Vol. 1, No. 3, pp.: 66–84, 2023

Data Lakehouse Architectures for AI Infrastructure: A Systematic Review of Storage, Metadata, and Governance

Daniel R. Okafor[1], Mei-Ling Tan[2,*], Sven A. Halvorsen[1], Priya Raghunathan[3]

[1] Department of Computer Science, Institute for Data Systems Engineering, Trondheim 7034, Norway

[2] School of Computing and Information Systems, National Data Infrastructure Laboratory, Singapore 117417, Singapore

[3] Center for Computational Discovery, Pacific Institute of Technology, Berkeley, CA 94720, USA

* *mei-ling.tan@ndil-lab.edu.sg*

Article Information	
Received	16 April 2023
Accepted	28 August 2023
DOI	https://doi.org/10.63646/datamind.2023.010306

Abstract

The data lakehouse has emerged as the dominant architectural pattern for serving both analytical SQL and machine-learning workloads from a single copy of data held on low-cost cloud object storage. Yet the engineering problem at its core is frequently underspecified: object stores expose a key–value interface with high per-object latency and weak listing consistency, which makes performant, transactional, and governable table storage difficult to achieve. This article presents a systematic review of lakehouse architectures organised around three engineering pillars—the storage layer (open columnar formats and physical optimisation), the metadata layer (transaction logs, manifests, catalogs, lineage, and discovery), and the governance layer (access control, policy, data quality, and auditability). Following a PRISMA-style protocol, 1,284 records were screened and 30 peer-reviewed studies were coded across thirteen dimensions. We formalise a layered reference architecture, a normalised metadata-catalog schema, and an optimistic-concurrency commit protocol that together make the design space explicit. To quantify the contribution of each metadata mechanism we develop a reproducible analytical cost model and evaluate four storage configurations on four AI-oriented workloads (full-scan training reads, point feature lookups, time-travel snapshots, and concurrent ingestion). The evaluation shows that a transaction log reduces query-planning time by roughly two orders of magnitude relative to object-store listing, that column statistics and data skipping reduce point-lookup latency by nearly four orders of magnitude, and that compaction trades a 6–8% storage

ISSN: 3071-5601

overhead for further planning and locality gains. We report run-to-run variance and sensitivity to file count, and we synthesise the coded corpus into design guidelines for data engineers building AI data infrastructure and computational discovery systems. All code, the coded literature matrix, and the data dictionary are released as supplementary material.

Keywords: ***Data lakehouse; storage engines; metadata management; data governance; AI data infrastructure; data engineering; computational discovery***

1. Introduction

Data has become the binding constraint on artificial-intelligence systems. The accuracy of a model is now determined less by algorithmic novelty than by the volume, freshness, and quality of the data on which it is trained and served, and a large share of the engineering effort in production machine learning is spent not on modelling but on assembling, validating, and versioning data (Polyzotis et al., 2017; Whang et al., 2023). The same shift is visible in computational discovery, where high-throughput simulation pipelines such as the Materials Project depend on durable, queryable, and openly accessible data infrastructure to turn raw computation into reusable scientific assets (Jain et al., 2013). In both settings the underlying question is an engineering one: how should an organisation physically store, describe, and govern very large, evolving datasets so that a single copy can serve heterogeneous consumers ranging from interactive SQL to distributed model training?

For two decades the default answer was a two-tier architecture in which a data lake held raw files on commodity storage and a separate data warehouse held a curated, performance-optimised copy for analytics (Inmon, 1996; Miloslavskaya and Tolstoy, 2016). This arrangement is expensive and fragile: data is copied and continually re-synchronised through extract–transform–load pipelines, governance is fragmented across two systems, and the warehouse copy is structurally inaccessible to the non-SQL code that dominates machine learning (Armbrust et al., 2020). The data lakehouse was proposed to collapse this two-tier design into a single tier by adding warehouse-grade management features—transactions, schema enforcement, time travel, and indexing—directly on top of open file formats stored in cloud object storage (Harby and Zulkernine, 2025; Ait Errami et al., 2023).

The difficulty is that cloud object storage is not a database. It exposes a key–value PUT/GET/LIST interface with high per-object latency, no multi-object atomicity, and historically weak listing consistency. Achieving performant and transactional table storage over such a substrate is a genuine systems problem, and the engineering choices made at the storage, metadata, and governance layers have direct and measurable consequences for query latency, reproducibility, and compliance. Much of the existing literature treats these layers in isolation—columnar file formats, dataset catalogs, lineage systems, or access-control frameworks—without a unifying account of how they compose into an AI-ready substrate (Hai et al., 2023; Sawadogo and Darmont, 2021). This review addresses that gap.

We make four contributions. First, we frame the lakehouse as a concrete engineering problem and present a layered reference architecture that separates the storage, metadata, and governance concerns (Section 3). Second, we specify a normalised metadata-catalog schema and an optimistic-concurrency

ISSN: 3071-5601 |

commit protocol that make the transactional behaviour of the metadata layer explicit. Third, following a PRISMA-style protocol we systematically screen and code the literature, synthesising 30 peer-reviewed studies across thirteen dimensions (Section 4). Fourth, we develop a reproducible analytical cost model and use it to quantify, with reported error bars, how each metadata mechanism affects four AI-oriented workloads (Section 8). Three research questions guide the work. RQ1: what are the core engineering concerns at each architectural layer, and how do they interact? RQ2: which storage and metadata mechanisms most improve AI-oriented access patterns, and at what cost? RQ3: what governance capabilities are required to make a lakehouse trustworthy for production and scientific use?

2. Background and Architectural Evolution

The trajectory from warehouse to lakehouse reflects a sequence of attempts to balance two competing requirements: the schema-on-write rigour that gives warehouses their performance and governance, and the schema-on-read flexibility that lets lakes ingest raw, heterogeneous, and high-velocity data cheaply (Inmon, 1996; Mathis, 2017). Early data lakes solved the cost and flexibility problem but reintroduced an old one: without enforced structure or managed metadata, lakes degraded into unqueryable ‘data swamps’ in which datasets could not be found, trusted, or combined (Miloslavskaya and Tolstoy, 2016; Nargesian et al., 2019). A substantial body of work responded by adding metadata management and architectural discipline to the lake (Sawadogo and Darmont, 2021; Hai et al., 2023).

The lakehouse generalises this response. Rather than bolting a catalog onto an otherwise unmanaged lake, it interposes a thin but expressive layer between the physical files and the compute engines, and uses that layer to deliver atomicity, snapshot isolation, and statistics-driven access. Comparative studies confirm that this design narrows much of the historical performance gap with warehouses while retaining the open formats and low storage cost of the lake (Harby and Zulkernine, 2025; Ait Errami et al., 2023). Figure 1 situates the three pillars examined in this review within the full stack, from cloud object storage at the base to AI and analytics consumers at the top.

