International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Unified Data Lakehouse Architecture for Real-Time and Batch Data Integration in Multi-Cloud Environments

Sahini Dyapa

Abstract: Enterprise data ecosystems increasingly span transactional platforms, event streaming infrastructure, cloud object storage, and machine learning environments — often without a unifying architectural layer to govern how data flows between them. Fragmentation across these systems produces inconsistencies in data quality, lineage, and access control that erode analytical trust and complicate regulatory compliance. This article presents a unified data lakehouse architecture designed to integrate real-time and batch data processing within a governed, scalable, and cloud-agnostic analytical foundation. The framework combines open table format storage, metadata-driven governance, zone-based data organization, and a unified access abstraction to support diverse workload types across multi-cloud environments. Streaming pipeline patterns are focused on low-latency manipulation of events. Batch patterns are used more commonly for reconciliation, historical analysis and regulatory reporting. Data quality checks can be enforced at ingestion, transformation, and consumption, which shows the architecture to be valuable in regulated, high-throughput application spaces, such as healthcare and financial services. The proposed framework may reduce architectural fragmentation, improve data trustworthiness, and establish a consistent governance model that scales with enterprise data complexity.

Keywords: *Batch Processing, Data Governance, Data Lakehouse, Data Quality, Multi-Cloud Architecture, Real-Time Streaming*

1. Introduction

Few challenges in enterprise data management have proven as persistent as architectural fragmentation. As organizations in healthcare, financial services, retail, and manufacturing have expanded their data collection footprints — incorporating transactional records, patient monitoring streams, payment event feeds, and customer behavioral data — the platforms built to handle these workloads have multiplied rather than converged [1]. The downstream consequence is a data environment where the same business entity may reside in multiple systems with different freshness levels, different definitional assumptions, and different governance standards applied to each copy.

Traditional responses to this problem — adding specialized platforms for structured reporting, raw storage, streaming processing, and machine learning — address individual workload requirements but intensify the coordination burden [2]. Data moves repeatedly between systems. Transformation logic is duplicated. Metadata is maintained independently in each platform. Quality controls are applied inconsistently, and access policies diverge across environments. For organizations in regulated industries, this inconsistency carries direct compliance risk: when auditors or regulators require an explanation of how a reported figure was derived, a fragmented architecture may not be able to provide one [3].

The data lakehouse architecture has emerged as a structural alternative that addresses these limitations. Combining the low-cost, flexible storage characteristics of a data lake with the reliability, governance, and query performance capabilities of a data warehouse, the lakehouse enables diverse workload types — batch transformation, real-time ingestion, interactive analytics, and model training — to operate over a shared, governed storage foundation [1][4]. Open table formats provide transactional consistency and schema evolution over cloud object storage. Separation of storage from compute allows workload-specific engines to access the same datasets without requiring duplication [2].

Multi-cloud deployment adds a further layer of architectural consideration. Organizations operating across multiple cloud providers face the risk of recreating platform fragmentation at the cloud level — isolated data environments per provider, inconsistent governance policies, and expensive

TEKsystems, Inc., USA

cross-cloud data movement [5]. A lakehouse architecture designed for multi-cloud deployment uses cloud-neutral storage standards, federated metadata governance, and shared access policies to maintain architectural coherence across provider boundaries [6][7].

The article outlines an architecture for unified data lakehouses, covers integration of real-time and batch data pipelines, outlines metadata governance and data quality, performance, security, and multi-cloud risk management, along with industry use cases in healthcare and financial services where operational complexities, high regulatory requirements, and need for improved efficiency are a challenge.

2. Core Lakehouse Architectural Framework

2.1 Data Lakehouse Concept and Design Principles

Understanding the lakehouse requires moving past the product framing that often surrounds it. At its core, the lakehouse is an architectural pattern that defines how storage, compute, metadata, and governance responsibilities are organized to support analytical workloads of different types over a common data foundation [1][7]. It is neither a data lake with added structure nor a data warehouse with cheaper storage — it is a distinct design that attempts to provide the advantages of both while avoiding their respective limitations.

Storage in the lakehouse relies on cloud object stores, which provide cost-effective, durable, and scalable storage accessible by compute engines deployed on any major platform [8]. Open columnar file formats organize data for efficient analytical reads, allowing query engines to retrieve only the columns required for a given query rather than scanning full records. Open table formats layer transactional semantics — including ACID consistency, schema evolution, partition management, and time-travel access to historical versions — over the columnar file base [2]. These capabilities matter particularly in enterprise environments where data definitions change over time: the ability to evolve schemas without reprocessing historical data, and to access prior table versions for audit or reprocessing purposes, directly addresses operational requirements that traditional data lakes cannot satisfy.

The separation of storage from compute allows different processing engines to address different workload requirements without competing for shared infrastructure resources [8]. Batch transformation engines may process multi-terabyte historical datasets during scheduled windows. Streaming engines may continuously update near-real-time summary tables from high-frequency event sources. Interactive query engines may serve concurrent analyst requests with low-latency response. Machine learning environments may read large feature datasets for model training. Because each engine accesses the same governed storage layer, their outputs remain mutually consistent, and the analytical foundation stays unified rather than fragmented across compute-specific silos [16].

