The Eastasouth Journal of Information System and Computer Science
Vol. 1, No. 01, August, pp. 288 - 298

A FinTech-Oriented Data Mesh and Lakehouse Architecture for Governed, Real-Time Financial Analytics

Gopinathan Rathinavelu
Data Architect, Arizona, USA

Article Info

Article history:

Received Jun, 2023
Revised Jul, 2023
Accepted Aug, 2023

Keywords:

AML;
Credit Risk;
Data Mesh;
Data Products;
Federated Governance;
Financial Data Architecture;
FinTech;
Fraud Analytics;
Lakehouse;
Regulatory Reporting

ABSTRACT

Financial institutions depend on timely, trustworthy, and traceable data to support fraud detection, anti-money-laundering (AML) monitoring, credit-risk assessment, payment operations, customer analytics, and regulatory reporting. Centralized data platforms can become delivery bottlenecks when business domains use different systems, definitions, access models, and release cycles. This paper proposes a FinTech-oriented enterprise data platform that combines Data Mesh principles with a Lakehouse foundation, and describes it through three complementary views: a platform overview that separates data creators from data consumers, a logical architecture in which domain-owned storage and compute are exposed as one logical lakehouse under governance, operational, and development planes, and an operating model in which domain, platform, and governance teams publish bronze, silver, and gold data layers. Domains such as payments, cards, customer identity, lending, fraud, and compliance publish governed data products through standard contracts. A shared platform provides batch and streaming ingestion, open table storage, virtualization and federated querying, metadata discovery, policy-based access, lineage, quality checks, and automated deployment. Federated computational governance sets common controls while domain teams manage products close to their business context. The paper presents requirements, components, representative financial workflows, governance controls, and an evaluation plan. It reports a reference design, not a measured deployment; expected benefits are therefore stated as hypotheses to be tested in a pilot.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Name: Gopinathan Rathinavelu
Institution: Data Architect, Arizona, USA
Email: gopinathan.r@gmail.com

1. INTRODUCTION

Financial technology systems generate high-volume, time-sensitive data across card transactions, payment rails, customer onboarding, digital channels, lending platforms, risk engines, and compliance systems. Analytics teams need to combine these signals while preserving data quality, privacy, access restrictions, and traceability. Traditional centralized architectures become difficult to scale when every new use case depends on a central team to ingest, model, approve, and publish data. At the other extreme, unconstrained domain-level platforms create duplicated datasets, inconsistent definitions, and fragmented security controls.

Data Mesh [1], [2] offers an organizational and architectural approach built on domain ownership, data as a product, self-service infrastructure, and federated computational governance. A Lakehouse [3], which builds on earlier data lake platforms [4], complements it by providing a shared foundation for structured, semi-structured, and unstructured data, with table management and analytical query capabilities. Combining the two is relevant to FinTech because data products must be independently useful while remaining governed across organizational and institutional boundaries.

Scope and review period. This paper is a retrospective review. It deliberately draws only on sources published before 1 January 2022, covering the period in which Data Mesh was first described (2019 to 2020) [1], [2] and the Lakehouse pattern was formalized (2020 to 2021) [3], [5]. Later work, including the book-length treatment of Data Mesh, the final text of the EU Digital Operational Resilience Act, the EU Artificial Intelligence Act, and version 2.0 of the NIST Cybersecurity Framework, falls outside this window; Section 8 notes how these developments affect the design without reviewing them.

This paper adapts an enterprise Data Mesh and Lakehouse design to financial-services workflows. It presents a reference architecture rather than a report of a completed production implementation. The contributions are:

a. a three-view description of the architecture (Figures 1 to 3) that links the platform overview, the logical structure, and the Data Mesh operating model;
b. a mapping of domain-oriented data products and shared governance controls to representative financial use cases (payments, fraud, AML, credit risk, and regulatory reporting); and
c. a practical framework for validating security, data quality, timeliness, portability across cloud providers, and operational performance in a pilot.

