© OCT 2025 | IRE Journals | Volume 9 Issue 4 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I4-1711258-3166

Data Lakehouse Architecture: Bridging the Gap Between Data Lakes and Data Warehouses

MAYA THOMAS[1], LAVANYA GONSALEZ[2], RIBIN JACOB[3], TINCY MATHEW[4]
[1] *Sikkim Manipal Institute of Technology, Gangtok, India*
[2] *Krishna Kanta Handiqui State Open University, Rani Guwahati, India*
[3] *Sikkim Manipal Institute of Technology, Gangtok, India*
[4] *Sohra Government College, Cherrapunji, India*

Abstract*— The exponential growth of enterprise and scientific data has challenged longstanding assumptions in data management. Traditional data warehouses deliver mature relational semantics and predictable performance, but struggle with semi-structured modalities, iterative data science, and real-time signals. Data lakes, by contrast, scale elastically on commodity object stores and support diverse data types through schema-on-read, yet historically lacked transactional guarantees, strong governance, and consistent query performance. The Data Lakehouse architecture reconciles these trade-offs by layering warehouse-like ACID transactions, versioned metadata, and query optimization over open file formats in a cloud-native design. This paper provides a deep, holistic treatment of Lakehouse principles and practice. We (i) trace the intellectual lineage from MapReduce, Dremel, and Hive to modern log-structured table formats; (ii) formalize a reference architecture encompassing storage, transaction/metadata, and processing layers with a cross-cutting governance plane; (iii) present a comparative analysis of Delta Lake, Apache Iceberg, and Apache Hudi; (iv) synthesize performance considerations for vectorized execution, small-file mitigation, and streaming upsets; (v) examine governance and interoperability patterns for multi-cloud deployments; and (vi) explore emerging directions-including vector/tensor extensions for AI, zero-ETL pipelines, semantic integration, and carbon-aware optimization. Throughout, we anchor discussion in peer-reviewed literature and production learnings, retaining resolvable DOIs for all referenced works. The result is a practitioner-ready, research-grounded blueprint for building resilient, interoperable, and AI-native data platforms.

***Index Terms*— Data Lakehouse, Delta Lake, Apache Iceberg, Apache Hudi, Parquet, ORC, ACID Transactions, Metadata Governance, Big Data Architecture, Cloud Analytics, Machine Learning, Vector Databases**

I. INTRODUCTION

Enterprises today generate unprecedented volumes of heterogeneous data: transactional records, clickstream logs, IoT telemetry, images and video, natural-language text, clinical measurements, and scientific experiments. Deriving value from such diversity requires an architecture that simultaneously supports batch analytics, interactive BI, real-time event processing, and iterative machine learning. Historically, organizations oscillated between two poles: data warehouses, emphasizing schema-on-write, optimized relational execution, and strong consistency; and data lakes, embracing schema-on-read, minimal ingestion friction, and scale-out storage on commodity cloud object stores. Each pole imposes structural compromises. Warehouses are costly for petabyte-scale retention, inflexible for rapidly evolving schemas, and suboptimal for unstructured modalities. Lakes are agile and inexpensive but can devolve into data swamps when metadata is inconsistent, governance is weak, and transactional correctness is absent. Data Lakehouse seeks to merge these affordances by combining object-storage economics and openness with warehouse-grade reliability. This is realized via log-structured table formats, notably Delta Lake [1], Apache Iceberg [11], and Apache Hudi—which add ACID transactions, schema evolution, and time travel to open columnar files (Parquet/ORC) [9], [10]. Compute engines (Spark, Trino, Flink, and Photon) [8], [3] then exploit vectorization, statistics, and pushdown to achieve competitive performance. This manuscript expands prior surveys of lakes [13], [14] and enterprise overviews [15] by: (1) elucidating architectural invariants underpinning lakehouses; (2) systematizing design trade-offs across table formats; (3) synthesizing empirical insights from benchmarks such as LST-Bench [4]; and (4) articulating a research agenda for interoperability, AI-native

storage, and sustainability. We retain citations and DOIs as published to preserve traceability.