The zone-based organizational model structures the internal hierarchy of the lakehouse. Raw zones preserve source data with minimal modification, providing the auditability and reprocessing capability that regulated industries require. Standardized zones apply format normalization and schema alignment. Curated zones apply business logic and quality certification. Consumption zones present approved data products to analytical consumers [9][6]. Each zone carries distinct ownership standards, quality controls, and access permissions appropriate to the nature of the data it holds.

Table 1. Comparison of Data Lake, Data Warehouse, and Data Lakehouse Architectural Attributes

Attribute	Data Lake	Data Warehouse	Data Lakehouse
Storage model	Raw files in object storage; no schema enforcement	Structured tables; rigid schema-on-write	Open table formats over object storage; schema-on-read with ACID guarantees
Governance	Minimal; catalog optional; inconsistent access control	Strong; built-in access control; auditable	Metadata-driven; unified catalog; consistent policy enforcement across engines
Query performance	Variable; no indexing; slow for ad-hoc analytics	High; optimized for structured SQL queries	Optimized via partitioning, clustering, and caching; supports multiple query engines
Schema handling	Schema-on-read; no evolution tracking	Schema-on-write; evolution requires DDL changes	Schema evolution supported; historical versions preserved via time-travel
Workload support	Batch and ML; poor real-time support	Batch reporting and BI; limited ML support	Batch, streaming, interactive, and ML over shared storage
Cost characteristics	Low storage cost; high processing cost for unstructured data	High storage and compute cost; fixed capacity	Low storage cost; elastic compute; tiered cost by workload priority

2.2 Multi-Cloud Integration Strategy

Multi-cloud data strategy introduces a category of challenge distinct from the technical complexities of any single platform. Each cloud provider maintains its own storage APIs, identity frameworks, cost models, and networking boundaries. Where data platforms are designed independently per provider, the organization may find itself managing the same class of fragmentation problem at a higher level — not across analytical workload types, but across cloud boundaries [5].

The mitigation strategy centers on enforcing cloud neutrality at the storage and metadata layers. Open table formats and open file formats are readable by processing engines deployed on any major cloud platform. Data stored in these formats does not require reformatting or re-ingestion when processing logic is executed in a different cloud environment, which preserves portability without requiring physical data consolidation [1][2]. This design choice also reduces vendor lock-in, giving organizations the flexibility to shift processing workloads between providers in response to cost, capability, or regulatory changes.

Cross-cloud data movement should be minimized rather than normalized. Large-scale data transfers between cloud providers generate latency, cost, and compliance exposure. A multi-cloud lakehouse that organizes data by regulatory region, source proximity, or workload affinity — while providing enterprise-wide discoverability through a federated metadata layer — may achieve the analytical objectives of centralization without incurring its data movement costs [5][7]. The federated catalog presents a unified view of data assets across all environments, allowing users to locate and access datasets through a consistent interface regardless of physical location [10].

Consistent access policy enforcement across cloud environments is a security and compliance requirement that demands deliberate architectural attention. Centralized identity management, role-based access policy, and audit logging of access events ensure that sensitive datasets are governed equivalently across providers [11]. Otherwise, governance asymmetries may expose sensitive data to more risk on one provider versus another in a multi-cloud environment, as data is handled differently across providers.

3. Real-Time and Batch Pipeline Integration

3.1 Streaming Data Pipelines

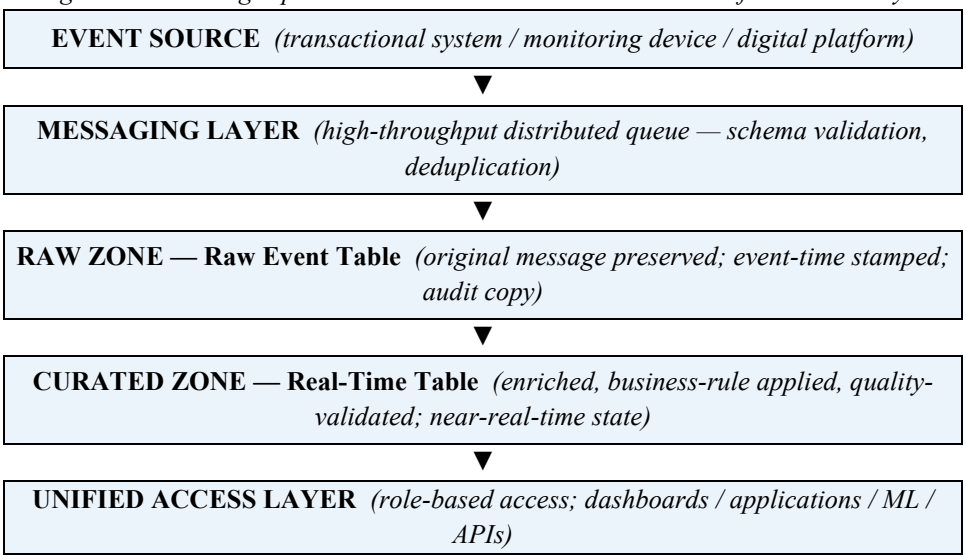
Event-driven analytics systems need to be able to receive, validate, improve and route events with very low latency, relative to the time of the underlying business event. For example in healthcare, alerts from patient monitoring, clinical events and admission records may need to be processed and made available to downstream systems within seconds or minutes [12][13]. In

financial services, payment authorization events, fraud scoring inputs and real-time balance updates have similar low latency requirements [14]. Batch processing windows spanning hours are insufficient for these use cases.

