International Journal of Computer Science Engineering Techniques – Volume 8 Issue 5, September - 2024

Zero ETL Integration and Data Fabric for Analytics Warehouses

Eliminating Pipeline Friction in the Modern Analytical Stack

Sandeep Reddy Kaidhapuram

Independent Researcher, Austin, TX

sandeep.kaidha@gmail.com

Abstract - *For the last 30 years, the extract-transform-load pipeline has been the dominant way to move analytical data. Operational databases held data, analysts needed it in a warehouse, and it had to be transported. By the end of 2023, however, the total cost of designing, maintaining, monitoring, and repairing ETL pipelines has become one of the largest line items in enterprise data engineering budgets, and a growing number of vendors and practitioners are asking whether much of that cost can be avoided. This paper examines two converging trends aimed at reducing or eliminating this overhead: zero ETL integration, which allows operational stores and analytical engines to communicate without physically moving data, and data fabric architecture, which overlays metadata-driven intelligence onto distributed data assets to make them accessible in place. We trace the origin and current state of both approaches, propose a unified evaluation framework, and examine five representative enterprise analytics scenarios to identify where each pattern truly adds value and where marketing exceeds reality. The paper is written for data engineers, architects, and researchers who must make informed decisions in a rapidly evolving analytical data integration landscape.*

Keywords - *Zero ETL, data fabric, data warehouse, data lakehouse, change data capture, metadata management, data virtualization, analytics architecture, data engineering, cloud data platforms, data mesh, semantic layer, data integration, Apache Iceberg, real-time analytics*

1. Introduction

There is an old joke among data engineers that ETL stands for "Expect Trouble Late at Night." Anyone who has been paged at two in the morning because a pipeline broke mid-way through a nightly load, corrupting a dimension table on which half the company's dashboards depend, understands the sentiment. ETL pipelines are the urban traffic of the data world: everyone agrees they are necessary, plans around them, tolerates them, and every few years someone promises a revolutionary replacement that turns out to be a modest improvement.

Something genuinely different is happening in 2023. AWS announced zero-ETL integration between Aurora and Redshift at re:Invent 2022 and has progressively expanded that capability. Google BigQuery has long supported federated queries, and its integration with AlloyDB continues the same zero-copy theme. Snowflake's data sharing paradigm eliminates the need to move data between Snowflake accounts at all. Databricks has invested heavily in Delta Sharing and the Lakehouse architecture, unifying the warehouse and the lake on a single platform and reducing intermediate staging. These are not fringe startups with slideware; these are the largest cloud data platforms placing large bets that the ETL pipeline, as we have known it, has a limited future.

At the same time, the data fabric concept has grown from Gartner's 2019 coinage into a recognizable architectural pattern with both commercial offerings and open-source components. A data fabric proposes

an intelligent metadata layer that spans an organization's data assets wherever they live on-premises or cloud, structured or unstructured, operational or analytical and uses that metadata to automate discovery, governance, integration, and access. The data fabric does not aim to eliminate data movement outright; rather, it aims to make movement (or virtualization) intelligent and self-managing, reducing the manual engineering effort that currently consumes 40–60% of most data teams' time.

Zero ETL and data fabric are often discussed as separate topics, although they address the same pain: the high cost of moving data from operational systems to analytical ones. This paper examines the technical foundations of both, how they complement each other, and the scenarios in which either approach—or a combination of both—offers the best outcomes. The audience is not marketing teams but data engineers, architects, and researchers who need to make sound decisions about analytical integration strategy.

The analytical data landscape in late 2023 includes warehouses (Snowflake, Redshift, BigQuery), lakehouses (Databricks, Apache Iceberg on various engines), streaming platforms (Kafka, Flink, Confluent), and a growing class of real-time OLAP engines (ClickHouse, Apache Druid, StarRocks). Our focus is on the integration patterns that feed these systems rather than on the internal architecture of the systems themselves. We cite vendor-specific implementations where they are illustrative; the analysis itself is platform-agnostic.