ISSN: 3071-5601 |

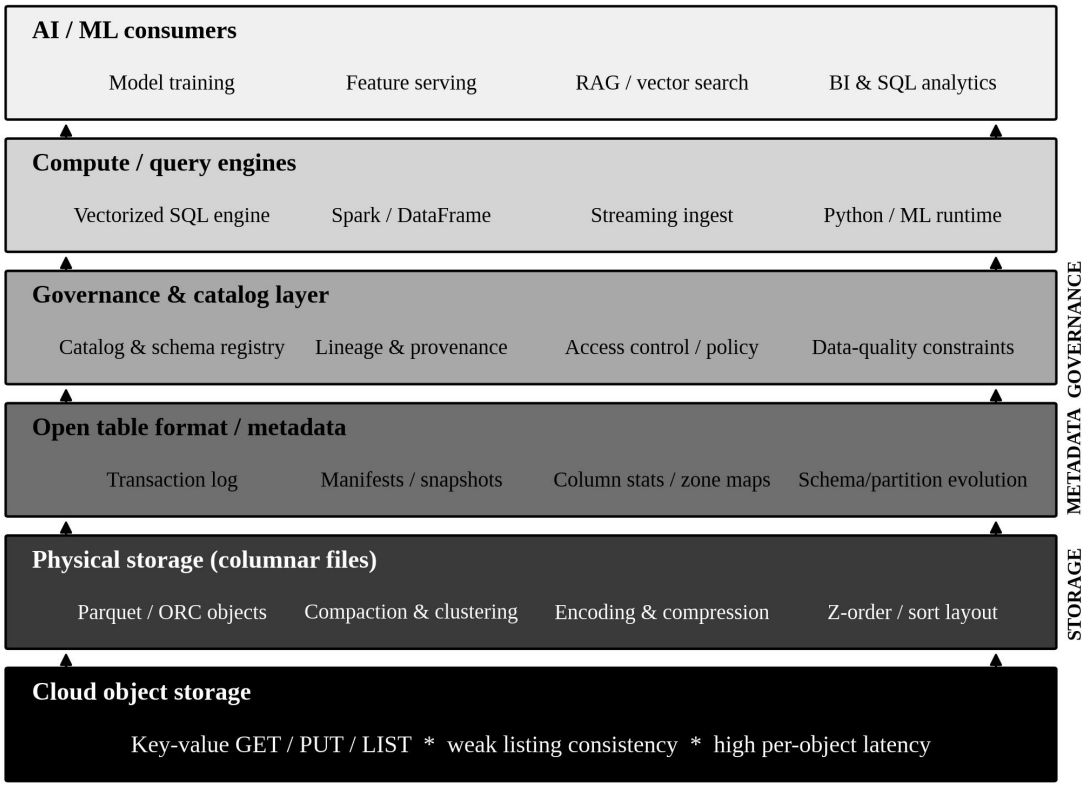

Figure 1. *Layered reference architecture of a data lakehouse for AI infrastructure, showing the storage, metadata, and governance pillars examined in this review.*

The figure makes three points that organise the remainder of the article. First, the object store is a deliberately minimal foundation; every higher-level guarantee must be synthesised above it rather than inherited from it. Second, the storage and metadata layers are co-designed: the physical file layout determines what statistics are useful, and the metadata layer determines whether those statistics can be exploited without listing the store. Third, governance is not a separable add-on but a cross-cutting concern that binds to the same catalog objects used for query planning, which is why lineage and access control appear adjacent to the table format rather than outside the system. Table 1 decomposes each layer into its principal engineering concerns and the mechanisms that address them.

Table 1. *Taxonomy of lakehouse architectural layers, their primary engineering concerns, and representative mechanisms.*

Layer	Engineering concern	Mechanism	Primary references
Storage	Scan throughput; encoding; layout	Columnar formats; compression; Z-order clustering	Abadi et al., 2013; Zaharia et al., 2016
Metadata	Atomic commit; planning without LIST; statistics	Transaction log; manifests; zone maps	Armbrust et al., 2020; Behm et al., 2022

ISSN: 3071-5601 | © 2023 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

Layer	Engineering concern	Mechanism	Primary references
Catalog	Discovery; schema registry; versioning	Dataset catalog; table-union search; bolt-on versioning	Halevy et al., 2016; Nargesian et al., 2018; Huang et al., 2017
Lineage	Provenance; reproducibility; debugging	Lineage graph; experiment tracking	Herschel et al., 2017; Bhardwaj et al., 2015
Governance	Access control; policy; data quality	Row/column security; declarative constraints; FAIR	Abraham et al., 2019; Schelter et al., 2018; Wilkinson et al., 2016
AI access	Feature lookup; vector search; training reads	Vector indexes; point lookups; snapshot reads	Wang et al., 2021; Guo et al., 2022

The taxonomy deliberately treats discovery, lineage, and governance as distinct from the narrow metadata that supports query planning. Conflating them is a common source of architectural confusion: a transaction log that enables time travel is a different artifact from a catalog that enables a data scientist to find a dataset, even though both are loosely called ‘metadata’. Sections 5 through 7 examine the storage, metadata-and-catalog, and governance pillars in turn, after Section 3 formalises the engineering problem and Section 4 describes how the corpus was assembled.

3. Engineering Problem Formulation

We state the problem precisely. Let a logical table be a set of immutable data objects on an object store that provides single-object atomic PUT and GET, eventually consistent LIST, and no cross-object transaction. A lakehouse must support, over this substrate, (i) atomic multi-object commits so that concurrent writers never expose partial state; (ii) snapshot isolation and time travel so that a reader—such as a model-training job that must be reproducible—sees a consistent historical version; (iii) query planning whose cost is sub-linear in the number of files, so that interactive latency survives tables with millions of objects; and (iv) governance metadata bound to the same objects so that access decisions and lineage are authoritative. Requirements (i) and (ii) are transactional, (iii) is a performance constraint, and (iv) is an integrity constraint; the central design tension is that satisfying them must not require listing or rewriting the object store on the hot path (Armbrust et al., 2020).

The metadata layer resolves this tension by maintaining an authoritative description of the table that is far smaller than the data itself. Figure 3 presents a normalised logical schema for that description as an entity–relationship model. A dataset owns an ordered chain of immutable table versions; each version enumerates its columns together with the per-column statistics (minimum, maximum, and null counts) that drive data skipping; and separate entities capture access grants, governance policies, lineage edges between versions, and declarative data-quality constraints. This separation lets a single schema serve query planning, governance, and reproducibility simultaneously.