II. HISTORICAL CONTEXT AND EVOLUTION

A. From Warehouses to Lakes

In the 1990s and early 2000s, enterprise data management centered on centralized warehouses. ETL pipelines materialized dimensional schemas for reporting and OLAP. Although performant, these systems struggled with semi-structured inputs and iterative analytics. A methodological pivot arrived with MapReduce [5], [22], which abstracted large-scale parallelism over commodity clusters. Dremel [6], [12] demonstrated interactive, massively parallel query processing over nested columnar storage, influencing the Parquet/ORC ecosystem. Hive [7] adapted SQL semantics for the Hadoop stack. Spark unified iterative, streaming, and batch computation [8], fueling schema-on-read data lakes built atop HDFS and, later, cloud object stores. Despite elasticity and openness, data lakes often lacked robust metadata curation, lineage, and transactions, impairing trust and reusability. Without strong governance, organizations accumulated countless small files, inconsistent schemas, and ad hoc ingestion patterns, degrading query performance and reproducibility.

B. From Lakes to Lakehouses

Lakehouses emerged to address these gaps by introducing table-level transaction logs, snapshot isolation, and schema evolution over immutable files. Delta Lake [1] pioneered an append-only transaction log with checkpointing; Iceberg [11] advanced manifest trees and partition-spec evolution for petabyte scale; Hudi focused on low-latency CDC via copy-on-write and merge-on-read. Collectively, these innovations restored correctness and governance while retaining open formats. Cloud vendors converged on lakehouse-aligned offerings: BigLake [2] unifies BigQuery governance with multi-cloud object storage; analogous controls exist in AWS and Azure.

C. Convergence and Interoperability

Modern enterprises operate heterogeneous stacks. Interoperability efforts (e.g., XTable [16]) translate metadata across Delta/Iceberg/Hudi, enabling cross-engine reads/writes and reducing migration friction. The field is coalescing toward open, engine-agnostic semantics while vendor ecosystems differentiate on performance and tooling.

III. CONCEPTUAL ARCHITECTURE AND DESIGN PRINCIPLES

We model a Lakehouse as three core layers: storage, metadata/transactions, and processing/consumption, governed by an orthogonal plane for security, lineage, and quality.

A. Storage: Open Columnar on Object Stores.

Object stores (e.g., S3, ADLS, GCS) offer durability, elasticity, and cost efficiency. Parquet and ORC encode columnar data with compression, encoding dictionaries, and statistics for predicate pruning [9], [10]. Practitioners should target file sizes (typically 128-1024MB) that balance overhead and parallelism, and use partitioning and clustering informed by workload predicates.

B. Transactions and Metadata: Log-Structured Tables

Delta Lake maintains a monotonic log of atomic commits and optimizes with checkpoint files to cap planning cost [1]. Iceberg represents table state as snapshots that reference manifest lists and data files, enabling scalable planning and efficient row-level mutation [11]. Hudi exposes a commit timeline with instant-based versioning; merge-on-read balances ingestion latency with compaction overhead. LST Bench [4] formalizes evaluation axes: update intensity, small file pressure, multi-writer concurrency, and snapshot-planning latency.

D. Processing: Multi-Engine, Multi-Modal

Spark, Trino, and Flink access the same tables, enabling SQL, streaming, and ML. Photon demonstrates that vectorized execution and runtime code generation can meet or exceed the performance of closed warehouses on Lakehouse tables [3]. Engines exploit Parquet/ORC statistics, partition specs, and table-metadata indices for pushdown and data skipping. D. Governance: Catalog, Lineage, and Policy Governance unifies identity and access management, schema/version control, data-quality rules, and lineage. Surveys emphasize that automated metadata capture, constraint enforcement, and privacy-aware controls are prerequisites for sustainable lakes [13], [14]. In multi-cloud contexts, the BigLake model [2] illustrates federated policy enforcement.

© OCT 2025 | IRE Journals | Volume 9 Issue 4 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I4-1711258-3166

IV. ADVANTAGES AND STRATEGIC OUTCOMES

A lakehouse consolidates ETL, BI, and ML on a single substrate, reducing copy-and-transform cycles and associated data drift. Additional benefits include:

- Economic efficiency via decoupled storage/compute and open formats (lower TCO, minimized lock-in).
- Reliability from ACID transactions, snapshot isolation, and governed schemas.
- Performance through vectorized execution, clustering, and statistics-driven pruning.
- Reproducibility via time travel and immutable snapshots underpinning rigorous analytics and audits. AI-readiness by exposing curated features and labels directly on governed tables, reducing extract duplication.