A streaming pipeline begins with a distributed event source — a transactional system, a monitoring device, or a digital interaction platform — that publishes messages to a high-throughput messaging layer. The pipeline applies validation, transformation, and routing logic to each event before writing it to governed storage tables [15]. Lakehouse raw event tables, stored in the raw zone with the raw message contents, are backed by minimal transformations, while curated zone transformations generate new real-time incrementally computed tables to represent the current business state in a query-ready manner [9]. Thus, the original event is replayable and auditable, and can also be used to create analytics outputs for dashboards and operational systems.

Many architectures used for streaming share similar characteristics, and go beyond simple event forwarding. Event-time semantics must be applied to distinguish the business timestamp embedded in the event from the processing timestamp assigned when the system receives it. This distinction is critical for aggregations and windows that must reflect business reality rather than system processing characteristics [16]. Deduplication logic based on event keys and checksums prevents duplicate delivery from appearing as duplicate business events in downstream datasets. Checkpointing mechanisms enable recovery from processing failures without re-ingesting the full event history. Each streaming dataset should carry a schema definition, an owner, quality rules, and a retention policy so that it may be governed with the same rigor applied to batch data assets [4][15].

Figure 1. Streaming Pipeline Architecture — Event Source to Unified Access Layer



3.2 Batch Processing Systems

Batch processing supports the full range of enterprise analytical workloads that require large-scale historical data access, complex multi-source joins, or deterministic reconciliation to authoritative source values. Financial reconciliation and regulatory position reporting, population health measurement, customer lifetime value calculation, and supply chain cost aggregation all depend on batch pipelines that process complete dataset snapshots or well-defined incremental windows [17][14]. These workloads prioritize correctness and completeness over latency.

Within the lakehouse, batch pipelines consume data from raw and standardized zones and apply transformation logic to produce curated data products. Modular pipeline design — where each transformation step has a defined scope, documented business rule assumptions, and associated quality checks — supports ongoing maintainability and reduces the risk that undocumented logic will produce incorrect outputs when source data characteristics change [17]. Incremental processing patterns apply transformations only to new or modified partitions, reducing compute and storage cost relative to full-

table reprocessing while producing equivalent results for append-oriented datasets [2].

Historical correction handling is a capability that batch pipeline architectures in regulated industries must address explicitly. Source systems in financial services and healthcare routinely issue amended records, retroactive adjustments, and corrected file deliveries. The lakehouse supports correction workloads through partition-level reprocessing and time-travel versioning, allowing affected data segments to be updated without destroying prior historical table states [2][4]. This preserves the ability to reproduce any previously reported figure from a consistent historical snapshot — a requirement in regulatory examination and internal audit contexts.

The complementary nature of streaming and batch pipelines is worth stating directly. Streaming pipelines provide freshness and operational responsiveness; batch pipelines provide completeness, historical accuracy, and reconciled final values [7]. A payment transaction may first appear in a real-time dashboard through a streaming pipeline, then be confirmed and reconciled in the final reported position through a batch pipeline. Both representations are valid within their defined scope when freshness levels and intended use are clearly documented.

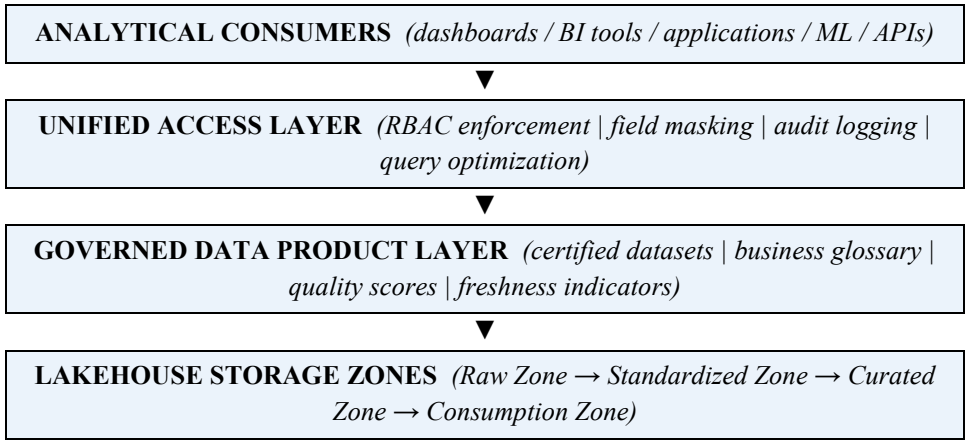
3.3 Unified Data Access Layer

Consumer-facing access to lakehouse data should be mediated through an abstraction layer that decouples the technical structure of the storage architecture from the interfaces through which data is queried and consumed. Analysts and application developers should not need to understand zone hierarchies, partition layouts, or pipeline execution schedules to access governed data products [6][7]. The access layer provides a consistent interface — structured query, API, or semantic model — through which data consumers interact with certified datasets.