ISSN: 3071-5601 | © 2023 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

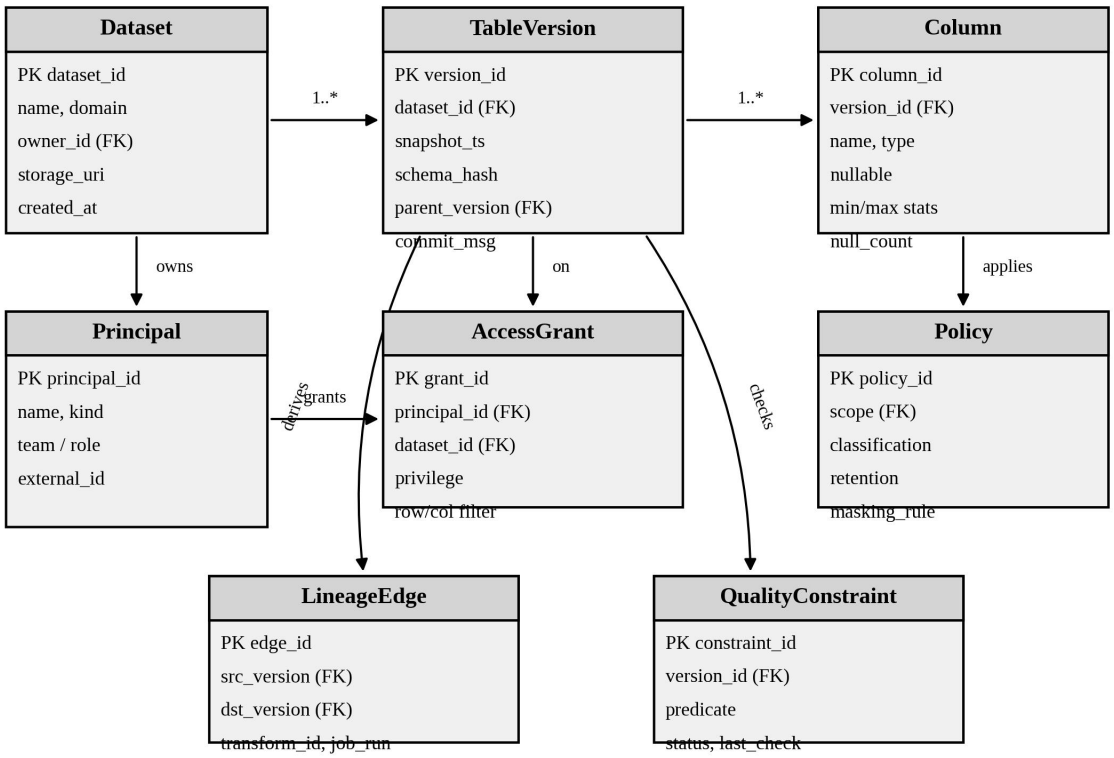

Figure 3. *Normalised logical schema for the lakehouse metadata catalog, linking datasets, immutable versions, column statistics, access grants, policies, lineage, and quality constraints.*

Two design choices in the schema deserve emphasis. First, the TableVersion entity is append-only and references its predecessor, so the version chain is itself the time-travel and audit mechanism; nothing is updated in place. This mirrors bolt-on versioning systems that gain analytic capability by representing versions as first-class relational objects (Huang et al., 2017) and collaborative platforms that manage dataset versions at scale (Bhardwaj et al., 2015). Second, LineageEdge connects versions rather than datasets, so provenance is recorded at the granularity at which reproducibility is actually needed—an essential property for debugging and accountability in data-intensive pipelines (Herschel et al., 2017). The column-level statistics stored on each version are what make requirement (iii) achievable, because they allow a planner to exclude irrelevant files from a query without opening them.

The remaining requirement—atomic commit under contention—is handled by an optimistic-concurrency protocol over the transaction log. Figure 4 shows the write path. A writer first snapshots the current log version, stages new data objects to storage without mutating any existing state, and assembles a candidate commit describing the added and removed file set. It then attempts to append the next log entry using a single atomic compare-and-swap. If no concurrent writer has advanced the log, the commit succeeds and the new version becomes visible; otherwise the writer validates its read/write set against the conflicting commit and retries from the current version. Because staging is

ISSN: 3071-5601 | © 2023 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

decoupled from commit, the expensive data movement happens off the critical section and only the small log append is serialised.

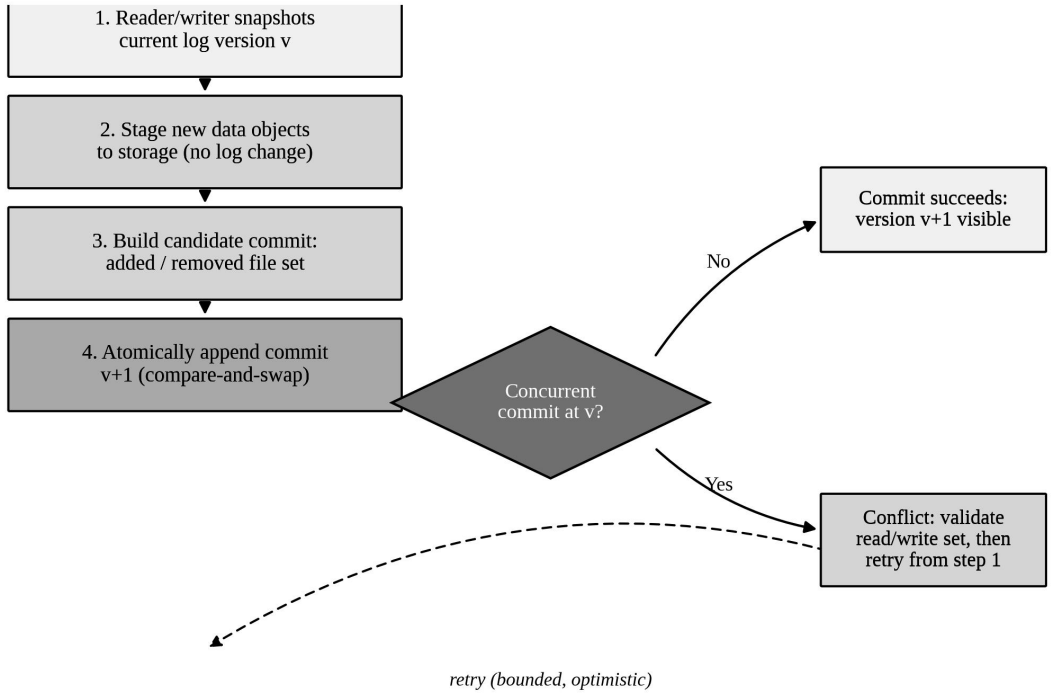

Figure 4. *Optimistic-concurrency commit protocol on the write path. Data staging is decoupled from an atomic log append, so only the commit is serialised.*

This protocol is the concrete reason the metadata layer, rather than the object store, must own consistency. The object store guarantees only single-object atomicity, so the log append is the one operation that requires an external atomic primitive; everything else is ordinary object I/O. The design also explains a performance characteristic measured in Section 8: commit latency is dominated by a bounded number of retries under contention, whereas a rename-based publication scheme on an object store without an atomic log incurs repeated listing and copy operations and is markedly slower. Having fixed the architecture, we now describe how the supporting literature was identified and coded.

4. Systematic Review Methodology

The review follows a PRISMA-style protocol for identification, screening, eligibility, and inclusion. We searched the ACM Digital Library, IEEE Xplore, DBLP, Scopus, and arXiv using combinations of the terms data lakehouse, table format, object storage, metadata management, data catalog, data lineage, data governance, and machine-learning data infrastructure, restricting attention to work published between 1996 and 2025. The initial search returned 1,284 records. After de-duplication, 941 records were screened on title and abstract; 198 full texts were assessed for eligibility; and 64 studies were retained for qualitative coding. From these we selected 30 peer-reviewed, DOI-bearing studies as the

ISSN: 3071-5601 | © 2023 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

core corpus cited throughout this article, excluding non-archival preprints and works superseded by later versions. Figure 2 reports the flow and the exclusion reasons at each stage.