Section 2 states the problem and objectives. Section 3 presents the architecture. Section 4 describes governance and financial controls, Section 5 walks through representative workflows, Section 6 outlines an implementation method, and Section 7 defines the evaluation framework. Section 8 discusses limitations, developments after the review period, and future work, and Section 9 concludes.

2. PROBLEM STATEMENT AND DESIGN OBJECTIVES

A FinTech data platform must reconcile several competing requirements. Transaction and event data arrive continuously, while financial reporting and model development often require reproducible historical snapshots. Customer, account, merchant, payment, and risk identifiers may differ across source systems. Sensitive information requires purpose-based access and minimization. Regulatory and internal reviews may require lineage from a report or model feature back to source records and transformation logic. Meanwhile, fraud and payment-risk use cases can be sensitive to processing delays.

The design addresses five problems: (1) fragmented data ownership and duplicated integration work; (2) inconsistent definitions of financial entities and metrics; (3) delays between event generation and analytical availability; (4) limited traceability of data transformations and access decisions; and (5) the risk that decentralized delivery weakens institution-wide controls. The principal objectives are:

a. Domain ownership: let accountable teams create and maintain data products for their subject areas.
b. Interoperability: define common schemas, identifiers, semantic definitions, and product contracts.
c. Secure sharing: enforce least-privilege access, classification, purpose restrictions, and auditable approvals.

d. Timeliness: support streaming and batch workloads according to business service objectives.
e. Trust: apply quality checks, lineage capture, versioning, and operational monitoring.
f. Self-service: let domain teams build and deploy products on shared infrastructure without depending on a central delivery queue.
g. Portability: avoid making the conceptual architecture dependent on a single cloud or vendor, and aim for functional parity across the cloud providers in use.

3. FINTECH DATA MESH AND LAKEHOUSE ARCHITECTURE

The architecture is presented first through three views and then in detail for each logical layer. Figure 1 is the platform overview, Figure 2 is the logical architecture, and Figure 3 is the Data Mesh operating model. The five logical layers are source systems and ingestion; Lakehouse storage and processing; domain data products; consumers and serving; and federated governance with shared platform services. These are logical boundaries and may be implemented with different products, depending on institutional standards.

3.1 Platform Overview: Data Creators and Data Consumers

Figure 1 shows the platform as a unified data platform built around a Lakehouse, with data creators on one side and data consumers on the other. Creators bring data in and publish products. The creator-side capabilities are source ingestion, data pipelines and virtualization, data storage, DataOps with CI/CD, a catalogue product, and policy-based access. Consumers discover and use the published products through a data marketplace, BI and reporting, data APIs, and data science tooling.

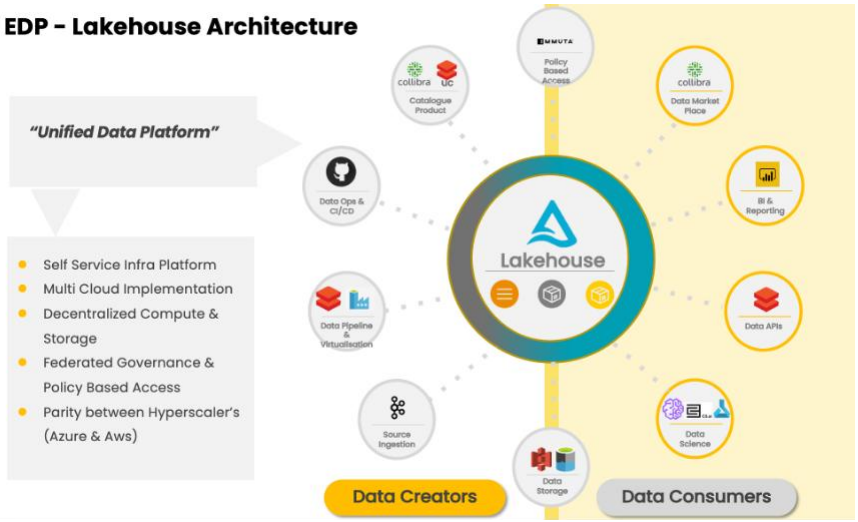


Figure 1. Overview of the Lakehouse-based enterprise data platform (EDP)
Data creators and data consumers are connected through a unified Lakehouse; the callout lists the platform's five design characteristics.