2. Literature Review

2.1 The ETL Paradigm and Its Critics

ETL emerged as a formal discipline in the 1990s, concurrent with the data warehouse movement launched by Bill Inmon and Ralph Kimball. Both Inmon's (1992) corporate information factory and Kimball's (1996) dimensional modeling approach assumed a pipeline: data would be extracted from operational source systems, transformed into an analytical schema, and loaded into a dedicated warehouse. For its time, this assumption was reasonable. Operational databases and analytical engines used different schemas, encodings, and platforms, and operational OLTP systems lacked the compute capacity to run large analytical queries.

The problems with ETL are numerous and have only grown more acute with time. Pipelines are fragile: a schema change in a source system can silently break a downstream transformation, producing incorrect outputs that go undetected for days. Pipelines introduce latency: even with hourly batch scheduling, warehouse data is always at least an hour stale, and many enterprises still run daily or weekly loads. Pipelines are expensive to build and maintain: Fivetran's 2022 State of Data Engineering report found that data engineers spend 44% of their time on pipeline maintenance activities such as bug fixes, schema drift, and backfills. Pipelines also create inconsistent copies of data, giving rise to governance and lineage complications that grow worse as pipelines multiply.

The ELT (extract-load-transform) variant, popularized by cloud warehouses with elastic compute, shifted transformation to the target system but did not fundamentally change the integration architecture. Data still had to be extracted, staged, and loaded. Tools such as dbt brought software-engineering discipline to the transformation layer, but the extract and load stages remained pipeline-dependent, fragile, and labor intensive.

2.2 The Origin and Current State of Zero ETL

The term "zero ETL" was popularized by AWS CEO Adam Selipsky's re:Invent 2022 keynote, but the concept predates the label. Database replication, change data capture (CDC), and cross-database materialized views have existed in various forms for decades. What distinguishes the current generation of zero ETL offerings is the depth of integration between source and target systems: the cloud provider operates the replication machinery as a platform feature, freeing the customer from building and running it.

AWS's zero-ETL integration between Aurora (MySQL and PostgreSQL) and Redshift uses a CDC-based replication mechanism that is transparent to the user. Changes committed in Aurora are automatically propagated to Redshift, where they appear as queryable tables. The user configures no pipelines, staging areas, or transformation jobs. As of mid-2023 the integration is generally available for Aurora MySQL and in preview for Aurora PostgreSQL, with DynamoDB-to-Redshift support also available.

Google's approach is based on BigQuery's federated query engine, which can query data stored in Cloud SQL, Cloud Spanner, Bigtable, and other locations without physically moving it. Rather than replicating data into BigQuery with zero-copy semantics, BigQuery reaches out to the source at query time. This is data virtualization and has different performance characteristics: federated queries against OLTP sources are slower than queries against natively stored data and can, if poorly managed, degrade source system performance.

Snowflake's data sharing paradigm allows one Snowflake account to grant another account access to live, read-only data without replication. The underlying storage is shared through Snowflake's metadata layer, while the consumer account's compute resources handle query execution. This eliminates data movement entirely between Snowflake accounts but requires both sides to be on Snowflake, limiting cross-platform applicability.

Databricks has taken a different route through Lakehouse architecture and open table formats. By unifying the data lake and the warehouse on a single platform backed by Delta Lake (or Apache Iceberg), Databricks eliminates the lake-to-warehouse ETL step that has traditionally transported data between storage tiers. Delta Sharing extends the idea beyond Databricks as an open protocol for cross-organization data sharing. The open table format approach—supported by Iceberg, Delta, and Hudi—is particularly interesting because it decouples the storage format from the query engine, enabling multiple engines to read the same data without replication.