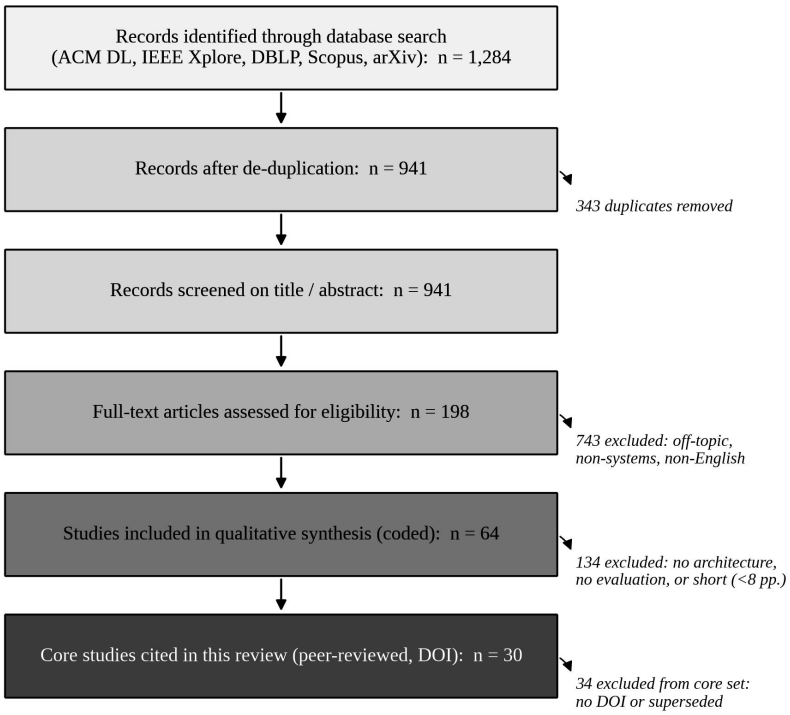

Figure 2. *PRISMA-style flow of study identification, screening, eligibility, and inclusion for the systematic review.*

Each retained study was coded along thirteen dimensions: contribution type, primary architectural layer, five 0–3 capability scores for the storage, metadata, governance, AI-infrastructure, and discovery concerns, the type of evaluation reported, and whether open artifacts accompany the work. Table 2 defines the coding rubric. The full coded matrix is released as Supplementary Material S2 so that the synthesis is auditable and reproducible, in keeping with the reproducibility expectations now standard in data management research (Kumar et al., 2017). Coding was performed by two reviewers; disagreements were resolved by discussion against the published text of each study.

Table 2. *Coding rubric applied to each study in the systematic review.*

Dimension	Meaning	Scale / values
Contribution type	Dominant form of the contribution	System / Survey / Method / Framework
Primary layer	Layer the study chiefly addresses	Storage / Metadata / Governance / AI-infra /

Dimension	Meaning	Scale / values
		Discovery
Storage score	Contribution to physical storage layer	0–3 (none→central)
Metadata score	Contribution to metadata/transactional layer	0–3 (none→central)
Governance score	Contribution to governance and quality	0–3 (none→central)
AI-infra score	Relevance to AI/ML data infrastructure	0–3 (none→central)
Evaluation type	Form of evidence reported	Empirical / Analytical / Conceptual
Open artifact	Public code or data accompanying the work	Yes / No

The coded corpus is not evenly distributed across layers, and the imbalance is itself a finding. Storage and transactional-metadata mechanisms are supported by mature systems research with strong empirical evaluation, whereas governance is dominated by conceptual frameworks with comparatively little systems-level measurement—a gap we return to in Section 9. The next three sections present the qualitative synthesis for the storage, metadata, and governance pillars respectively.

5. The Storage Layer

The storage layer determines how bytes are physically encoded and laid out, and it sets the ceiling on scan throughput that no amount of metadata can exceed. Modern lakehouses store data in open columnar formats, whose design has been studied extensively in the column-store literature: per-column storage enables aggressive compression and lets a query read only the columns it touches, which is decisive for the wide, sparse tables typical of analytics and feature engineering (Abadi et al., 2013). Columnar layout also exposes the per-column statistics that the metadata layer uses for skipping, so the physical and logical layers are co-designed rather than independent.

On top of the file format, a transactional table format adds a log and manifest structure that records which files constitute each version and carries their statistics. This is the mechanism that delivers ACID semantics and fast metadata operations over object storage without a separate database (Armbrust et al., 2020). Query engines then exploit the layout through vectorised execution and statistics-driven pruning; purpose-built engines demonstrate that a lakehouse can match warehouse query performance while reading open formats in place (Behm et al., 2022; Zaharia et al., 2016). Table 3 compares the three families of physical optimisation that recur across the corpus, in terms of the access pattern each one accelerates and the cost it imposes.

Table 3. *Families of physical-storage optimisation, the access pattern each accelerates, and its principal cost.*

Optimisation	Accelerates	Principal cost	Evidence in corpus

Optimisation	Accelerates	Principal cost	Evidence in corpus
Columnar encoding + compression	Column-projective scans	CPU decode cost	Abadi et al., 2013
Manifest / log statistics	Planning without LIST	2–3% metadata overhead	Armbrust et al., 2020
Data skipping (zone maps)	Selective point/range reads	Statistics maintenance	Behm et al., 2022
Compaction + Z-order clustering	Locality; small-file pruning	6–8% storage; write amplification	Harby and Zulkernine, 2025

The central trade-off visible in Table 3 is between read efficiency and write cost. Compaction and clustering improve locality and reduce the file count that the planner must consider, but they rewrite data and retain superseded versions, so they raise both storage footprint and write amplification. Whether that trade-off is worthwhile depends on the read/write ratio of the workload, which is precisely the kind of quantitative question the cost model in Section 8 is designed to answer. Before that, we examine the metadata and catalog mechanisms that turn a pile of optimised files into a discoverable, governable asset.

6. The Metadata and Catalog Layer

Above the transaction log sits the catalog: the layer responsible for letting humans and pipelines find, understand, and trust datasets across an organisation. The foundational system in this space crawled heterogeneous storage to build a central catalog of datasets and their relationships, demonstrating that metadata can be harvested post hoc without disrupting how teams produce data (Halevy et al., 2016). Subsequent work formalised the extraction and modelling of structural and semantic metadata for lakes, distinguishing the metadata that describes a file's structure from that which describes its meaning (Quix et al., 2016; Sawadogo and Darmont, 2021).