Access control is enforced at this layer through role-based permission policies that define what datasets each user class or application may access, what operations they may perform, and what field-level protections apply to sensitive attributes [11]. Row-level filtering restricts data visibility based on the consumer's organizational context. Audit logging captures all access events in an immutable record that supports regulatory accountability and internal governance review.

A meaningful distinction between technical pipeline tables and business-certified data products improves analytical usability and reduces misinterpretation risk. Data products published through the access layer should carry business glossary definitions, certified quality scores, and documented freshness characteristics that allow consumers to evaluate their fitness for a specific analytical purpose [10][7]. Query performance optimization — including partition pruning, materialized view serving, and workload-aware caching — is applied within the access layer so that consumers benefit from execution efficiency without needing to implement optimization logic in their own queries [8].

Figure 2. Unified Data Access Layer — Consumer Interface Above Governed Data Product Layer



4. Metadata, Governance, and Data Quality

4.1 Data Catalog and Discovery

A data catalog does more than inventory datasets — it provides the organizational infrastructure through which data assets become searchable, understandable, and reusable across the enterprise [10]. Without a catalog, users in separate teams may independently build ingestion pipelines for the same source data, applying different transformation logic and producing different analytical outputs from

nominally identical inputs. This duplication is both wasteful and a source of analytical inconsistency that erodes trust in data-driven decisions [7].

Effective catalog design captures not only technical metadata — schema definitions, data types, partition keys, file formats — but also business metadata that communicates meaning and context to non-engineering consumers [10]. Dataset descriptions explain what a table represents and how it should be used. Whereas technical metadata, such as possession and sensitivity, define who is responsible for maintaining freshness and authenticity and what limitations and protection logs may be in place, the business glossary defines how these terms are used cross-organizationally to avoid confusion when source systems define the same terms differently [10][6].

Catalog discoverability is especially important in multi-cloud environments. Data assets distributed across multiple cloud platforms may be invisible to users operating in a different provider environment. A federated catalog presenting a unified inventory across all environments enables users to locate relevant datasets regardless of where they are physically stored [5][7]. Certification status indicators — distinguishing approved production datasets from experimental or deprecated assets — guide consumers toward appropriate data products and reduce the risk of analytical work being built on unstable or incorrect foundations.

4.2 Data Lineage and Compliance

Lineage is the connective tissue between the technical mechanics of data processing and the governance accountability that regulated industries require [3]. At the technical level, lineage records which source tables fed which transformation jobs, which columns were derived from which inputs, and which downstream tables depend on a given upstream dataset. At the business level, lineage connects these technical dependencies to the business rules and metric definitions that govern how data is interpreted. At the operational level, lineage records the execution history of each pipeline run — when it started, how long it took, whether it succeeded, and what volume of data it processed [3].

Lineage serves distinct purposes depending on who is using it. For data engineers, lineage supports impact analysis — understanding which downstream systems will be affected by a proposed schema change or a source data correction. For analysts, lineage provides confidence that a reported figure can be traced to a reliable, unmodified source. For compliance and audit functions, lineage supplies the evidentiary record that demonstrates how sensitive data was handled throughout the processing lifecycle [14][3].

Healthcare data pipelines that convert HL7-formatted clinical records to FHIR-compliant representations illustrate the compliance value of lineage clearly [12]. Each conversion step must be traceable: which records were transformed, what mapping rules were applied, and whether the output meets the structural and semantic requirements of the target standard. Auditable lineage transforms this traceability from a manual documentation effort into a platform-maintained record produced as a byproduct of normal pipeline execution.

Table 2. Data Lineage Types — Descriptions, Use Cases, and Regulated Industry Relevance

Lineage Type	Description	Primary Use Case	Regulated Industry Relevance
Technical Lineage	Tracks jobs, tables, columns, and transformation dependencies at the system level	Impact analysis for schema changes; dependency mapping for pipeline debugging	Demonstrates data handling paths for regulatory audit and examination
Business Lineage	Connects data movement to business rules, metric definitions, and reporting outcomes	Validates that reported figures align with defined business logic	Supports regulatory inquiries into how financial or clinical metrics were derived
Operational Lineage	Records pipeline execution history, failure events, retry attempts, and data freshness status	Monitors pipeline reliability; supports SLA measurement and incident response	Provides evidence of timely and complete data processing for compliance reporting

4.3 Data Quality Enforcement

Data quality that is only validated at the end of a pipeline is a posture that may allow significant processing to occur on incorrect data before any problem is detected [17]. Embedding quality enforcement across multiple pipeline stages — ingestion, transformation, and consumption — catches problems earlier, reduces the downstream impact of quality failures, and provides continuous visibility into data health rather than a binary pass/fail outcome at delivery.