Industry analyses report substantial cost and cycle-time reductions following lakehouse adoption [15].

V. COMPARATIVE ANALYSIS OF TABLE FORMATS

We compare the three dominant table formats along with capabilities pertinent to enterprise deployment. Evidence draws on [4], [11], [16] and vendor-neutral observations. Observations. Delta Lake emphasizes compaction and log checkpointing, yielding robust upsert performance under mixed streaming/batch. Iceberg's manifest-based planning scales snapshot enumeration at extreme file counts and supports position deletes. Hudi's merge-on-read offers ingestion latency advantages but relies on compaction policy tuning for read performance. Interoperability layers (e.g., XTable [16]) mitigate format lock-in for multi-engine estates. Table I summarizes the comparative study.

VI. USE CASES AND SECTORAL PATTERNS

A. Healthcare and Life Sciences

Healthcare and biomedical research generate diverse datasets; electronic health records (EHR), diagnostic imaging, genomics, clinical notes, and wearable telemetry. Integrating such data requires strong governance, auditability, and interoperability while preserving privacy and compliance. The Data Lakehouse provides a practical foundation by consolidating heterogeneous sources under ACID-compliant transactional control and enabling reproducible cohort analysis.

Recent studies, such as the IEEE publication on AI-Driven Data Lakehouse for Healthcare [20], demonstrate how lakehouse architecture can enhance interoperability, improve diagnostic accuracy, and support real-time anomaly detection for clinical decision support. The framework outlined in that work integrates secure data acquisition, unified metadata models, and advanced machine learning to drive predictive analytics in healthcare.

Hospitals and public-health institutions increasingly employ lakehouse-based solutions for population health management, risk stratification, and operational forecasting, illustrating the societal benefits of this architecture when combined with responsible AI and robust data governance.

B. Financial Services

Banks combine transactional ledgers, behavioral telemetry, and third-party risk datasets. Snapshot isolation and lineage underpin regulatory reporting; streaming upserts enable near real-time fraud detection. Partition-evolution (Iceberg) and log compaction (Delta) facilitate long-horizon analytics.

C. Academic and Research Institutions

Research ecosystems integrate publications, citation graphs, grants, and datasets. ALITE [19] exemplifies scalable integration logic. Lakehouses buttress FAIR data principles, enabling reproducible, cross-domain analytics.

D. Manufacturing and IoT

Industrial telemetry requires low-latency ingestion with consistent analytical views. Hudi's merge-on-read patterns pair with curated batch views for planning/optimization; Iceberg's scalable snapshots assist global estates executing multi-writer workloads.

© OCT 2025 | IRE Journals | Volume 9 Issue 4 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I4-1711258-3166

TABLE I
COMPARISON OF LEADING DATA LAKEHOUSE TABLE FORMATS

Feature	*Delta Lake*	*Apache Iceberg*	*Apache Hudi*
Metadata Model	Append-only transaction log (JSON/Parquet) with checkpointing	Hierarchical manifests and snapshot metadata	Commit timeline with instant versioning
ACID Transactions	Full ACID compliance via optimistic concurrency	Snapshot isolation with atomic replace commits	ACID on copy-on-write; eventual for merge-on-read
Schema Evolution	Supports add/drop columns; automatic type promotion	Full schema evolution with partition spec updates	Supports column evolution and nullable enforcement
Time Travel / Versioning	Supported through commit history.	Supported via snapshot rollback	Supported via commit instants.
Streaming Support	Strong: micro-batch and streaming merges	Moderate: via Flink and Spark streaming connectors	Strong: native incremental pull and CDC ingestion
Upsert/Delete Efficiency	Optimized via data skipping and compaction	Efficient delete/merge through position deletes.	Native merge and delta streamer
Multi-engine Interoperability	Broad (Spark, Trino, Presto, Photon)	Broad (Spark, Flink, Trino, Snowflake)	Growing (Spark, Hive, Presto)
Governance Integration	Unity Catalog, AWS Glue, HMS	Hive Metastore, AWS Glue, custom catalogs	Hive Metastore, custom catalogs
Typical Use Cases	Unified analytics + ML, mixed workloads	Petabyte-scale batch analytics; multi-writer concurrency	Real-time ingestion, CDC-heavy pipelines

VII. PERFORMANCE ENGINEERING IN LAKEHOUSES

A. Query Execution and Vectorization

Photon's results [3] demonstrate that vectorized execution, runtime codegen, and cache-aware operators can rival closed warehouses on lakehouse tables. Parquet's nested encoding continues to evolve; efficient scanning of nested structures improves complex analytics [21].