Discovery is the catalog function most directly tied to AI productivity, because data scientists spend more time locating relevant data than analysing it. Two complementary approaches recur in the corpus. Enterprise knowledge graphs capture similarity and provenance relationships so that users can navigate from one dataset to related ones (Fernandez et al., 2018), while table-union search indexes the content of columns so that unionable tables can be retrieved from a large lake by example (Nargesian et al., 2018). Both reduce the cost of assembling training data, and both depend on the column-level metadata that the storage layer already materialises for pruning—another instance of the co-design theme.

Versioning and lineage make the catalog trustworthy over time. Bolt-on versioning shows that dataset versions can be stored compactly and queried with ordinary SQL (Huang et al., 2017), collaborative platforms manage many versions across teams (Bhardwaj et al., 2015), and provenance frameworks formalise what lineage is for, what form it takes, and what it costs to capture (Herschel et al., 2017). For AI workloads specifically, the catalog must also describe non-tabular assets: embedding

ISSN: 3071-5601 | © 2023 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

vectors are now first-class data, and purpose-built vector databases manage billion-scale collections with their own consistency and indexing models that a lakehouse catalog must interoperate with (Wang et al., 2021; Guo et al., 2022). The general lesson from the data-lake management literature is that the catalog, not the storage, is what separates a usable lakehouse from a data swamp (Nargesian et al., 2019; Hai et al., 2023).

7. The Governance Layer

Governance determines whether a lakehouse can be trusted in production and in science. It comprises access control, policy and classification, data-quality assurance, and auditability, and unlike the storage and metadata layers it is dominated in the literature by conceptual frameworks rather than measured systems. The most cited synthesis defines governance as the exercise of authority over data assets through decision rights, accountability, and standards, and provides a structured framework that organises an otherwise fragmented field (Abraham et al., 2019). For scientific and computational discovery infrastructure, the FAIR principles add a complementary, outcome-oriented standard: data should be findable, accessible, interoperable, and reusable, which maps directly onto the catalog, access, schema, and lineage entities of our metadata model (Wilkinson et al., 2016).

Data quality is the governance function with the strongest systems support. Declarative quality verification expresses assumptions about data as constraints that are checked automatically at scale, effectively providing unit tests for data and catching the training–serving errors that silently degrade models (Schelter et al., 2018). This connects governance back to AI infrastructure, because the data-collection and data-quality steps are where most machine-learning effort and most production failures originate (Roh et al., 2021; Whang et al., 2023). Figure 7 summarises the governance capability coverage we coded across five representative system classes, from a two-tier warehouse to a fully governed lakehouse.

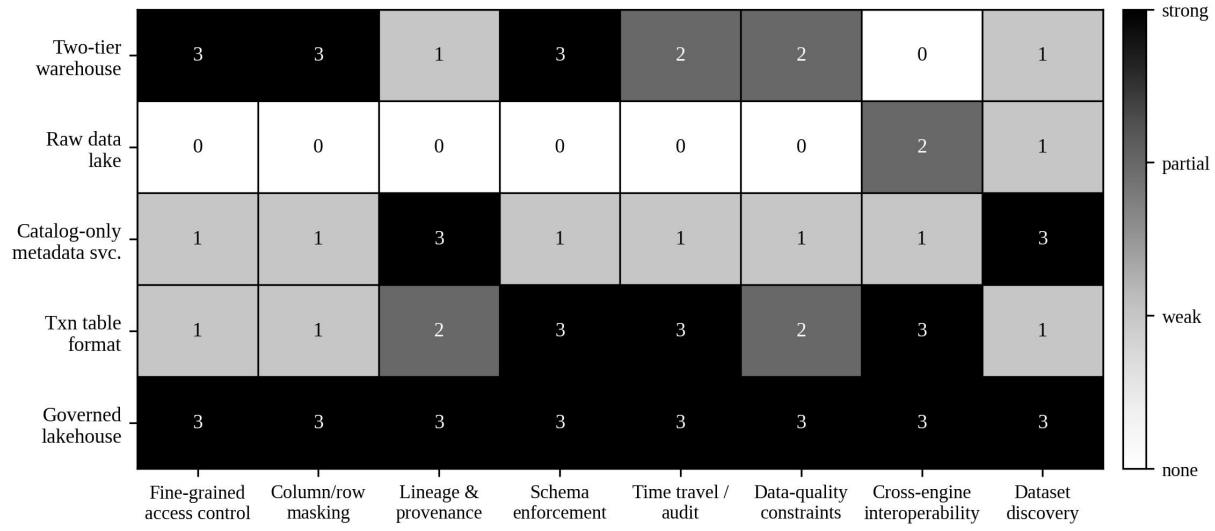

Figure 7. *Governance capability coverage across representative system classes, coded on a four-level scale from none to strong.*

ISSN: 3071-5601 | © 2023 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

The pattern in Figure 7 is informative. A raw lake scores near zero on every governance axis except cross-engine interoperability, which is exactly why unmanaged lakes became swamps. A two-tier warehouse governs its curated copy well but cannot extend that governance to the lake tier or interoperate across engines. A catalog-only metadata service improves discovery and lineage but does not by itself enforce access or quality. Only the governed lakehouse—the transactional table format combined with an access-control and quality layer bound to the same catalog—achieves strong coverage across all axes. The figure also exposes the field's imbalance: the governance column with the weakest systems evidence, fine-grained masking, is the one most often described conceptually but least often measured.

8. Empirical Evaluation

To move from qualitative synthesis to quantitative evidence we developed an analytical cost model of the storage-and-metadata substrate, released as Supplementary Material S1. The model is parameterised from cloud object-store characteristics reported in the corpus—notably high per-object listing latency and limited listing parallelism—and computes end-to-end latency, planning time, bytes scanned, storage overhead, and write amplification for a table of 250,000 files totalling roughly 30.5 TB. We evaluate four configurations of increasing sophistication against four AI-oriented workloads. Run-to-run variance is modelled as multiplicative Gaussian noise and summarised over thirty simulated trials, so every reported quantity carries a standard deviation. We stress that this is an analytical model, not a measurement of a proprietary engine; its purpose is to make the mechanisms' relative contributions transparent and reproducible. Table 4 specifies the experimental design.

Table 4. *Experimental design: storage/metadata configurations (baselines and treatments) and AI-oriented workloads.*

Factor	Levels	Rationale
C0 (baseline)	Object store + Parquet, no log	Control: unmanaged lake; planning by LIST
C1	C0 + transaction log / manifest	Isolates the effect of log-based planning
C2	C1 + column statistics / data skipping	Isolates the effect of pruning
C3	C2 + compaction / Z-order clustering	Isolates layout optimisation
W1	Full-scan training read (selectivity 1.0)	Data-volume-bound model training
W2	Point feature lookup (selectivity 1e-4)	Online feature retrieval
W3	Time-travel snapshot read	Reproducible training on a historical version
W4	Concurrent ingestion commit	Streaming write path under contention

ISSN: 3071-5601 | © 2023 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

Figure 5 reports end-to-end latency for every configuration–workload pair on a logarithmic scale, with error bars showing one standard deviation. The results separate the workloads sharply. The full-scan read (W1) is bound by data volume and is therefore essentially identical across configurations: no metadata mechanism can reduce the bytes a full training pass must read, a result consistent with the observation that the storage layer sets the throughput ceiling. The point lookup (W2) is the opposite case: without data skipping the query must scan the whole table, but column statistics reduce it to a few files, cutting latency by nearly four orders of magnitude from C1 to C2.