At the ingestion stage, structural validation checks that incoming data conforms to expected schema patterns, that required fields are present, and that record volumes fall within anticipated ranges. Validation during the transformation step checks for acceptable ranges of derived values, join cardinalities, and the retention of non-duplicate records [9]. Validation during the consumption step checks the freshness of certified data products (i.e., were they refreshed according to their SLAs) and reconciliation of aggregate values against authoritative reference data sources [7].

Quantitative quality indicators — null rates, duplicate rates, rejection counts, reconciliation variances, and freshness timestamps — provide the measurability that allows both engineering teams and analytical consumers to reason about data reliability objectively [3]. Records that fail quality checks are routed to quarantine zones with full error detail and source identifiers, preserving the ability to investigate and remediate without introducing bad data into certified outputs or creating silent gaps in downstream analytical coverage [17].

5. Performance and Cost Optimization

5.1 Query and Storage Optimization

Query performance shapes analytical adoption more directly than almost any other platform characteristic. Users who encounter slow query response times may stop relying on governed lakehouse datasets and instead maintain local extracts or shadow environments that replicate the governed data without its governance controls [8][7]. Preventing this outcome requires that optimization be treated as a structural design responsibility rather than a reactive tuning exercise.

Partitioning remains the most impactful query optimization for large analytical datasets. By organizing data into logical file groups aligned with common filter dimensions — date, region, product category, source system — partitioning allows query engines to skip irrelevant data when filter conditions are present on the partitioned columns [2][8]. The compaction of small files produced by streaming writes into efficiently sized files reduces both query planning overhead and the metadata volume that query engines must traverse before reading data. Clustering on frequently filtered non-partition columns provides further scan reduction for high-selectivity query patterns.

Storage efficiency benefits from both format selection and lifecycle policy design. Columnar formats reduce storage volume substantially compared with row-oriented alternatives for analytical workloads [8]. Compression codecs provide additional volume reduction. Validation during the transformation step checks for acceptable ranges of derived values, join cardinalities, and the retention of non-duplicate records [9]. Validation

during the consumption step checks the freshness of certified data products (i.e., were they refreshed according to their SLAs) and reconciliation of aggregate values against authoritative reference data sources [7].

Table 3. Query and Storage Optimization Techniques

Technique	Applicable Layer	Expected Benefit	Implementation Consideration
Partition pruning	Curated and consumption zones	Reduces scan volume by reading only relevant data segments	Partition keys should align with dominant query filter patterns
File compaction	All zones; especially raw zone (streaming)	Reduces query planning overhead; improves parallel read efficiency	Schedule compaction during low-traffic windows to minimize compute contention
Clustering / Z-ordering	Curated zone	Improves scan efficiency for high-selectivity filter columns	Most effective for columns with high cardinality and frequent range filters
Materialized views	Consumption zone	Eliminates repeated aggregation execution for high-frequency queries	Refresh schedule must align with upstream data product freshness requirements
Result caching	Access layer	Reduces query engine load for identical repeated queries	Cache invalidation policies must reflect data product refresh frequency
Tiered storage lifecycle	Raw and archival zones	Reduces storage cost for aging or infrequently accessed datasets	Retention policies must satisfy compliance and audit hold requirements
Columnar format + compression	All zones	Reduces storage volume; improves I/O efficiency for analytical reads	Codec selection should balance compression ratio with decompression speed

5.2 Cost-Aware Scaling

Cloud-based lakehouse platforms decouple storage cost from compute cost and provide on-demand compute provisioning — advantages that are realized only when compute usage is actively governed [5][7]. Without workload-level cost visibility, the flexibility of on-demand compute can produce expenditure growth that is difficult to attribute, justify, or optimize.

Workload-level cost attribution assigns processing expenses to specific pipelines, data domains, or organizational teams. This attribution creates accountability and exposes specific sources of inefficiency — queries with unnecessarily broad scan patterns, clusters that remain provisioned during extended idle periods, or data movement charges generated by workloads that transfer data across cloud provider boundaries without necessity [8]. That allows optimization efforts to focus on the largest bottlenecks in the workload, rather than be applied indiscriminately with potentially negative effects on the most critical parts of the pipeline.

Also, tiering compute resources by workload criticality helps align cost with business value. Regulatory reporting jobs, fraud detection pipelines, and patient safety monitoring workloads might require dedicated capacity or a performance guarantee, to meet service-level targets. Exploratory analytics, model development, and non-time-sensitive batch processing are well-suited to spot or preemptible compute configurations that may reduce per-hour cost at the cost of occasional interruption [5].

6. Security and Multi-Cloud Risk Management

Security architecture in a multi-cloud lakehouse starts from a position of assumed perimeter exposure rather than perimeter defense. Data is distributed across multiple provider environments, accessed by diverse compute engines, and consumed by users operating from different organizational contexts. The security model must therefore rely on data-centric controls — encryption, identity verification, field-level protection, and audit logging — rather than on network boundary controls alone [11].