The callout in Figure 1 describes five characteristics of the platform: a self-service infrastructure platform; a multi-cloud implementation; decentralized compute and storage; federated governance with policy-based access; and parity between the two hyperscalers used (Azure and AWS). In this paper they are treated as design goals, not demonstrated properties. Multi-cloud parity in particular needs explicit testing (Section 7). Table 1 summarizes how the capabilities in the figures map to the logical roles discussed in the rest of the paper, together with the example tooling that the diagrams label. The tools are

examples of how each role can be filled; the conceptual design does not depend on them.

Table 1. Platform capabilities and the example tooling labeled in Figures 1 and 3

Capability	Role in the architecture	Example tooling shown
Source ingestion	Batch extracts, change data capture, and event streams delivered into domain pipelines	Event-streaming connector (Fig. 1); pub/sub per domain (Fig. 2)
Pipelines and virtualization	Transform data and expose it across domain environments without central copying	Databricks SQL Analytics (Fig. 3)
Open table storage	Transactional, versioned tables for bronze, silver, and gold layers	Delta format; Databricks notebooks (Fig. 3)
Catalogue and metadata	Enterprise and technical catalogue; discovery; lineage metadata	Collibra; Databricks Unity Catalog (Figs. 1, 3)
Policy-based access	Attribute- and purpose-aware enforcement at query time	Immuta (Fig. 1)
DataOps and CI/CD	Common build, test, and deployment flow for pipelines and policies	GitHub; GitHub Actions (Figs. 1, 3)
Consumption	Marketplace, BI and reporting, data APIs, data science	Collibra data marketplace (Fig. 1)

3.2 *Logical Architecture and Platform Planes*

Figure 2 shows the logical structure. Storage and compute are decentralized: each line of business (LOB) or domain runs its own storage and compute environment (LOB Storage and Compute-1 to -N). Each environment has a storage layer, a data-virtualization layer, its own data pipelines, and a publish-subscribe channel for event management. Domain data products are exposed above these environments through one logical lakehouse. Consumers therefore deal with a single logical platform, while the data stays with the domain that owns it. Three consumer-facing channels sit on top of the data products: visualization and reporting tools, data APIs, and a data marketplace.

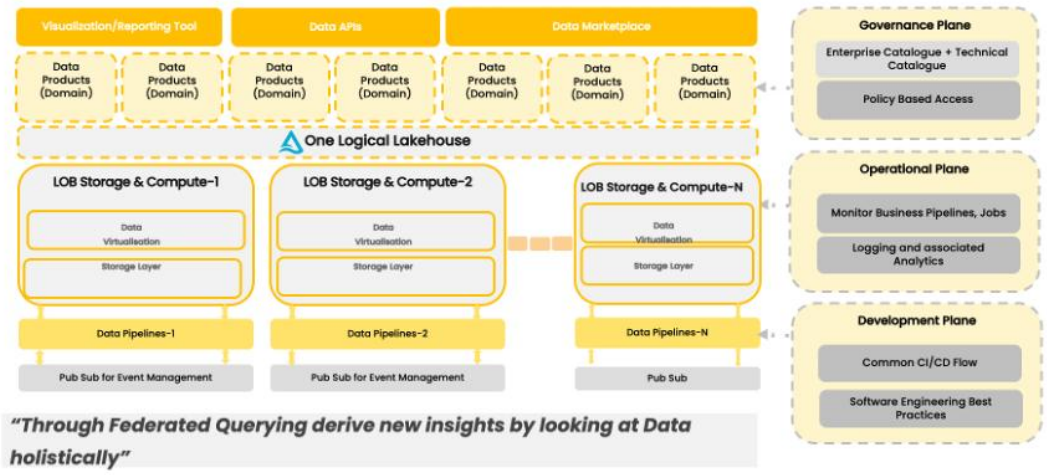


Figure 2. Logical architecture
Domain-owned storage and compute environments are exposed as one logical lakehouse beneath domain data products and consumer channels, and are supported by governance, operational, and development planes.