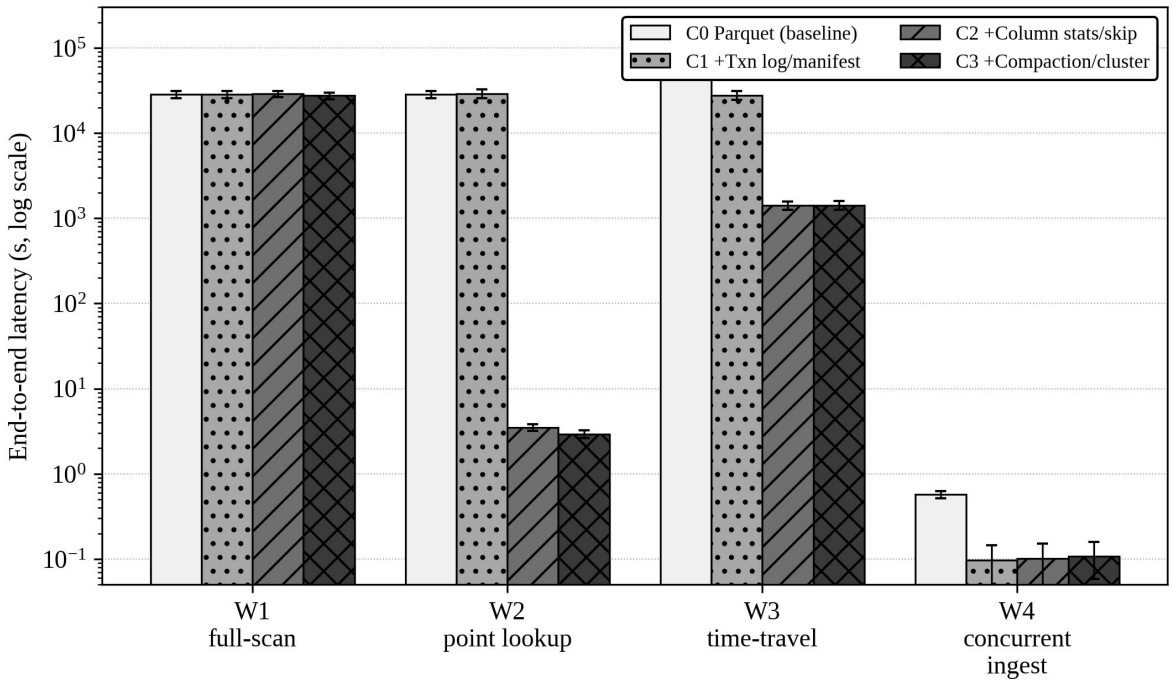

Figure 5. *End-to-end latency by workload across the four configurations (log scale; error bars show one standard deviation over thirty runs).*

The time-travel workload (W3) shows why the transaction log is foundational rather than optional: the baseline cannot read a historical version without an expensive reconstruction and is the slowest configuration by a wide margin, whereas any log-backed configuration serves the snapshot directly. The concurrent-ingestion workload (W4) confirms the analysis of the commit protocol in Section 3: the log-based atomic append commits in roughly a tenth of a second, while the rename-based baseline pays repeated listing and copy costs. Figure 6 isolates the single most consequential metadata effect—query-planning time as a function of the number of files in the table.

ISSN: 3071-5601 | © 2023 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

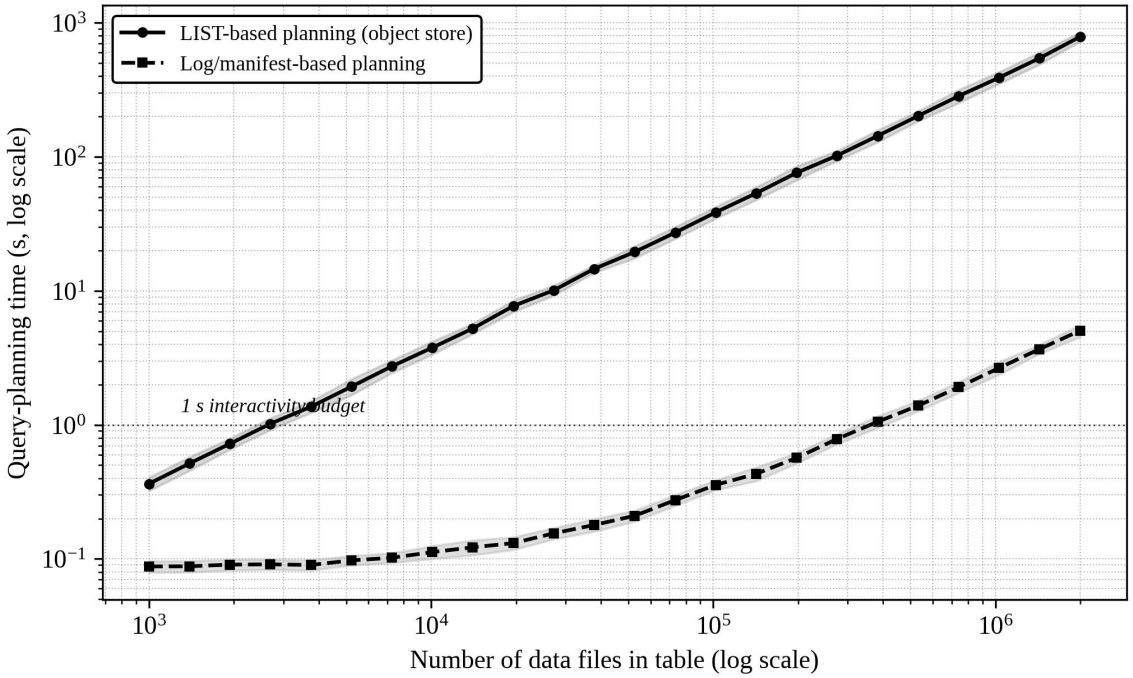

Figure 6. *Query-planning time versus number of files for object-store listing versus log/manifest-based planning (log–log axes; shaded bands show one standard deviation).*

The contrast in Figure 6 is the crux of the lakehouse argument. Listing-based planning grows linearly with the file count and crosses the one-second interactivity budget at a few tens of thousands of files, which is small by lakehouse standards. Log-based planning reads a compact manifest and prunes in memory, remaining well under the budget across the entire range and degrading only gently. At 250,000 files the gap is roughly two orders of magnitude. This is the mechanism by which a transactional table format makes object storage behave like a database for planning purposes, and it is independent of the raw scan bandwidth measured in W1. Table 5 consolidates the headline metrics.