B. File and Layout Policies

Right-sizing Parquet/ORC files, materializing statistics, and clustering data by high-selectivity predicates reduce scan volume. Avoiding small files requires compaction and write rate control. Layout strategies should co-evolve with query patterns, e.g., adaptive Z-ordering or manifest-level clustering.

C. Metadata Scalability

Iceberg's snapshot/manifest structure reduces planning overhead at billions of files [11]. Delta's checkpoints bound log replay; Hudi's timeline organizes instants for concurrent writers. LST-Bench [4] quantifies differences across update-heavy and multi-writer regimes.

D. Streaming Upserts and CDC

CDC requires sophisticated merge semantics with late-arriving events and idempotency. Delta's MERGE INTO with structured streaming and Hudi's DeltaStreamer illustrate patterns for

© OCT 2025 | IRE Journals | Volume 9 Issue 4 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I4-1711258-3166

reconciling operational and analytical states without dual data stores.

VIII. GOVERNANCE, SECURITY, QUALITY, AND LINEAGE

A. Catalogs and Policy

Central catalogs (Glue, Hive Metastore, Unity Catalog) hold schema versions, ACLs, and lineage. Column- and row-level security integrate with identity providers. Surveys [13], [14] stress continuous data-quality checks (freshness, uniqueness, referential integrity) and automated anomaly detection.

B. Lineage and Reproducibility

Transaction logs and snapshots furnish runtime lineage; integrating with orchestration metadata forms end-to-end provenance graphs. Time travel anchors reproducible analytics by pinning queries to immutable table states. These lineage artifacts not only enable auditability but also serve as a foundation for impact analysis, allowing teams to trace how downstream metrics or models are influenced by upstream schema or data changes. Furthermore, coupling lineage with policy-aware metadata supports automated compliance reporting and facilitates trust in data-driven decision systems.

C. Multi-Cloud Governance

Federated policy enforcement across clouds and regions is essential for global data estates. BigLake [2] models cross-location access with consistent semantics, foreshadowing wider standardization.

IX. INTEROPERABILITY AND OPEN STANDARDS

A. Format Interop

As estates mix Delta, Iceberg, and Hudi, cross-format metadata interoperability becomes critical. XTable [16] demonstrates bidirectional translation with minimal duplication, enabling incremental migration and choice of engines.

B. Commit and Statistics Semantics

Standardizing commit semantics (conflict detection, isolation levels) and file-level statistics (min/max, null counts, bloom filters) would enable more portable optimizations while preserving innovation. Establishing a unified specification across table formats would simplify cross-engine interoperability, ensuring consistent behavior for concurrent writes, schema evolution, and query planning. In addition, harmonized metadata definitions would facilitate advanced cost-based optimization and allow intelligent caching or prefetching strategies to operate uniformly across heterogeneous compute environments.

X. AI, ML, AND VECTOR/TENSOR EXTENSIONS

A. Vector-Native Tables

Modern AI stacks require efficient storage for embeddings. Delta Tensor [17] proposes vector/tensor types in lake tables, enabling co-resident BI and vector search. Tensor Lakehouse [18] targets scalable model-training corpora. These patterns underpin RAG, recommendations, and multimodal analytics without separate vector silos.

B. Feature Stores and Reproducibility

Lakehouse tables serve as durable feature stores with snapshot isolation, ensuring consistent alignment between model training and serving. Governance policies can protect PII/PHI while enabling masked experimentation.

XI. CHALLENGES AND OPEN PROBLEMS

A. *Metadata Explosion*

At exabyte scale, listing operations, snapshot planning, and commit contention become acute. Research directions include predictive compaction, workload-aware clustering, and learned cost models [4].

B. Cross-Cloud Latency and Policy Drift

Geographically distributed data requires policy convergence and latency-aware planning. BigLake's patterns [2] suggest designs for federated governance.

C. Semantic Integration and Entity Resolution

Automating schema matching and entity resolution (e.g., ALITE [19]) reduces manual curation. Embedding-based similarity over governed tables offers promising improvements in accuracy.