Federated querying across the logical lakehouse lets analysts combine data from several domains without first copying it into a central store; the figure expresses this as deriving new insights by looking at data holistically. In financial settings this must be bounded by policy. A federated query that joins payment and customer products should run only if the permitted purposes of both products allow it (Section 4.1).

Three planes support the domain environments. Table 2 lists the components that Figure 2 places in each plane and the financial-services controls they support.

Table 2. Platform planes in the logical architecture

Plane	Components shown in Figure 2	Financial-services controls supported
Governance	Enterprise catalogue and technical catalogue; policy-based access	Classification, product ownership, purpose restrictions, access approvals, lineage metadata
Operational	Monitoring of business pipelines and jobs; logging and associated analytics	Freshness and failure monitoring, access and operational logs, incident evidence
Development	Common CI/CD flow; software engineering best practices	Tested and versioned pipeline releases, schema-compatibility checks, controlled rollback

3.3 *Data Mesh Operating Model and Data Layers*

Figure 3 shows how the Data Mesh operating model maps onto the platform. External and internal sources (databases, files, and APIs) feed domain teams. Domain teams create and publish data products on the data platform. A platform team supports them by operating the shared infrastructure, and a governance team controls the platform through centralized policy with federated management backed by extensive automation. Consumers use the published products and give feedback to the domain teams, which closes the loop.

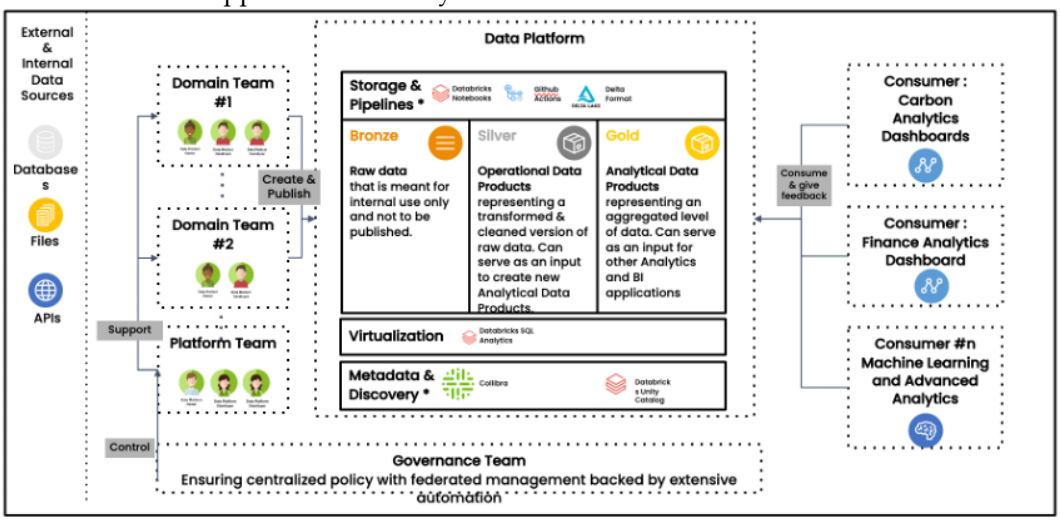


Figure 3. Data Mesh operating model: domain teams, platform team, and governance team around a bronze, silver, and gold data platform with virtualization and metadata services

The consumers drawn in Figure 3 (carbon analytics dashboards, a finance analytics dashboard, and machine learning and advanced analytics) come from the enterprise instance of the design. FinTech consumers such as fraud, AML, credit-risk, and regulatory-reporting workloads (Section 5) take the

same position as the figure's generic consumer.

Data in the platform moves through three layers, summarized in Table 3. Only silver and gold products are published to consumers. The bronze layer holds raw data for internal use and is not published.