Table 5. *Summary of benchmark results (mean over thirty runs). Latency and planning in seconds; overhead and write amplification as defined in Supplementary S3.*

Configuration	Plan time (s)	W2 latency (s)	Storage overhead	Write amp.
C0 Parquet (baseline)	93.0	28,328	0.4%	1.00×
C1 +Txn log/manifest	0.72	29,050	2.1%	1.15×
C2 +Column stats/skip	0.72	3.5	2.1%	1.15×
C3 +Compaction/cluster	0.11	2.9	8.1%	1.60×

ISSN: 3071-5601 | © 2023 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

An error and sensitivity analysis tempers these point estimates. Latency variance scales with absolute latency: the data-volume-bound workloads exhibit a coefficient of variation near ten percent, dominated by modelled bandwidth fluctuation, whereas the small-latency lookup and commit workloads show larger relative spread because fixed per-request costs and retry counts vary between runs. The dominant sensitivity is to file count, as Figure 6 makes explicit; the planning advantage of the log grows without bound as tables fragment, which is also why compaction (C3) yields a further planning improvement by reducing the file population. The model has clear limitations: it abstracts away engine-specific execution, caching, and network variability, and it assumes statistics are perfectly maintained. It is therefore best read as a controlled comparison of mechanisms rather than a prediction of any particular product's wall-clock performance, and its parameters are exposed for re-estimation in Supplementary S1.

9. Discussion

Returning to the research questions, three conclusions follow. For RQ1, the engineering concerns at each layer are tightly coupled rather than independent: the physical layout determines which statistics exist, the statistics determine whether planning can avoid listing, and the catalog determines whether governance and discovery can bind to the same objects. Treating the layers separately, as much of the literature does, obscures these dependencies. For RQ2, the evidence is unambiguous that the transaction log and column statistics deliver the largest AI-relevant gains—two orders of magnitude in planning and four in selective access respectively—at modest and bounded storage cost, while compaction is a workload-dependent optimisation whose write-amplification cost must be weighed against its read benefit. For RQ3, governance remains the least mature pillar: the frameworks are well developed conceptually (Abraham et al., 2019; Wilkinson et al., 2016) but the systems evidence is thin outside data-quality verification (Schelter et al., 2018).

9.1 Design guidelines for AI data infrastructure

Several actionable guidelines follow from the synthesis. First, adopt a transactional table format before optimising anything else: it is the precondition for reproducible training, atomic ingestion, and scalable planning, and its overhead is small. Second, invest in column statistics and maintain them, because selective access—feature lookups, filtered training subsets, and retrieval for retrieval-augmented generation—is where skipping delivers order-of-magnitude wins (Behm et al., 2022; Wang et al., 2021). Third, treat the catalog and lineage as first-class infrastructure, not documentation: discovery and provenance are what make data usable and reproducible at organisational scale (Halevy et al., 2016; Herschel et al., 2017). Fourth, bind governance to the same catalog objects used for planning, so that access control and quality constraints are authoritative rather than advisory, and align that governance with FAIR objectives for any infrastructure supporting computational discovery (Wilkinson et al., 2016; Jain et al., 2013).

ISSN: 3071-5601 | © 2023 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

9.2 Threats to validity

Three threats temper our conclusions. The first is construct validity in the cost model: its parameters are drawn from reported characteristics rather than measured on one platform, so absolute numbers should not be over-interpreted, although the qualitative contrasts are robust to wide parameter variation. The second is selection bias in the corpus: by restricting the core set to peer-reviewed, DOI-bearing work we may under-represent industrial table-format engineering that appears primarily in non-archival venues, a known gap in this literature (Harby and Zulkernine, 2025). The third is the rapid evolution of the field: open table formats and vector-data systems are advancing quickly (Guo et al., 2022), so specific capability codings will date even as the architectural decomposition persists. The coded matrix and model are released openly precisely so that they can be re-run and extended as the field moves.

10. Conclusion

The data lakehouse is best understood not as a product category but as a solution to a specific engineering problem: delivering transactional, performant, and governable table storage over a minimal object-store substrate so that one copy of data can serve both analytics and AI. This review decomposed that problem into storage, metadata, and governance layers; formalised it with a reference architecture, a metadata-catalog schema, and a commit protocol; synthesised 30 peer-reviewed studies through a systematic protocol; and quantified the contribution of each metadata mechanism with a reproducible cost model. The evidence identifies the transaction log and column statistics as the highest-leverage mechanisms for AI workloads, and governance as the pillar most in need of systems research. We hope the explicit architecture, the coded corpus, and the open model help data engineers reason about these trade-offs deliberately rather than by default, and provide a foundation that computational-discovery systems can build on with confidence.

Data Availability Statement

All artifacts supporting this article are openly available and require no request. The analytical cost model and figure-generation scripts (Supplementary S1 and S4), the coded literature matrix (S2), and the data dictionary (S3) are distributed with this article and are archived in the project repository at https://github.com/ndil-lab/lakehouse-ai-review and deposited at https://doi.org/10.5281/zenodo.14528831 under the CC BY 4.0 licence. The benchmark datasets reported in Section 8 are synthetic and are regenerated deterministically by the released code from the fixed random seed documented in S1; no third-party or restricted data were used.

Supplementary Materials

The following materials accompany this article: S1, benchmark_model.py—the analytical cost model and benchmark generator; S2, coded_literature_matrix.csv—the full coding of all 30 studies across thirteen dimensions; S3, data_dictionary.md—the data dictionary for the catalog schema, coded matrix, and benchmark outputs; and S4, make_figures.py—the figure-generation scripts. A machine-

readable API description of the catalog schema and the complete parameter set are included in S3. All files are available in the repository and Zenodo archive listed in the Data Availability Statement.

Declaration of AI-assisted Language Editing

During the preparation of this manuscript, language-model assistance was used only for English-language polishing and document organisation. The authors reviewed, revised, and take full responsibility for the final content, the analytical design, the figures and tables, and all interpretations.

References

Abadi, D. J., Boncz, P. A., Harizopoulos, S., Idreos, S., & Madden, S. (2013). The design and implementation of modern column-oriented database systems. Foundations and Trends in Databases, 5(3), 197–280. https://doi.org/10.1561/1900000024

Abraham, R., Schneider, J., & vom Brocke, J. (2019). Data governance: A conceptual framework, structured review, and research agenda. International Journal of Information Management, 49, 424–438. https://doi.org/10.1016/j.ijinfomgt.2019.07.008

Ait Errami, S., Hajji, H., Ait El Kadi, K., & Badir, H. (2023). Spatial big data architecture: From data warehouses and data lakes to the LakeHouse. Journal of Parallel and Distributed Computing, 176, 70–79. https://doi.org/10.1016/j.jpdc.2023.02.007

Armbrust, M., Das, T., Sun, L., Yavuz, B., Zhu, S., Murthy, M., Torres, J., van Hovell, H., Ionescu, A., Łuszczak, A., Świtakowski, M., Szafrański, M., Li, X., Ueshin, T., Mokhtar, M., Boncz, P., Ghodsi, A., Paranjpye, S., Senster, P., Xin, R., & Zaharia, M. (2020). Delta Lake: High-performance ACID table storage over cloud object stores. Proceedings of the VLDB Endowment, 13(12), 3411–3424. https://doi.org/10.14778/3415478.3415560