2.3 Data Fabric: Architecture and Concepts

Gartner defined data fabric as an architecture that uses continuous analytics over existing, discoverable, and inferred metadata assets to support the design, deployment, and consumption of integrated and reusable data across all environments. Unpacking that dense sentence, the core idea is that if an organization maintains rich, connected, and active metadata—information about what data exists, where it resides, who owns it, how it relates to other data, how fresh it is, who has accessed it, what quality issues exist—then many of the integration tasks that data engineers perform manually today can be automated or virtualized.

Data fabric is distinct from data mesh (Dehghani, 2022). Data mesh is an organizational paradigm emphasizing decentralization, domain-team ownership of data, and data-as-product thinking. Data fabric is a technical architecture that can operate under either centralized or decentralized organizational models. A data mesh organization could use data fabric technologies to implement the self-serve platform layer on

which domain teams build. In vendor marketing the two terms are frequently conflated, but they are not interchangeable.

A data fabric typically comprises a knowledge graph or metadata catalog maintaining relationships among all data assets; an active metadata engine that observes data access patterns, quality metrics, and lineage in real time; a data virtualization layer capable of federating queries across disparate sources without moving data; a policy engine that applies governance rules (access control, data masking, retention) consistently across environments; and an integration orchestration layer that can recommend and automate the optimal pipeline based on metadata analysis when physical data movement is required.

Commercial data fabric products include IBM Cloud Pak for Data, Informatica Intelligent Data Management Cloud, Talend Data Fabric, and Denodo (which emphasizes the virtualization pillar). Open-source components such as Apache Atlas (for metadata management), OpenMetadata, DataHub (LinkedIn's metadata platform), and Amundsen (Lyft's data discovery tool) allow organizations to assemble their own data fabric stack.

3. Methodology and Evaluation Framework

3.1 Research Approach

The evaluation framework was developed through three inputs: a structured review of vendor documentation, technical blogs, and academic literature on zero ETL, data fabric, and related integration patterns published between 2020 and September 2023; an analysis of publicly available benchmarks, case studies, and community experience reports from data-engineering practitioners; and comparative prototyping of representative integration scenarios using AWS (Aurora-Redshift zero ETL), Snowflake (data sharing), Databricks (Delta Sharing and Lakehouse), and Denodo (data virtualization) as representative platforms.

The framework is intended to be platform-agnostic. We reference specific vendor capabilities to illustrate patterns, but the analytical dimensions—latency, governance, operational cost, schema flexibility, cross-platform reach—apply across implementations. The goal is to give data architects a decision tool, not a product recommendation.

3.2 Taxonomy of Integration Patterns

Before examining zero ETL and data fabric, it is useful to establish a clear taxonomy of how data moves (or does not move) from operational sources to analytical targets. We identify four distinct patterns, each with distinct characteristics.

Table 1. Taxonomy of Analytical Integration Patterns

Pattern	Mechanism	Data Movement	Latency	Source Impact
Traditional ETL/ELT	Scheduled extraction, staging, transformation, loading	Full physical copy	Hours (batch)	Moderate (extraction queries)
CDC-Based Replication	Change data capture from transaction log; continuous apply to target	Incremental physical copy	Seconds to minutes	Low (log-based)

Pattern	Mechanism	Data Movement	Latency	Source Impact
Zero-Copy Sharing	Metadata pointer to shared storage; consumer queries in place	None (same storage layer)	Sub-second (query time only)	None
Query Virtualization	Federated query engine reaches source at query time	None (on-demand fetch)	Variable (depends on source)	High (query load on source)

Current zero-ETL implementations from the major cloud providers occupy patterns two and three in this taxonomy. AWS's Aurora-Redshift integration uses CDC-based replication; Snowflake's data sharing uses zero-copy sharing. Most data fabric architectures employ all four patterns, with the metadata layer selecting the optimal approach for each integration context.

3.3 Evaluation Dimensions

We evaluate integration approaches along six dimensions that consistently surface as decision criteria in enterprise data architecture projects.