D. Benchmarking Beyond SQL

LST-Bench [4] is a start. The community needs benchmarks for streaming joins, vector search, and ML ETL to capture real workloads.

E. Sustainability

Carbon-aware scheduling, cache placement, and storage tiering are underexplored. Lakehouses can expose energy metrics to optimizers for greener plans.

XII. ARCHITECTING LAKEHOUSES IN PRACTICE

A. *Design Heuristics*

- Choose table format by workload: heavy CDC/low latency (Hudi); massive batch with complex deletes (Iceberg); mixed workloads and ecosystem breadth (Delta).
- Enforce naming, partitioning, and schema-evolution conventions to avoid drift.
- Instrument lineage and data-quality checks early to prevent swamp regression.
- Co-design compaction and clustering with query predicates; reassess quarterly.
- Separate storage and compute accounts/projects for blast radius control.

B. Migration Patterns

Incremental migration via external tables and interop (XTable [16]) reduces cutover risk. Start with non-critical domains, validate lineage, and expand.

C. Operations

Define SLOs for freshness, latency, and correctness. Automate compaction/optimize jobs. Track cost and performance regressions; iterate on partition specs and file sizes.

XIII. FUTURE OUTLOOK

Deeper unification, stronger guarantees, and greater automation across the data lifecycle will define the next generation of data lakehouses. First, zero-ETL pipelines will mature from point integrations to policy-driven materialization, where operational changes propagate to governed lakehouse tables with bounded freshness and declarative conflict resolution. This will be paired with true streaming-by-default semantics, exactly-once, idempotent merges, and late-data reconciliation as first-class behavior across engines and table formats.

Second, the lakehouse will become AI-native. Vector and tensor-aware tables will co-reside with classical columns, enabling retrieval-augmented generation, recommendation, and multimodal analytics without separate vector silos. Model-aware governance (feature lineage, consent provenance, and license tracking) will turn reproducibility from aspiration into default. Accelerated compute (GPU/TPU/DPU) will be scheduled jointly with data placement, merging query planning with model serving to minimize data movement.

Third, interoperability will advance beyond metadata translation toward shared transaction semantics, statistics schemas, and row/column-level security models. Cross-cloud sharing will stabilize around portable, cryptographically verifiable manifests, making "read anywhere, govern centrally" routine. Domain-oriented data mesh will ride atop these standards, with contract-driven SLAs and autonomous quality remediation.

Fourth, privacy and safety will move from bolt-ons to compiled-in guarantees. Native differential privacy, policy-aware query optimization, confidential computing enclaves, and audit-by-construction logs will enable high-utility analytics under strict regulatory regimes. Semantic layers will encode organizational ontologies, letting users ask business questions while engines compile optimized, policy-compliant plans.

Fifth, sustainability will become a planning objective. Carbon-aware cost models will steer storage tiering, compaction cadence, and accelerator selection, exposing energy metrics for governance and reporting. Finally, autonomous operations, learned compaction, adaptive partitioning, skew-aware shuffles, and reinforcement-learned caching, will close the loop between observability and optimization. In sum, the lakehouse is evolving into an open, intelligent substrate where BI, AI, and real-time decisioning converge, governed, portable, and increasingly self-optimizing.

CONCLUSION

The Data Lakehouse architecture has become a defining milestone in the evolution of data management. By integrating the flexibility of data lakes with the governance, performance, and transactional integrity of warehouses, it delivers a unified, scalable, and open foundation for analytics and artificial intelligence. Through innovations in log-structured metadata, ACID transactions, and

open columnar formats, the lakehouse resolves long-standing challenges of data duplication, schema rigidity, and inconsistent governance. Empirical research and industry adoption show that modern lakehouses now rival proprietary warehouses in performance while offering cost efficiency and interoperability. Beyond analytics, the lakehouse is transforming how enterprises operationalize data for real-time intelligence, machine learning, and domain-driven design. With developments such as vector-native tables, federated governance, and zero-ETL architectures, the ecosystem is advancing toward intelligent, self-optimizing platforms that support both business and scientific innovation. Yet, challenges remain in metadata scalability, cross-cloud governance, and sustainability areas ripe for continued research. Ultimately, the Data Lakehouse represents more than a convergence of technologies; it embodies a paradigm shift toward openness, reproducibility, and agility. As organizations increasingly rely on data for strategic differentiation, the lakehouse stands poised to serve as the cornerstone of next-generation, AI-driven enterprises.