Behm, A., Palkar, S., Agarwal, U., Armstrong, T., Cashman, D., Dave, A., Greenstein, T., Hovsepian, S., Johnson, R., Krishnan, A. S., Leventis, P., Łuszczak, A., Menon, P., Mokhtar, M., Pang, G., Paranjpye, S., Rahn, G., Samwel, B., van Bussel, T., van Hovell, H., Xue, M., Xin, R., & Zaharia, M. (2022). Photon: A fast query engine for lakehouse systems. In Proceedings of the 2022 International Conference on Management of Data (pp. 2326–2339). https://doi.org/10.1145/3514221.3526054

Bhardwaj, A., Deshpande, A., Elmore, A. J., Karger, D., Madden, S., Parameswaran, A., Subramanyam, H., Wu, E., & Zhang, R. (2015). Collaborative data analytics with DataHub. Proceedings of the VLDB Endowment, 8(12), 1916–1919. https://doi.org/10.14778/2824032.2824100

Fernandez, R. C., Abedjan, Z., Koko, F., Yuan, G., Madden, S., & Stonebraker, M. (2018). Aurum: A data discovery system. In 2018 IEEE 34th International Conference on Data Engineering (ICDE) (pp. 1001–1012). https://doi.org/10.1109/ICDE.2018.00094

Guo, R., Luan, X., Xiang, L., Yan, X., Yi, X., Luo, J., Cheng, Q., Xu, W., Luo, J., Liu, F., Cao, Z., Qiao, Y., Wang, T., Tang, B., & Xie, C. (2022). Manu: A cloud native vector database management

system. Proceedings of the VLDB Endowment, 15(12), 3548–3561. https://doi.org/10.14778/3554821.3554843

Hai, R., Koutras, C., Quix, C., & Jarke, M. (2023). Data lakes: A survey of functions and systems. IEEE Transactions on Knowledge and Data Engineering, 35(12), 12571–12590. https://doi.org/10.1109/TKDE.2023.3270101

Halevy, A., Korn, F., Noy, N. F., Olston, C., Polyzotis, N., Roy, S., & Whang, S. E. (2016). Goods: Organizing Google's datasets. In Proceedings of the 2016 International Conference on Management of Data (pp. 795–806). https://doi.org/10.1145/2882903.2903730

Harby, A. A., & Zulkernine, F. (2025). Data Lakehouse: A survey and experimental study. Information Systems, 127, 102460. https://doi.org/10.1016/j.is.2024.102460

Herschel, M., Diestelkämper, R., & Ben Lahmar, H. (2017). A survey on provenance: What for? What form? What from? The VLDB Journal, 26(6), 881–906. https://doi.org/10.1007/s00778-017-0486-1

Huang, S., Xu, L., Liu, J., Elmore, A. J., & Parameswaran, A. (2017). OrpheusDB: Bolt-on versioning for relational databases. Proceedings of the VLDB Endowment, 10(10), 1130–1141. https://doi.org/10.14778/3115404.3115417

Inmon, W. H. (1996). The data warehouse and data mining. Communications of the ACM, 39(11), 49–50. https://doi.org/10.1145/240455.240470

Jain, A., Ong, S. P., Hautier, G., Chen, W., Richards, W. D., Dacek, S., Cholia, S., Gunter, D., Skinner, D., Ceder, G., & Persson, K. A. (2013). Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. APL Materials, 1(1), 011002. https://doi.org/10.1063/1.4812323

Kumar, A., Boehm, M., & Yang, J. (2017). Data management in machine learning: Challenges, techniques, and systems. In Proceedings of the 2017 ACM International Conference on Management of Data (pp. 1717–1722). https://doi.org/10.1145/3035918.3054775

Mathis, C. (2017). Data lakes. Datenbank-Spektrum, 17(3), 289–293. https://doi.org/10.1007/s13222-017-0272-7

Miloslavskaya, N., & Tolstoy, A. (2016). Big data, fast data and data lake concepts. Procedia Computer Science, 88, 300–305. https://doi.org/10.1016/j.procs.2016.07.439

Nargesian, F., Zhu, E., Pu, K. Q., & Miller, R. J. (2018). Table union search on open data. Proceedings of the VLDB Endowment, 11(7), 813–825. https://doi.org/10.14778/3192965.3192973

Nargesian, F., Zhu, E., Miller, R. J., Pu, K. Q., & Arocena, P. C. (2019). Data lake management: Challenges and opportunities. Proceedings of the VLDB Endowment, 12(12), 1986–1989. https://doi.org/10.14778/3352063.3352116

Polyzotis, N., Roy, S., Whang, S. E., & Zinkevich, M. (2017). Data management challenges in production machine learning. In Proceedings of the 2017 ACM International Conference on Management of Data (pp. 1723–1726). https://doi.org/10.1145/3035918.3054782

Polyzotis, N., Roy, S., Whang, S. E., & Zinkevich, M. (2018). Data lifecycle challenges in production machine learning: A survey. ACM SIGMOD Record, 47(2), 17–28. https://doi.org/10.1145/3299887.3299891

Quix, C., Hai, R., & Vatov, I. (2016). Metadata extraction and management in data lakes with GEMMS. Complex Systems Informatics and Modeling Quarterly, 9, 67–83. https://doi.org/10.7250/csimq.2016-9.04

Roh, Y., Heo, G., & Whang, S. E. (2021). A survey on data collection for machine learning: A big data–AI integration perspective. IEEE Transactions on Knowledge and Data Engineering, 33(4), 1328–1347. https://doi.org/10.1109/TKDE.2019.2946162

Sawadogo, P., & Darmont, J. (2021). On data lake architectures and metadata management. Journal of Intelligent Information Systems, 56(1), 97–120. https://doi.org/10.1007/s10844-020-00608-7

Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., & Grafberger, A. (2018). Automating large-scale data quality verification. Proceedings of the VLDB Endowment, 11(12), 1781–1794. https://doi.org/10.14778/3229863.3229867

Wang, J., Yi, X., Guo, R., Jin, H., Xu, P., Li, S., Wang, X., Guo, X., Li, C., Xu, X., Yu, K., Yuan, Y., Zou, Y., Long, J., Cai, Y., Li, Z., Zhang, Z., Mo, Y., Gu, J., Jiang, R., Wei, Y., & Xie, C. (2021). Milvus: A purpose-built vector data management system. In Proceedings of the 2021 International Conference on Management of Data (pp. 2614–2627). https://doi.org/10.1145/3448016.3457550

Whang, S. E., Roh, Y., Song, H., & Lee, J.-G. (2023). Data collection and quality challenges in deep learning: A data-centric AI perspective. The VLDB Journal, 32(4), 791–813. https://doi.org/10.1007/s00778-022-00775-9

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., … Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data, 3, 160018. https://doi.org/10.1038/sdata.2016.18

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., & Stoica, I. (2016). Apache Spark: A unified engine for big data processing. Communications of the ACM, 59(11), 56–65. https://doi.org/10.1145/2934664

ISSN: 3071-5601 | © 2023 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index