Table 3. Data layers in the operating model, with illustrative financial examples

Layer	Content (as shown in Figure 3)	Published?	Illustrative financial example
Bronze	Raw data meant for internal use only	No	Raw authorization messages and source extracts, retained subject to privacy and retention rules
Silver	Operational data products: a transformed and cleaned version of raw data; can feed new analytical data products	Yes	Deduplicated, schema-validated payment events with standardized identifiers
Gold	Analytical data products: aggregated data; can feed other analytics and BI applications	Yes	Fraud feature aggregates, AML alert-enrichment views, reconciled finance measures

3.4 Source Systems and Ingestion

Source systems include card authorization and settlement, payment processing, customer and account platforms, digital banking, merchant services, lending, identity verification, case management, and external reference or sanctions data. Batch connectors capture periodic extracts and change data capture where available. Event streaming [6] transports transaction and status events when low-latency consumption is required; in Figure 2 each domain environment has its own publish-subscribe channel for this purpose. Every event should carry or be associated with event time, ingestion time, source identity, schema version, and a correlation identifier where appropriate. Schema validation and quarantine paths keep malformed messages from silently entering trusted products.

3.5 Lakehouse Storage and Processing

Each domain environment stores data in open, transactional table formats [5] organized in the bronze, silver, and gold layers of Table 3. Bronze preserves source context, subject to retention and privacy requirements. Silver applies schema checks, deduplication, normalization, and basic integrity rules, and publishes operational data products. Gold applies domain semantics and aggregation, and publishes versioned analytical products. ACID-capable table formats and schema-evolution controls support consistent reads and controlled change, subject to the implementation of the chosen platform. The virtualization layer in each environment, together with the logical lakehouse, exposes these tables to consumers and to other domains without moving the underlying data.

Batch processing suits reconciliations, historical aggregation, model training, and periodic regulatory calculations. Streaming or incremental processing suits transaction monitoring, payment status, and selected operational alerts. The architecture does not assume that every workload must run in real time. The processing mode should follow the business latency and correctness requirements.

3.6 Domain Data Products

Domain teams own the meaning, quality, documentation, lifecycle, and service expectations of their products. Illustrative products include:

a. Payments: payment events, settlement status, reversals, and reconciliation exceptions.
b. Cards: authorization events, card-account relationships, merchant attributes, and dispute outcomes.

c. Customer: governed customer and account reference data, identity-resolution results, and approved customer dimensions.
d. Fraud: labeled fraud outcomes, investigation signals, confirmed typologies, and model-feedback datasets.
e. Lending and credit risk: application attributes, repayment events, exposure snapshots, and risk features approved for specified purposes.
f. Compliance: alert dispositions, case status, sanctions-screening outcomes, and regulatory reporting data products.

Each product should publish an owner, description, schema, semantic definitions, classification, permitted purposes, freshness expectation, quality rules, lineage, version, support contact, and deprecation policy. These product contracts help consumers understand both the data and its limits.

3.7 Consumer and Serving Layer

Consumers include BI and reporting tools, fraud and AML analytics, risk-model pipelines, customer operations, finance, and approved APIs, reached through the channels at the top of Figure 2 (reporting tools, data APIs, and a data marketplace). Access should be mediated through governed query, feature-serving, API, or data-sharing interfaces. Consumer-specific views can mask or omit fields that are not needed for a purpose. Models and dashboards should record the product versions and time windows they used, so that results can be reproduced and investigated. Consumer feedback, shown in Figure 3, should be routed to the owning domain team as a tracked request or defect.

3.8 Shared Platform Services

The platform team supplies reusable ingestion templates, orchestration, catalog and discovery, identity integration, secrets management, policy enforcement, observability, lineage collection, quality monitoring, and deployment automation. These services are organized in the governance, operational, and development planes of Figure 2. Central platform teams provide capabilities. Domain teams remain accountable for product semantics and operational readiness.

4. FEDERATED GOVERNANCE AND FINANCIAL CONTROLS

Federated governance combines institution-wide minimum controls with domain-level accountability. In Figure 3 the governance team controls the platform through centralized policy with federated management, backed by extensive automation. A central governance function defines policies, common classifications, approved identity patterns, interoperability standards, retention expectations, and evidence requirements. Domain owners implement these controls in product pipelines and access interfaces. Policy should be enforced through automation wherever feasible, not through documentation alone. Standards such as the NIST Cybersecurity Framework [7] and the BCBS 239 principles for risk data aggregation and risk reporting [8] can inform the control catalogue.