Table 2. Evaluation Dimensions for Integration Approaches

Dimension	Description	Why It Matters
Data Freshness	Time between a change in the source and its availability for analytics	Determines whether real-time dashboards and operational analytics are feasible
Governance Continuity	Ability to maintain lineage, access control, and quality rules across the integration boundary	Regulatory compliance (GDPR, HIPAA, SOX) and internal audit requirements
Operational Burden	Engineering effort to build, monitor, and maintain the integration	Directly impacts data-team velocity and total cost of ownership
Source System Impact	Performance or availability degradation imposed on the operational source	Production stability; DBA teams will block integrations that threaten OLTP performance
Schema Evolution Tolerance	Handling of source schema changes (new columns, type changes, table renames)	Schema drift is the leading cause of pipeline failures in practice
Cross-Platform Reach	Ability to integrate across different databases, cloud providers, and data formats	Most enterprises have multi-vendor, multi-cloud data estates

3.4 Proposed Convergence Model: Fabric-Orchestrated Zero ETL

We argue that zero ETL and data fabric are not competing paradigms but complementary components of a modern integration architecture. Zero ETL provides the physical (or zero-copy) data-movement mechanisms that eliminate pipeline coding. Data fabric supplies the intelligence that decides when to move data, how to move it, and how to govern it across the integration boundary. We call this convergence fabric-orchestrated zero ETL.

In this architecture, the data fabric's metadata layer maintains a real-time map of all data assets, including their freshness, quality scores, governance classifications, and consumer demand patterns. When a new

analytical use case requires data from an operational source, the fabric evaluates the six dimensions above and selects the optimal integration pattern: CDC-based replication for high-frequency, latency-sensitive workloads; zero-copy sharing for same-platform analytics; query virtualization for low-volume, ad-hoc access; and traditional ELT reserved only for cases requiring heavy transformation before analysis.

This is more than theoretical. Organizations using Databricks with Unity Catalog or Snowflake with its governance capabilities and data sharing model are already operating elements of this architecture. What is typically missing is cross-platform metadata intelligence—the ability to orchestrate integration across multiple engines, clouds, and on-premises systems from a single governance plane. Open metadata projects such as OpenMetadata, DataHub, and OpenLineage are moving toward this goal, but as of September 2023 no single product delivers the full fabric-orchestrated zero ETL vision. Organizations assembling this architecture will need to integrate multiple components.

4. Results and Analysis

4.1 Scenario-Based Evaluation

To ground the framework in practice, five common enterprise analytics scenarios were examined, with the best-fitting integration pattern identified for each along the six evaluation dimensions. The summary appears in Table 3 and each scenario is discussed individually below.

Table 3. Scenario Fit for Zero ETL and Data Fabric Patterns

Scenario	Recommended Pattern	Zero ETL Fit	Data Fabric Value	Key Constraint
Operational reporting (real-time dashboards)	CDC replication or zero-copy	Strong	Medium — governance and routing	Source system load for CDC; platform lock-in for zero-copy
Cross-department ad-hoc analytics	Virtualization + selective caching	Moderate	Strong — discovery and access control	Query performance on heterogeneous sources
Regulatory data aggregation (SOX, GDPR)	CDC replication with audit layer	Strong	Strong — lineage and policy enforcement	Schema evolution across regulated sources
Data science feature engineering	Lakehouse unification	Strong	Medium — feature discovery and versioning	Compute cost for large-scale feature pipelines
Multi-cloud data sharing with partners	Open table formats + Delta Sharing	Moderate	Strong — cross-cloud governance	Partner platform compatibility; network egress costs

4.1.1 Operational Reporting and Real-Time Dashboards

This is the scenario where zero ETL integration delivers the clearest value. A retail business that wants a live dashboard showing current inventory levels, order volumes, and revenue cannot wait for a nightly

batch pipeline. CDC-based replication from the transactional database to the analytical warehouse provides sub-minute freshness with minimal source system impact, because log-based CDC reads the transaction log rather than querying tables directly.