Encryption of data at rest and in transit should be applied uniformly. For datasets containing protected health information or financial account data, customer-managed encryption keys may be required by regulatory policy or organizational security

standards. Field-level masking and tokenization protect sensitive attributes in datasets accessed by consumers who require surrounding context but not the sensitive values themselves — for example, an analyst who requires transaction amounts and dates but does not require full account numbers [11]. Compliance scheme design for lakehouse environments should embed these controls at the data product layer rather than relying on downstream application-level enforcement [18].

Role-based access control policies must be managed in one place within the identity system and uniformly enforced across all cloud environments of the lakehouse, so that a dataset in a cloud environment of one provider that's designated as restricted is still considered restricted in other provider cloud environments of the lakehouse. It prevents access policy gaps for new cloud environments [5][11] and provides immutable audit logs with important evidence (identity, timestamp, dataset, and operation) for regulatory scrutiny in healthcare and financial services contexts [12][14].

Data residency compliance requires that geographic storage and processing constraints be enforced as governance controls rather than as informal conventions. The metadata layer should carry jurisdictional classification for sensitive datasets, and storage and processing policies should enforce regional boundaries automatically based on those classifications [5]. For decentralized data strategy models with data ownership split across domains, federated governance patterns should be established to denote residency and sensitivity policies that can be homogeneously respected by multiple datasets across domains [19]. Disaster recovery procedures should specify recovery time and recovery point objectives for critical data products, and ensure that backup, replication and failover configurations can be tested to meet those objectives.

7. Industry Applications

7.1 Healthcare Analytics

Healthcare data environments are characterized by high sensitivity, strict regulatory requirements, diverse data formats, and the direct connection between data quality and care outcomes. Electronic health records, claims submissions, patient monitoring streams, scheduling data, and population registry information each carry different format expectations, freshness requirements, and governance obligations [20][12]. A unified lakehouse architecture provides a governed foundation on which these sources may be integrated without requiring their homogenization into a single proprietary schema.

Streaming pipelines address the real-time requirements of clinical monitoring, admission event routing, and alert generation. Batch pipelines support the periodic workloads that characterize healthcare analytics: claims reconciliation, quality measure reporting, population health stratification, and cost variance analysis [13]. FHIR-based data conversion pipelines transform HL7-formatted clinical records into standardized representations that enable cross-system analytics while preserving semantic accuracy [12]. Governance controls implemented through the metadata layer ensure that patient-level data is classified, access-controlled, and audited in conformance with HIPAA requirements across the full processing lifecycle [11].

Healthcare organizations that consolidate fragmented analytical environments onto a lakehouse foundation may observe improvements in reporting timeliness, reductions in duplicate ingestion effort, and a more defensible audit posture for regulatory and accreditation reviews [20][13]. The shared governance foundation also supports longitudinal data analysis that is difficult to perform when patient records, claims data, and clinical observations reside in siloed, incompatible systems. Medallion Architecture patterns — organizing healthcare data through bronze, silver, and gold layers — provide a practical implementation template aligned with the lakehouse zone model [21].

7.2 Financial Services Analytics

Financial services analytics operates under constraints that simultaneously demand real-time operational responsiveness and rigorous historical accuracy. Fraud detection systems evaluate transaction events within tight latency windows following authorization requests. Regulatory risk reports require that every position figure be traceable to a verified source record. Customer analytics needs to account for activity across products, channels, and time periods using consistent business definitions [14][22].

The solution for this in the lakehouse architecture is a dual-pipeline approach. Transaction data is streamed to dashboards and fraud detection tools in near real time, and batch reconciled authoritative data sets are used for position reporting, regulatory reporting, and historical trend analysis. The

separation between real-time and batch outputs is managed through clearly defined freshness indicators in the data catalog, so that consumers understand which dataset is appropriate for their use case [7][3].

For governance, a financial services lakehouse may deploy field level masking of account numbers and transaction amounts in non-production environments, role-based access control to compliance-sensitive datasets, and immutable lineage from reported numbers back to source records[3][11]. These controls support the regulatory examination process and reduce the operational risk associated with unauthorized data access or analytical inconsistency in reporting environments.

Table 4. Industry Application Characteristics — Healthcare and Financial Services

Use Case	Pipeline Type	Governance Requirement	Key Challenge	Lakehouse Capability Applied
Clinical monitoring and alerting	Real-time streaming	HIPAA access control; audit logging	Sub-minute event latency with durable storage	Streaming to raw zone; curated real-time tables; RBAC enforcement
Population health analysis	Batch	HIPAA; data use agreements; FHIR compliance	Integrating heterogeneous clinical and claims sources	Zone-based pipeline; FHIR conversion; catalog with sensitivity classification
Claims reconciliation and reporting	Batch	HIPAA; regulatory quality standards	Late-arriving and amended claims records	Partition reprocessing; time-travel versioning; quality quarantine
Real-time fraud detection	Real-time streaming	PCI-DSS; field-level masking; audit trail	High-volume event throughput with low false-positive rate	Streaming pipeline; event deduplication; access-layer field masking
Regulatory risk reporting	Batch	SOX-adjacent controls; lineage documentation	Traceability from the reported figure to the source record	Technical and business lineage; reconciliation quality checks; immutable audit logs
Transaction monitoring dashboard	Real-time + batch hybrid	Role-based access; data residency compliance	Reconciling real-time and batch views of the same transaction	Dual freshness indicators in catalog; unified access layer; tiered compute