4.1 Identity, Authorization, and Data Minimization

Access follows the principle of least privilege, using role- or attribute-based rules supported by the institution's identity platform. The policy-based access component in Figures 1 and 2 is the enforcement point. Sensitive fields should be classified and protected through masking, tokenization, pseudonymization, row or column restrictions, or controlled secure environments, as appropriate. Authorization should consider the consumer identity, data classification, declared purpose, and approved workload. This applies equally to federated queries that span domains. Credentials and secrets should not be

embedded in notebooks, pipelines, or source code.

4.2 *Data Quality and Reconciliation*

Financial data products should define rules for completeness, uniqueness, valid state transitions, referential integrity, currency and amount formats, event ordering, and reconciliation totals. Quality checks should produce observable outcomes: accepted records, quarantined records, rule violations, and remediation ownership. Late-arriving or corrected events should be handled through explicit replay, correction, and restatement policies.

4.3 *Lineage and Auditability*

Lineage should connect source systems, transformations, product versions, model features, dashboards, and reports. The enterprise and technical catalogues in the governance plane are the natural home for this metadata. Access logs and operational events, collected in the operational plane, should be retained under approved schedules and protected against unauthorized modification. For investigations, the platform should make it possible to identify which data version and transformation logic contributed to a result. The design proposes these controls. The specific audit and retention configuration must be aligned with the institution's legal, regulatory, and internal requirements, including data-protection law such as the GDPR [9] and the operational-resilience requirements proposed for the EU financial sector [10].

4.4 *Model and Regulatory Use*

A data product suitable for exploratory analytics is not automatically approved for a regulated decision. Products used in credit, fraud, AML, or other material decisions should include documented provenance, quality status, permitted-use statements, and relevant limitations. Model governance, explainability, validation, human review, and reporting obligations, including those anticipated in the European Commission's 2021 proposal for AI regulation [11], are separate control areas. The data platform can support them but cannot replace them.

5. REPRESENTATIVE FINTECH WORKFLOWS

5.1 *Fraud and Payment-Risk Analytics*

A payment event enters through the payments domain's ingestion channel with source and event-time metadata. It is stored in bronze, validated against a versioned schema, and promoted to a silver payment-event product. Approved merchant, account, and historical outcome products can be joined, through the logical Lakehouse, to produce gold features or analytical views. A fraud system consumes the resulting view through an authorized interface. Confirmed outcomes flow back to the fraud domain as labeled data, subject to quality review and appropriate controls. The architecture can support streaming features and retrospective analysis, but achievable latency and detection quality must be measured in the target environment.

5.2 *AML Monitoring and Case Management*

Transaction, customer, account, and reference-data products can be combined to support monitoring rules, alert enrichment, and case investigation. Case outcomes can be published as a governed product for typology analysis and controlled model feedback. Access to personal and investigative information should be limited by role and purpose, with auditable access, consistent with international AML standards [12]. The platform does not determine whether a transaction is suspicious. Detection logic, investigator decisions, and regulatory reporting remain the responsibility of the institution's compliance processes.

5.3 *Credit-Risk and Lending Analytics*

Application, exposure, repayment, delinquency, and customer-reference products can support portfolio analysis and approved model-development datasets. Product

versioning and point-in-time availability are important to reduce temporal leakage and to enable reproducible backtesting. Features should document their calculation windows, source dependencies, missing-value behavior, and intended use. Any credit decision must remain subject to the institution's model-risk, fairness, consumer-protection, and approval processes.

5.4 Regulatory and Finance Reporting

Curated, reconciled gold products can provide reusable inputs to finance, risk, and regulatory reporting. Reports should retain mappings from reported measures to definitions, transformations, source periods, and product versions. Reconciliation controls should identify differences between operational systems and analytical outputs. A governed Data Mesh can improve traceability and support risk data aggregation practices [8], but it does not itself certify that a report complies with any particular regulation.