This is exactly what AWS's Aurora-Redshift zero ETL integration is designed for. In our testing, changes committed in Aurora MySQL appeared in Redshift within 8–15 seconds at moderate write volumes. The operational burden was far lower than with a standard ETL pipeline: no extraction scripts, no staging tables, no scheduling. Schema changes in Aurora such as adding a column were propagated automatically to Redshift, eliminating the most common class of pipeline failure.

The constraint is platform scope. The Aurora-Redshift zero ETL feature works only within AWS. If the operational database runs on Azure SQL or Google Cloud SQL, or if the warehouse is Snowflake, this specific zero ETL integration is not available. Organizations with multi-cloud or multi-vendor estates need either a third-party CDC tool (such as Debezium, Fivetran, or Airbyte) or a data fabric layer that abstracts the differences. The fabric is valuable here because it can route integration through whichever mechanism is optimal for the source-target pair.

4.1.2 Cross-Departmental Ad-Hoc Analytics

When a finance analyst needs to combine revenue data from an ERP system with marketing attribution data from a cloud data platform and customer satisfaction scores from a survey tool, the challenge is not primarily latency but discovery and access. The analyst may not even know that the marketing attribution data exists, let alone where it lives or how to query it.

This is the scenario where data fabric genuinely shines. A metadata catalog listing all available data assets, along with descriptions, quality scores, ownership information, and sample data previews, allows the analyst to find what they need through search or recommendation rather than by emailing three different teams. A virtualization layer can then federate a query across the three sources without the analyst needing to understand the underlying storage systems. If the fabric observes that the combination is being accessed frequently, it can suggest materializing the join into a curated dataset, shifting from virtualization to physical replication.

Zero ETL is less central here because the value is in discovery and heterogeneous access, not in the elimination of a single flow. However, if the cross-departmental analytics requirement becomes recurring and latency-sensitive, a zero-ETL replication from the ERP to the analytical platform becomes the natural next step, recommended and governed by the fabric layer.

4.1.3 Regulatory Data Aggregation

SOX compliance in financial reporting, GDPR right-of-access requests, and HIPAA audit trails all require the ability to aggregate data across multiple systems with full lineage. This means being able to demonstrate exactly where every number in a regulatory report came from, how it was transformed, and who accessed it. Traditional ETL pipelines, when properly instrumented with lineage metadata, meet this requirement, but the instrumentation is laborious and tends to drift.

CDC-based zero ETL improves this picture by eliminating the transformation steps where lineage is most often lost. If data flows directly from source database to analytical warehouse through log-based replication, the lineage is straightforward: a one-to-one mapping from source table to target table. Transformations occur in the warehouse as SQL views or dbt models, which are version-controlled and auditable. Data fabric provides the governance layer: automatic classification of sensitive data (PII, PHI,

financial), policy-based access control that follows the data as it moves between environments, and lineage graphs that connect source systems to analytical outputs.

The combination of zero ETL replication and fabric governance is particularly strong for regulatory use cases, because it provides both the freshness required for timely reporting and the governance continuity required for audit—while reducing the manual burden of maintaining hand-coded pipeline lineage metadata.

4.1.4 Data Science Feature Engineering

Feature engineering, the process of transforming raw data into signals that machine-learning models can consume, is one of the most data-movement-intensive workflows in modern analytics. Data scientists typically pull data from multiple sources, join it in various ways, compute derived features, and persist the results into a feature store for training and serving. The lakehouse architecture simplifies this by consolidating raw data and structured analytical access on a single platform.

In a lakehouse, raw data lands in Delta Lake or Iceberg tables via CDC connectors or native ingestion interfaces (such as Databricks' AutoLoader), without a separate ETL integration step from operational sources to lake. Feature engineering runs as Spark or SQL transformations against the same platform. The feature store can serve both batch training and real-time inference. Compared to a traditional architecture where data traverses OLTP → data lake → warehouse → feature store through distinct pipeline stages, the source-to-feature path is substantially shorter.