8. Implementation Roadmap

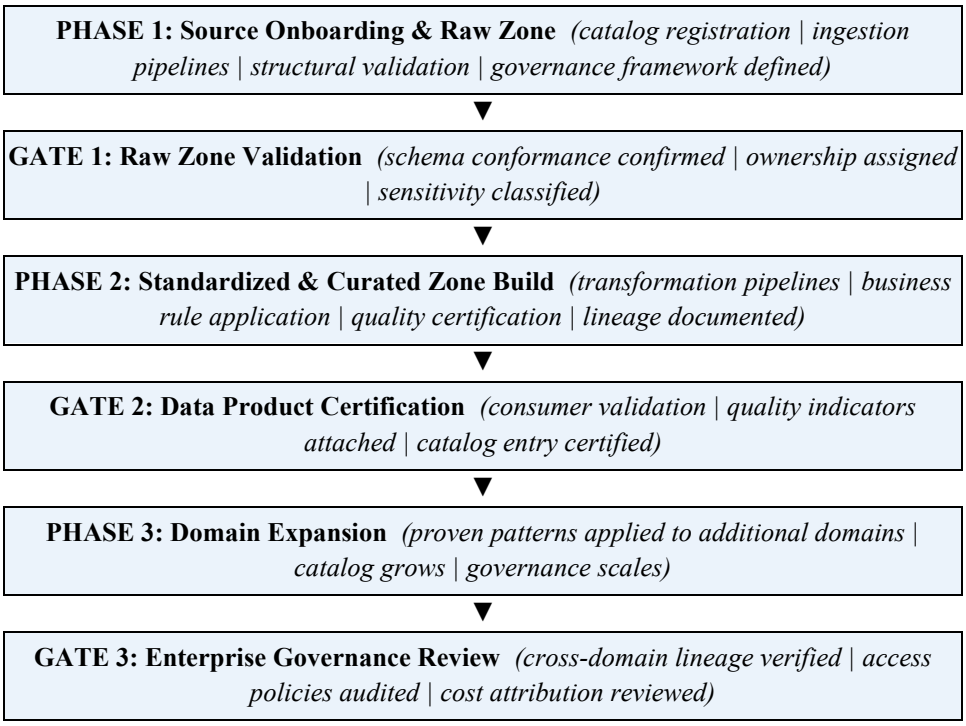
Successful lakehouse adoption typically follows an incremental path that begins with a high-value, well-bounded domain and expands systematically as governance standards and pipeline patterns mature [7][17]. An attempt to migrate all analytical workloads simultaneously introduces unnecessary coordination risk and may delay the delivery of business value while the full platform is being built.

The initial phase focuses on source registration and raw zone establishment. Data sources are cataloged with ownership, sensitivity, schema expectations, and refresh frequency. Ingestion pipelines land data into the raw zone with minimal transformation and apply structural validation at arrival. The catalog becomes the authoritative inventory of data assets, and the governance framework for ownership, quality, and access is defined before data products are built [10].

In the second phase, standardized and curated zone pipelines are developed for the target domain. Transformation logic is modular, documented, and quality-checked at each stage. Certified data products are published to the consumption layer and validated by their intended consumers before formal certification. Lineage is documented from source to consumption for each data product [9][6].

The third phase extends the architecture to additional domains using the governance patterns, pipeline templates, and quality frameworks established in the initial domain. This replication of proven patterns reduces per-domain onboarding effort and ensures consistency across the growing catalog of data products [7].

Figure 3. Implementation Roadmap — Phased Lakehouse Adoption with Validation Gates



9. Conclusion

The unified data lakehouse architecture addresses a structural challenge that many enterprise data environments have not solved: enabling diverse analytical workloads to operate over a shared, governed, and trusted data foundation without requiring data duplication, inconsistent governance, or platform proliferation. By integrating open table format storage, zone-based data organization, real-time and batch pipeline coordination, metadata governance, and a unified access abstraction, the architecture reduces fragmentation and may improve the reliability of data-driven decision-making.

The framework's applicability in regulated industries rests on its governance characteristics as much as its technical capabilities. Lineage documentation, quality certification, access control enforcement, and audit logging provide the evidentiary foundation that compliance and regulatory requirements demand. The multi-cloud design principles ensure that these governance characteristics are maintained consistently as organizational data footprints continue to expand across provider boundaries.

Organizations that adopt this architecture incrementally — beginning with a high-value domain and extending systematically — may establish a governed analytical foundation that supports current workloads while remaining extensible to future requirements as data volumes, workload complexity, and regulatory expectations continue to evolve.