REFERENCES

[1] M. Armbrust et al., "Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores," PVLDB, vol. 13, no. 12, 2020. doi:10.14778/3415478.3415560

[2] J. Levandoski et al., "BigLake: BigQuery's Evolution Toward a Multi Cloud Lakehouse," SIGMOD, 2024. doi:10.1145/3626246.3653388

[3] T. B. Samwel et al., "Photon: A Fast Query Engine for Lakehouse Systems," SIGMOD, 2022. doi:10.1145/3514221.3526054

[4] J. Camacho-Rodr´ıguez et al., "LST-Bench: Benchmarking Log-Structured Tables in the Cloud," Proc. ACM on Management of Data, 2024. doi:10.1145/3639314

[5] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Commun. ACM, 51(1), 2008. doi:10.1145/1327452.1327492

[6] S. Melnik et al., "Dremel: Interactive Analysis of Web-Scale Datasets," VLDB, 2010. doi:10.1145/1953122.1953148

[7] A. Thusoo et al., "Hive: A Warehousing Solution over a Map-Reduce Framework," PVLDB, 2009. doi:10.14778/1687553.1687609

[8] M. Zaharia et al., "Apache Spark: A Unified Engine for Big Data Processing," Commun. ACM, 59(11), 2016. doi:10.1145/2934664

[9] X. Zeng et al., "An Empirical Evaluation of Columnar Storage Formats," PVLDB, 17(1), 2023. doi:10.14778/3626292.3626298

[10] T. Ivanov et al., "The Impact of Columnar File Formats on SQL on-Hadoop," Concurrency and Computation: Practice and Experience, 32(20), 2020. doi:10.1002/cpe.5523

[11] A. Okolnychyi et al., "Petabyte-Scale Row-Level Operations in Data Lakehouses," PVLDB, 17(12), 2024. doi:10.14778/3685800.3685834

[12] S. Melnik et al., "Dremel: A Decade of Interactive SQL Analysis at Web Scale," PVLDB, vol. 13, no. 12, 2020. doi:10.14778/3415478.3415568

[13] R. Hai et al., "Data Lakes: A Survey of Functions and Systems," IEEE TKDE, 35(12), 2023. doi:10.1109/TKDE.2023.3270101

[14] P. Wieder and H. Nolte, "Toward Data Lakes as Central Building Blocks for Data Management and Analysis," Frontiers in Big Data, 2022. doi:10.3389/fdata.2022.945720

[15] A. Nambiar and D. Mundra, "An Overview of Data Warehouse and Data Lake in Modern Enterprise Data Management," Big Data and Cognitive Computing, 6(4), 2022. doi:10.3390/bdcc6040132

[16] A. Agrawal et al., "XTable in Action: Seamless Interoperability in Data Lakes," arXiv, 2401.09621, 2024. doi:10.48550/arXiv.2401.09621

[17] Z. Bao et al., "Delta Tensor: Efficient Vector and Tensor Storage in Delta Lake," arXiv, 2405.03708, 2024. doi:10.48550/arXiv.2405.03708

[18] R. Kienzler et al., "Tensor Lakehouse for Foundation Model Training," arXiv, 2309.02094, 2023. doi:10.48550/arXiv.2309.02094

[19] A. Khatiwada et al., "Integrating Data Lake Tables," PVLDB, 16(4), 2022. doi:10.14778/3574245.3574274

[20] S. M. Shaffi, S. Vengathattil, and J. Mehta, "Enhancing Cloud Security Through AI-Driven Anomaly Detection and Advanced Machine Learning Algorithms," Proc. 8th IEEE Int. Symp. on Big Data and Applied

Statistics (ISBDAS 2025), 2025. doi:10.1109/ISBDAS64762.2025.11116832

[21] N. Rey et al., "Nested Parquet Is Flat, Why Not Use It? How To Scan Nested Parquet Efficiently," SIGMOD, 2025. doi:10.1145/3725329

[22] J. Dean and S. Ghemawat, "MapReduce: A Flexible Data Processing Tool," Commun. ACM, 53(1), 2010. doi:10.1145/1629175.1629198