Data fabric adds value in feature discovery and reuse. A metadata catalog that lists available features along with their computations, freshness, and usage across models avoids the familiar problem of individual data scientists independently computing slightly different versions of the same feature.

4.1.5 Multi-Cloud Data Sharing with Partners

Sharing data with external partners—suppliers, customers, research collaborators—is one of the fastest-growing integration use cases and historically one of the hardest to handle cleanly. The common approaches include SFTP file transfers, API-based data pulls, and ETL pipelines that push data into a partner's system. Each method involves data movement, governance ambiguity (who controls access after the data is shared?), and significant operational overhead.

Open table formats offer a compelling alternative. Apache Iceberg and Delta Lake both provide a storage layer that can be read directly by multiple engines. Delta Sharing is an open protocol that allows a data producer to grant read access to specific datasets without replication; the consumer can query the data using their choice of engine (Spark, Pandas, Trino, or any engine with a Delta Sharing connector). The data stays in the producer's storage, which reduces egress costs and maintains governance control.

The constraint is that not all partners operate on compatible engines or cloud platforms. Cross-cloud data transfer still incurs egress charges per query, which can be prohibitive for large datasets. Data fabric governance is essential here for tracking which partners have shared which datasets, enforcing retention and revocation policies, and monitoring access patterns across organizational boundaries.

4.2 Comparative Performance Observations

During prototyping we recorded key performance and operational metrics across the integration patterns. The numbers below reflect observations from controlled environments and should be read as indicative rather than definitive.

Table 4. Comparative Performance and Operational Metrics

Metric	Traditional ETL	CDC Zero ETL	Zero-Copy Sharing	Query Virtualization
Data freshness (typical)	1–24 hours	10–60 seconds	Real-time (query-time)	Real-time (query-time)
Setup effort	2–6 weeks	1–3 days	< 1 day (same platform)	1–2 weeks
Ongoing maintenance	High (schema drift, failures)	Low (platform-managed)	Minimal	Medium (source tuning)
Source system impact	Moderate (extraction queries)	Low (log-based)	None	High (query load)
Cross-platform support	Broad (any source/target)	Limited (vendor-specific pairs)	Limited (same platform)	Broad (with connectors)
Governance integration	Manual (lineage instrumentation)	Semi-automated	Native (platform governance)	Variable (depends on product)

The most striking observation is the difference in setup and ongoing maintenance cost. Building a conventional ETL pipeline for each source-target pair required 2–6 weeks of engineering effort, including schema mapping, transformation logic, error handling, monitoring, and documentation. CDC-based zero ETL integrations required 1–3 days, most of which was configuration and testing rather than coding. Zero-copy sharing required minutes to hours. The maintenance differential is even larger over the lifetime of an integration: ETL pipelines demand continual attention to accommodate schema changes, resolve failures, and manage performance, while platform-managed zero ETL integrations absorb most of this burden automatically.

The trade-off, as expected, is cross-platform flexibility. ETL and query virtualization can work across arbitrary combinations of sources and targets. Zero ETL integrations currently work only for specific vendor pairs (Aurora-Redshift, DynamoDB-Redshift, Snowflake-to-Snowflake). This limitation is the single largest barrier to enterprise adoption of pure zero ETL approaches, because most enterprises have heterogeneous data estates spanning multiple databases, cloud providers, and SaaS applications. A pure zero ETL strategy is feasible only for organizations whose data is heavily concentrated on a single cloud provider, which is a small fraction of enterprises in practice.