References

[1] M. Armbrust, A. Ghodsi, R. Xin, and M. Zaharia, "Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics," in Proc. 11th Conf. on Innovative Data Systems Research (CIDR), 2021. Available: http://cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf

[2] 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. Available: https://dl.acm.org/doi/10.14778/3415478.3415560

[3] S. Nadal et al., "Operationalizing and Automating Data Governance," J. Big Data, vol. 9, 2022. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9736715/

[4] Dražen Oreščanin and Tomislav Hlupić, "Data Lakehouse — A Novel Step in Analytics Architecture," in 2021 44th International

Convention on Information, Communication and Electronic Technology (MIPRO), 2021. Available: https://ieeexplore.ieee.org/document/9597091/

[5] S. A. Errami, H. Hajji, K. A. El Kadi, and H. Badir, "Spatial Big Data Architecture: From Data Warehouses and Data Lakes to the LakeHouse," J. Parallel Distrib. Comput., vol. 176, pp. 70–79, 2023. Available: https://doi.org/10.1016/j.jpdc.2023.02.007

[6] T. Hlupić; D. Oreščanin; D. Ružak; M. Baranović, "An Overview of Current Data Lake Architecture Models," in 2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO), 2022. Available: https://ieeexplore.ieee.org/document/9803717/

[7] Ahmed A. Harby, Farhana Zulkernine a, "Data Lakehouse: A Survey and Experimental Study," Inf. Syst., vol. 127, 2025. Available: https://doi.org/10.1016/j.is.2024.102460

[8] D. Durner, V. Leis, and T. Neumann, "Exploiting Cloud Object Storage for High-Performance Analytics," Proc. VLDB Endow., vol. 16, no. 11, pp. 2769–2782, 2023. Available: https://dl.acm.org/doi/10.14778/3611479.3611486

[9] Y. Zhao et al., "A Zone-Based Data Lake Architecture for IoT, Small and Big Data," in IDEAS '21: Proceedings of the 25th International Database Engineering & Applications Symposium, 2021. Available: https://dl.acm.org/doi/10.1145/3472163.3472185

[10] A. Boufassil et al., "Data Catalog: Approaches, Trends, and Future Directions," in 2023 17th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), 2024. Available: https://ieeexplore.ieee.org/document/10472799/

[11] R. Gupta et al., "Secured and Privacy-Preserving Multi-Authority Access Control System for Cloud-Based Healthcare Data Sharing," Sensors, vol. 23, no. 5, 2023. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10007450/

[12] T. K. H. Le et al., "Enhancing Healthcare Interoperability with FHIR: A Systematic Approach to Online Data Management," in ICISE '24: Proceedings of the 2024 9th International Conference on Information Systems Engineering, 2024. Available: https://dl.acm.org/doi/10.1145/3711954.3711958

[13] D. Chrimes, "Big Data Analytics of Predicting Annual US Medicare Billing Claims with Health Services," in 2022 IEEE International Conference on Big Data (Big Data), 2023. Available: https://ieeexplore.ieee.org/document/10020524/

[14] A. Kumar et al., "Integrating Big Data Analytics with Financial Risk Management: Challenges and Opportunities," in 2025 Seventh International Conference on Computational Intelligence and Communication Technologies (CCICT), 2025. Available: https://ieeexplore.ieee.org/abstract/document/11087991/

[15] K. Peddireddy, "Streamlining Enterprise Data Processing, Reporting and Realtime Alerting using Apache Kafka," in 2023 11th International Symposium on Digital Forensics and Security (ISDFS), 2023. Available: https://ieeexplore.ieee.org/document/10131800

[16] M. Zaharia et al., "Apache Spark: A Unified Engine for Big Data Processing," Communications of the ACM, vol. 59, no. 11, pp. 56–65, 2016. Available: https://dl.acm.org/doi/10.1145/2934664

[17] L. Zineb and F. Rachid, "ETL Technologies for Big Data: A Comparative Study," in 2023 IEEE International Conference on Advances in Data-Driven Analytics And Intelligent Systems (ADACIS), 2024. Available: https://ieeexplore.ieee.org/document/10424341

[18] C. Ma, X. Hu et al., "A Data Analysis Privacy Regulation Compliance Scheme for Lakehouse," in ADMIT '23: Proceedings of the 2023 2nd International Conference on

Algorithms, Data Mining, and Information Technology, 2023. Available: https://dl.acm.org/doi/abs/10.1145/3625403.3625405

[19] Vijay Kumar Butte and Sujata Butte, "Enterprise Data Strategy: A Decentralized Data Mesh Approach," 2022 International Conference on Data Analytics for Business and Industry (ICDABI), 2023. Available: https://ieeexplore.ieee.org/document/10041672/

[20] E. Begoli, I. Goethert, and K. Knight, "A Lakehouse Architecture for the Management and Analysis of Heterogeneous Data for Biomedical Research and Mega-biobanks," in 2021 IEEE International Conference on Big Data (Big Data), 2021. Available: https://ieeexplore.ieee.org/document/9671534/

[21] V. K. Pamula, "The Medallion Architecture in Practice: A Framework for Building Scalable and Governed Data Lakehouses on Microsoft Fabric," in 2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC), 2025. Available: https://ieeexplore.ieee.org/document/11395677/

[22] D. Chowdhury and P. Kulkarni, "Application of Data Analytics in Risk Management of Fintech Companies," in 2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA), 2023. Available: https://ieeexplore.ieee.org/abstract/document/10099795/