusable business logic
- Easier onboarding of analysts and BI developers
- Long-term scalability as data volume and complexity grow

Trying to retrofit these properties into transactional tables undermines their value.

Separation Enables Scale

The most scalable analytical platforms embrace separation of concerns:

- OLTP systems remain optimized for transaction processing
- Warehouses and semantic layers are optimized for analysis
- Data is transformed intentionally between the two

This separation does not increase complexity—it localizes it. Each layer can evolve independently, and each can be optimized for its purpose without destabilizing the other.

Dimensional modelling is not an outdated technique. It is a deliberate response to analytical scale. When respected as such, it remains one of the most effective ways to build data platforms that are performant, trustworthy, and resilient over time.

---

4. ETL Architecture for Throughput and Repeatability

4.1 Moving Away from Row-by-Row Thinking

ETL redesign focused on replacing procedural, row-oriented logic with set-based, bulk-oriented patterns. Data movement was structured into clear stages, separating extraction, transformation, and loading responsibilities.

This approach not only improved throughput but also simplified error handling. Loads could be restarted from known checkpoints, and partial failures no longer required full pipeline re-execution.

4.2 Staging as a First-Class Concept

Staging layers were explicitly defined and persisted, serving both operational and analytical purposes. They provided isolation from source system volatility and enabled consistent transformation logic regardless of upstream changes.

Where appropriate, metadata-driven patterns were introduced to reduce duplication and improve maintainability. However, these patterns were applied judiciously, avoiding over-generalization that could obscure logic or complicate debugging.

4.3 Designing for Operational Reality

ETL processes were designed with operational concerns in mind, including load monitoring, failure recovery, and performance diagnostics. This reduced reliance on manual intervention and enabled proactive identification of issues before they impacted downstream consumers.

---

5. Performance Patterns for Analytical Workloads

5.1 Partitioning Based on Query Shape

Partitioning strategies were aligned with actual query patterns rather than arbitrary time boundaries. Fact tables were partitioned to support common filtering scenarios and incremental refresh strategies, reducing both load time and query latency.

This approach avoided the common pitfall of implementing partitioning as a blanket optimization without understanding its interaction with query execution plans.

5.2 Indexing for Read-Heavy Access

Indexing strategies were tailored to analytical workloads, balancing scan efficiency with manageable maintenance overhead. Indexes were designed to support common aggregation paths and join patterns, rather than mirroring OLTP indexing conventions.

By aligning indexing with query shape, the warehouse achieved predictable performance even as data volume increased.

---

6. Business Outcomes

The redesigned warehouse delivered tangible improvements across multiple dimensions. Refresh cycles became faster and more predictable, significantly reducing “broken pipeline” incidents. BI teams spent less time troubleshooting data issues and more time building insights, accelerating report delivery and iteration.

Most importantly, metrics became trustworthy. With business definitions encoded directly into the model, discrepancies between reports decreased, and stakeholders gained confidence in the data. Over time, this reduced the total cost of ownership for the BI platform, as clean models required less ongoing maintenance and fewer corrective interventions.

---

7. Why This Matters for Organizations

A data warehouse is not merely a reporting backend; it is a decision-making platform. When ETL pipelines are fragile and models are misaligned, organizations pay the price in delayed insights, wasted effort, and lost trust.

This case study demonstrates that thoughtful data warehousing and modelling—grounded in real workload behaviour and business semantics—can transform analytical platforms into durable assets that scale with organizational needs.

---

8. Conclusion

Successful data warehousing is less about tools and more about alignment. Aligning ETL design with throughput requirements, aligning dimensional models with business meaning, and aligning performance strategies with actual query behaviour are what distinguish robust warehouses from brittle ones.

By applying these principles systematically, organizations can move beyond firefighting and build analytical platforms that deliver consistent value over time.

References

1. Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. Wiley.
2. Inmon, W. H. (2005). *Building the Data Warehouse*. Wiley.
3. Microsoft. *Dimensional Modeling for SQL Server Analysis Services*. https://learn.microsoft.com/sql/analysis-services/multidimensional-models/dimensional-modeling
4. Microsoft. *SQL Server Data Warehousing and ETL Best Practices*. https://learn.microsoft.com/sql/data-warehouse

5. Microsoft. *Partitioning Strategies for Data Warehouses*. https://learn.microsoft.com/sql/relational-databases/partitions/partitioned-tables-and-indexes

6. Kimball Group. *Design Tips for ETL Systems*. https://www.kimballgroup.com