Governance integration likewise varied significantly. Zero-copy sharing inherits the governance model of the underlying platform; Snowflake's data sharing, for example, respects the row-level access policies and dynamic data-masking rules defined on the provider account. CDC-based zero ETL replicates data physically, so governance controls must be configured on the target independently of the source. Traditional ETL, in the worst case, requires governance metadata to be hand-instrumented in pipeline code, with each pipeline's lineage entirely dependent on the discipline of the engineer who built it. Query virtualization sits in between, depending on whether the virtualization product includes its own governance layer (Denodo does; basic federated queries often do not).

4.3 The Remaining Role of ETL

It would be incorrect to claim that ETL is disappearing entirely. Even in an architecture that maximally exploits zero ETL and data fabric capabilities, certain scenarios continue to require physical data movement with transformation.

Complex business-logic transformations that span multiple source systems are often not expressible as simple CDC replications or virtualized queries. Consider currency conversions applied to financial data across geographies, customer matching algorithms that deduplicate records across CRM and marketing databases, or hierarchical rollups that cross organizational structures. These require a transformation engine such as dbt, Spark, or a purpose-built integration tool capable of heavy computation.

Historical data migration—moving years of archived data into a new analytical platform—is inherently a batch operation that does not align with CDC or real-time patterns. Compliance-driven data archiving, in which data must be moved to long-term storage in a specific format with specific retention guarantees, similarly requires physical movement and transformation.

ETL does not disappear; it shrinks. In a fabric-orchestrated zero ETL architecture, conventional pipelines are reserved for cases that genuinely require transformation, while most integration is handled by platform-native replication, zero-copy sharing, or intelligent virtualization.

4.4 Organizational and Cultural Implications

The shift from pipeline-centric to platform-centric integration has organizational consequences that mirror those observed in previous architectural transitions. Data engineering teams that have built their expertise and identity around pipeline development may feel threatened by zero ETL announcements that appear to make pipeline-building skills obsolete. This anxiety is understandable but misplaced. The need for data engineering talent does not decrease; the work itself changes. Instead of building and maintaining extraction scripts, engineers increasingly design governance models, define semantic layers, tune CDC performance, manage schema evolution strategies, and build the metadata infrastructure on which fabric-orchestrated integration depends.

Organizations that fail to manage this transition well—treating zero ETL as a cost-cutting opportunity to shrink data-engineering headcount—tend to lose exactly the deep data knowledge they need to build the governance and metadata layers that make zero ETL work at scale. The better-performing organizations treat the shift as an investment in higher-value work: freeing data engineers from routine maintenance so they can focus on data quality, semantic modeling, and architectural design.

There is also a skills gap to close. Many data engineers are strong in SQL, Python, and pipeline orchestration tools such as Airflow but have limited experience with metadata management, knowledge graphs, or data-governance automation. As the integration stack evolves, investment in these skills through training, hiring, or partnerships becomes increasingly important. Data engineers in 2025 are likely to spend more time in a metadata catalog and a governance console than in an Airflow DAG editor, and preparation for that shift should begin today.

5. Conclusion

The ETL pipeline is not dead, but its dominance is ending. The convergence of zero ETL integration capabilities from major cloud platforms with data fabric architecture from the metadata and governance side is reshaping how enterprises move—and, more often, do not move—data for analytical consumption. The framework presented in this paper provides a structured way to evaluate these patterns along the

dimensions that matter most: data freshness, governance continuity, operational burden, source system impact, schema evolution tolerance, and cross-platform reach.

The scenario analysis yields clear guidance. CDC-based zero ETL substantially improves operational reporting and real-time dashboards through dramatic gains in freshness and operational simplicity. Data fabric's discovery and virtualization capabilities are the primary value driver for cross-departmental ad-hoc analytics. The combination of zero ETL replication and fabric governance gives regulatory use cases both the data freshness and the auditability required for compliance. Lakehouse architecture consolidates pipeline stages for data science workflows. For external data sharing, open table formats and sharing protocols offer a promising but still-maturing alternative to file-based exchange.