6. IMPLEMENTATION METHODOLOGY

Implementation should proceed incrementally, beginning with a bounded use case and a small number of domains. A suggested sequence is:

a. Discover and classify: inventory sources, consumers, data owners, sensitivity classes, retention constraints, and existing controls.
b. Select a pilot: choose a use case with clear consumers and measurable service objectives, such as payment reconciliation or fraud analytics. Confirm access approvals and baseline measures before implementation.
c. Define product contracts: document schemas, semantic definitions, owners, quality rules, freshness targets, permitted purposes, and versioning expectations.
d. Establish the platform path: implement ingestion, storage, orchestration, catalog registration, identity integration, lineage, monitoring, and controlled deployment templates, covering the governance, operational, and development planes of Figure 2.
e. Publish and consume: expose the product through a governed interface, and validate consumer usability, access enforcement, and downstream compatibility.
f. Evaluate and improve: compare outcomes against the baseline, record exceptions, review access and quality evidence, and prioritize improvements before adding domains.

The implementation should use automated testing for schema compatibility, data-quality rules, policy enforcement, replay behavior, and lineage completeness. Rollback and deprecation procedures should be tested before a product is treated as production-critical.

7. EVALUATION FRAMEWORK

This paper presents a reference design. It does not report a completed FinTech deployment, benchmark dataset, or measured results, and it therefore claims no numerical performance gains. The following framework is proposed for a pilot.

a. Data freshness: measure event-to-product and product-to-consumer delay using defined percentiles and business-hour windows.
b. Throughput and scalability: measure sustained records per second, peak handling, backlog recovery, and cost under representative loads.
c. Data quality: track completeness, duplicate rate, schema violations, reconciliation differences, and time to resolve incidents.
d. Governance effectiveness: test whether unauthorized requests are denied, approved requests are scoped correctly, and access events are auditable, including for federated queries across domains.

e. Federated query behavior: measure latency and result correctness for queries that span domain environments, compared with the same query on a centralized copy.
f. Lineage coverage: measure the proportion of critical fields and outputs traceable to source records and transformation versions.
g. Reliability: record availability, pipeline failures, recovery time, replay correctness, and data-loss incidents.
h. Multi-cloud parity: run the same pilot workload on each cloud provider in use and compare functionality, performance, cost, and operational effort.
i. Consumer productivity: assess time to discover, obtain access to, understand, and safely reuse a data product.
j. Financial-use-case outcomes: for fraud or AML, evaluate relevant business metrics such as investigation workload, alert quality, and case-processing time using a controlled and approved study design.

Comparisons should use a documented baseline, representative data volumes, fixed measurement windows, and consistent definitions. Results should include workload characteristics and known limitations. No metric should be presented as an observed result until the pilot has been executed and its evidence reviewed.

8. DISCUSSION, LIMITATIONS, AND FUTURE WORK

The combination of Data Mesh and Lakehouse architecture aims to balance domain autonomy with institution-wide controls. Its potential advantages include clearer ownership, reusable data products, less repeated integration, better discovery, and stronger traceability. These benefits depend on organizational readiness and implementation quality. Adopting the terminology alone does not deliver them.

Several limitations remain. First, federated ownership requires product-management skills, operational accountability, and clear escalation paths. Second, common identifiers and semantic definitions can be difficult to establish across legacy platforms. Third, streaming pipelines add complexity around event ordering, duplicates, late data, replay, and reconciliation [13]. Fourth, federated querying across domain-owned environments can add latency and complicate policy enforcement, and sensitive financial data may restrict cross-domain sharing even when the platform supports it. Fifth, the diagrams name specific commercial products. Cloud- and vendor-specific services can affect portability, cost, and operational complexity, and the stated goal of parity between Azure and AWS is untested. Finally, the design has not been validated through reported production measurements.

Developments after the review period. Several sources cited here have since been superseded or finalized. The proposed operational-resilience regulation [10] was adopted as Regulation (EU) 2022/2554 (DORA), the AI proposal [11] became Regulation (EU) 2024/1689, NIST published version 2.0 of the Cybersecurity Framework in 2024, and Data Mesh received a book-length treatment in 2022. These changes do not alter the architecture's structure, but anyone applying it today should map the control catalogue in Section 4 to the current versions rather than to the versions reviewed here.