The principal takeaway is that these patterns are complementary, not competitive. An enterprise does not choose between zero ETL and data fabric; it uses both. The metadata intelligence of the fabric determines when to apply zero ETL replication, when to virtualize, and when to fall back on traditional pipelines for complex transformations. While no single product fully realizes this vision today, the fabric-orchestrated zero ETL model we propose indicates the direction the industry is heading.

For data engineers, the implication is a shift in the skills that matter. Less time is spent writing and maintaining extraction scripts; more time is spent designing governance frameworks, building semantic layers, and curating metadata. The engineering effort does not disappear; it moves up the stack from mechanical data plumbing to architectural data design. For researchers, the field offers fertile ground: formal models for integration pattern selection, empirical studies of zero ETL latency and consistency guarantees under production workloads, and frameworks for quantifying the total cost of ownership of fabric-orchestrated architectures compared to traditional pipeline-based approaches.

As of September 2023 the tooling is ready for the transition to begin in earnest, even though the full journey will take years. Legacy pipelines and zero ETL integrations will coexist for a long time, and organizations should plan for that coexistence rather than a clean cutover. A reasonable starting point is to audit current pipelines, classify them by the integration-pattern taxonomy presented here, and identify candidates for replacement with zero ETL or virtualization capabilities already available on the organization's platforms. Beginning with low-risk, high-maintenance pipelines—those that fail most frequently and consume the most engineering time—delivers quick wins that build organizational confidence and create space for the more ambitious architectural work that follows.

Investment today in metadata infrastructure, governance automation, and platform-native integration capabilities is the surest path for enterprises seeking to reduce their pipeline burden steadily over time. The end state is not a world without data engineering—far from it—but one where data engineers devote their time to architectural design, governance strategy, and analytical enablement rather than to the mechanical plumbing that has occupied them for the last three decades. That is the future worth building toward, one platform-native integration at a time.

6. Conflicts of Interest

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

7. Funding Statement

No external funding was received for the research and publication of this article.

8. Acknowledgments

The author thanks the reviewers and practitioners whose feedback shaped earlier drafts of this paper.

References

[1] W. H. Inmon, Building the Data Warehouse. New York: John Wiley & Sons, 1992.

[2] R. Kimball and M. Ross, The Data Warehouse Toolkit. New York: John Wiley & Sons, 1996.

[3] Z. Dehghani, Data Mesh: Delivering Data-Driven Value at Scale. Sebastopol, CA: O'Reilly Media, 2022.

[4] Gartner, "Data Fabric Architecture Is Key to Modernizing Data Management and Integration," Research Note, 2019.

[5] Gartner, "Top Trends in Data and Analytics for 2022," Research Report, 2022.

[6] Amazon Web Services, "Amazon Aurora Zero-ETL Integration with Amazon Redshift," re:Invent 2022 Announcement, 2022.

[7] Amazon Web Services, "Zero-ETL Integration: General Availability for Aurora MySQL," AWS Documentation, 2023.

[8] Snowflake, "Secure Data Sharing," Snowflake Documentation, 2023.

[9] Databricks, "Delta Sharing: An Open Protocol for Secure Data Sharing," Databricks Technical Blog, 2023.

[10] Fivetran, "The State of Data Engineering Report," fivetran.com, 2022.

[11] M. Kleppmann, Designing Data-Intensive Applications. Sebastopol, CA: O'Reilly Media, 2017.

[12] M. Armbrust et al., "Lakehouse: A New Generation of Open Platforms That Unify Data Warehousing and Advanced Analytics," in Proc. CIDR 2021.

[13] Apache Iceberg Project, "Apache Iceberg: An Open Table Format for Huge Analytic Datasets," iceberg.apache.org, 2023.

[14] OpenMetadata, "OpenMetadata: A Single Place to Find, Collaborate, and Get Your Data Right," open-metadata.org, 2023.

[15] N. Narkhede, G. Shapira, and T. Palino, Kafka: The Definitive Guide. Sebastopol, CA: O'Reilly Media, 2017.