Future work should implement a pilot with synthetic or appropriately authorized de-identified data, publish reproducible workload and test specifications, and evaluate the metrics defined in Section 7. Further work could examine data-product service-level objectives, automated policy testing, point-in-time feature reproducibility, lineage completeness, federated-query policy enforcement, and cross-cloud data-sharing patterns. A production study should also document the governance approvals and security boundaries used during evaluation.

9. CONCLUSION

This paper presents a FinTech-oriented reference architecture that combines Data Mesh principles with a governed Lakehouse platform, described through a platform overview, a logical architecture, and an operating model. It assigns product accountability to financial domains and uses shared infrastructure for ingestion, storage, virtualization, discovery, policy enforcement, quality monitoring, and lineage. Representative workflows show how payment, fraud, AML, credit-risk, and regulatory-reporting consumers could use governed data products without removing domain accountability or institutional control. The architecture is a design proposal, not a report of measured deployment outcomes. Its effectiveness should be established through a bounded pilot with documented baselines and explicit measures for freshness, quality, security, traceability, reliability, cost, multi-cloud parity, and consumer productivity.

REFERENCES

[1] Z. Dehghani, "How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh," Martin Fowler. [Online]. Available: https://martinfowler.com/articles/data-monolith-to-mesh.html

[2] Z. Dehghani, "Data Mesh Principles and Logical Architecture," Martin Fowler. [Online]. Available: https://martinfowler.com/articles/data-mesh-principles.html

[3] M. Armbrust, A. Ghodsi, R. Xin, and M. Zaharia, "Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics," in *proceedings of 11th Annual Conference on Innovative Data Systems Research (CIDR '21)*, 2021. [Online]. Available: https://vldb.org/cidrdb/2021/lakehouse-a-new-generation-of-open-platforms-that-unify-data-warehousing-and-advanced-analytics.html?utm_source=chatgpt.com

[4] F. Nargesian, E. Zhu, R. J. Miller, K. Q. Pu, and P. C. Arocena, "Data lake management: challenges and opportunities," *Proc. VLDB Endow.*, vol. 12, no. 12, pp. 1986–1989, 2019, doi: 10.14778/3352063.3352116.

[5] M. Armbrust *et al.*, "Delta lake: high-performance ACID table storage over cloud object stores," *Proc. VLDB Endow.*, vol. 13, no. 12, pp. 3411–3424, 2020, doi: https://doi.org/10.14778/3415478.3415560.

[6] J. Kreps, N. Narkhede, and J. Rao, "Kafka: A distributed messaging system for log processing," in *Proceedings of the 6th International Workshop on Networking Meets Databases (NetDB '11)*, Athens, Greece, 2011.

[7] M. Barrett, "Framework for Improving Critical Infrastructure Cybersecurity Version 1.1," National Institute of Standards and Technology, 2018. doi: https://doi.org/10.6028/NIST.CSWP.04162018.

[8] Basel Committee on Banking Supervision, "Principles for effective risk data aggregation and risk reporting," BCBS 239, Bank for International Settlements, 2013.

[9] European Parliament and Council of the European Union, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 (General Data Protection Regulation)," in *Official Journal of the European Union*, L 119, 2016, pp. 1–88.

[10] European Commission, *Proposal for a regulation of the European Parliament and of the Council on digital operational resilience for the financial sector and amending Regulations (EC) No 1060/2009, (EU) No 648/2012, (EU) No 600/2014 and (EU) No 909/2014*. COM (2020) 595 final, 2020.

[11] European Commission, *Proposal for a regulation of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)*. COM (2021) 206 final, 2021.

[12] Financial Action Task Force, "International standards on combating money laundering and the financing of terrorism & proliferation: the FATF recommendations," FATF, 2020.

[13] M. Kleppmann, *Designing data-intensive applications*. O'Reilly Media, 2017.