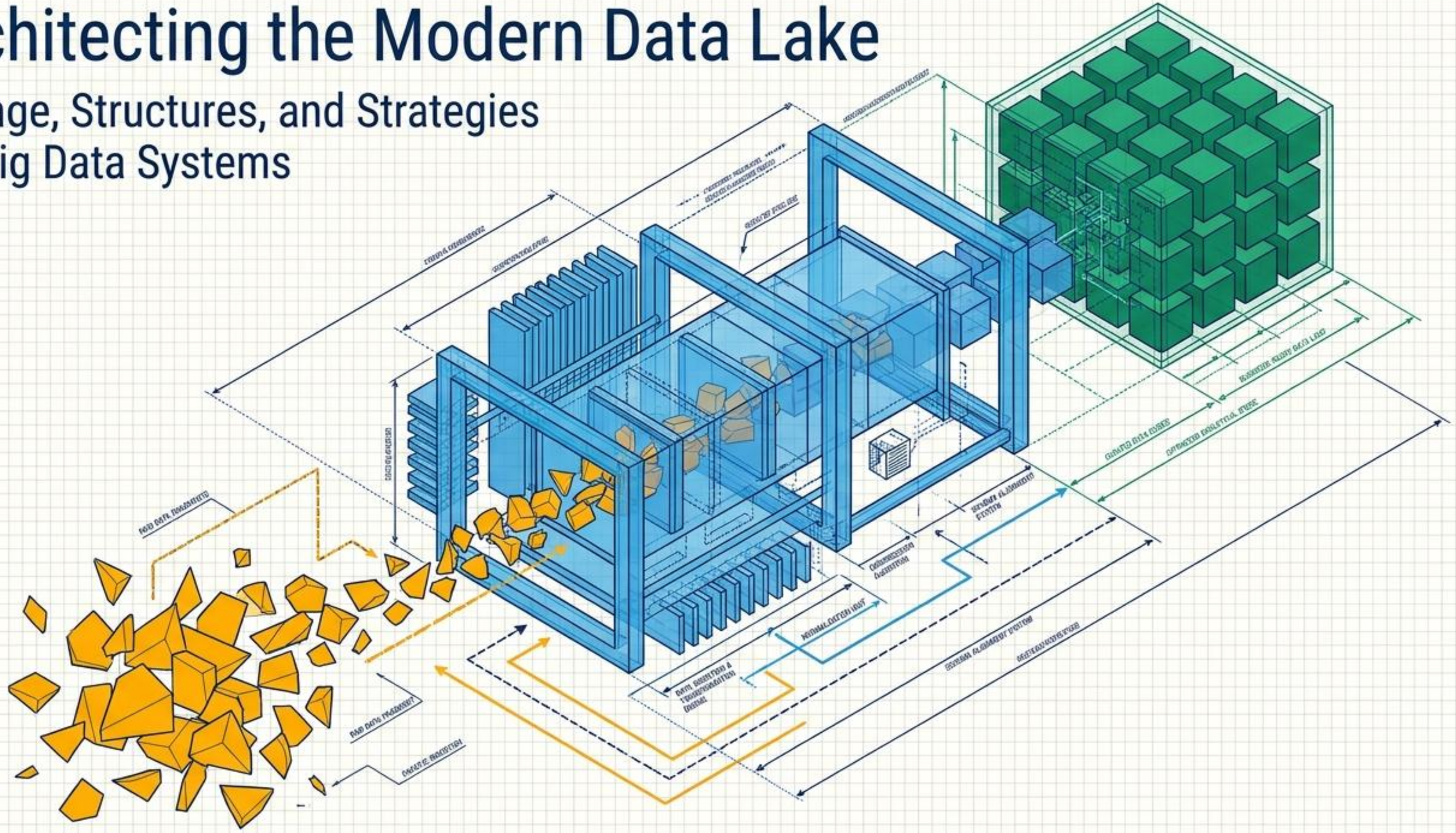
Introduction to Big Data
IN411

Course Contents

- **Chapter 1: Overview of Big Data**
- **Chapter 2: Hadoop Ecosystem**
- **Chapter 3: Apache Spark Basics**
- **Chapter 4: Introduction to NoSQL Databases**
- **Chapter 5: Big Data Storage & Data Lake Concepts**

Architecting the Modern Data Lake

Storage, Structures, and Strategies for Big Data Systems

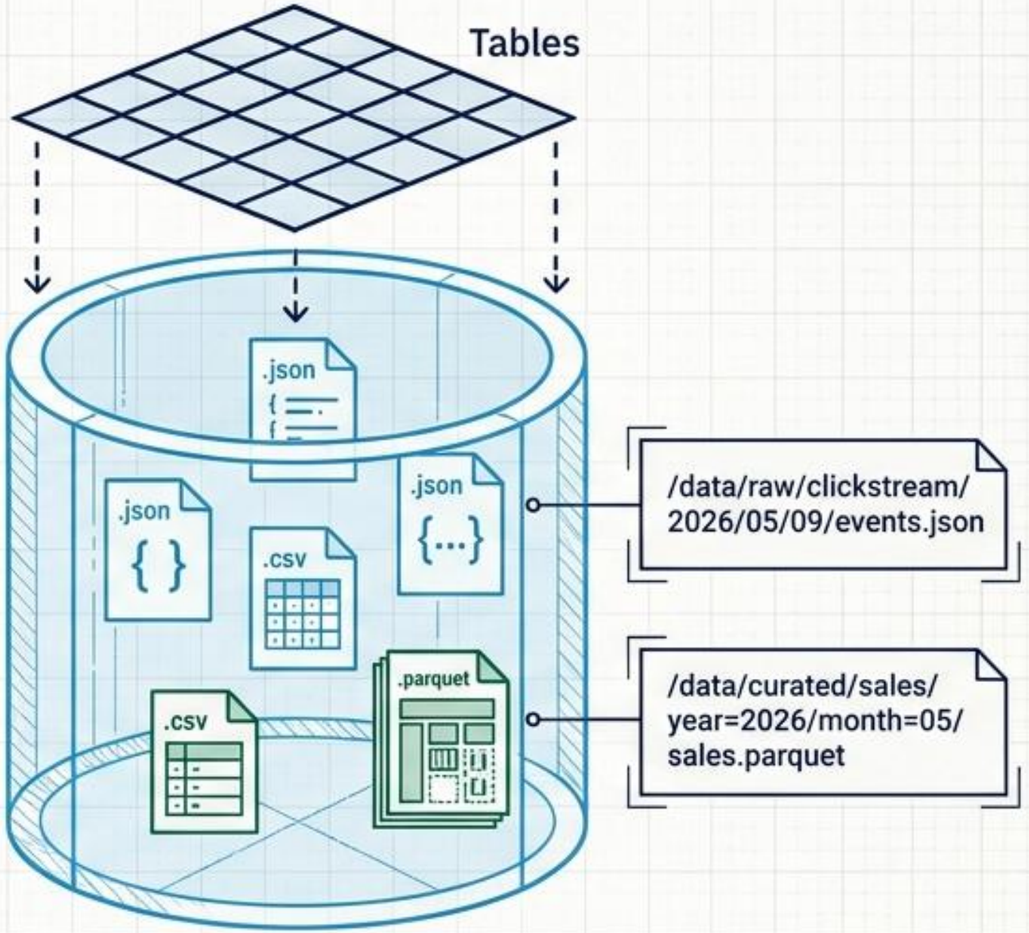


TRADITIONAL SQL DATABASE



Rigid structure must be defined before insertion

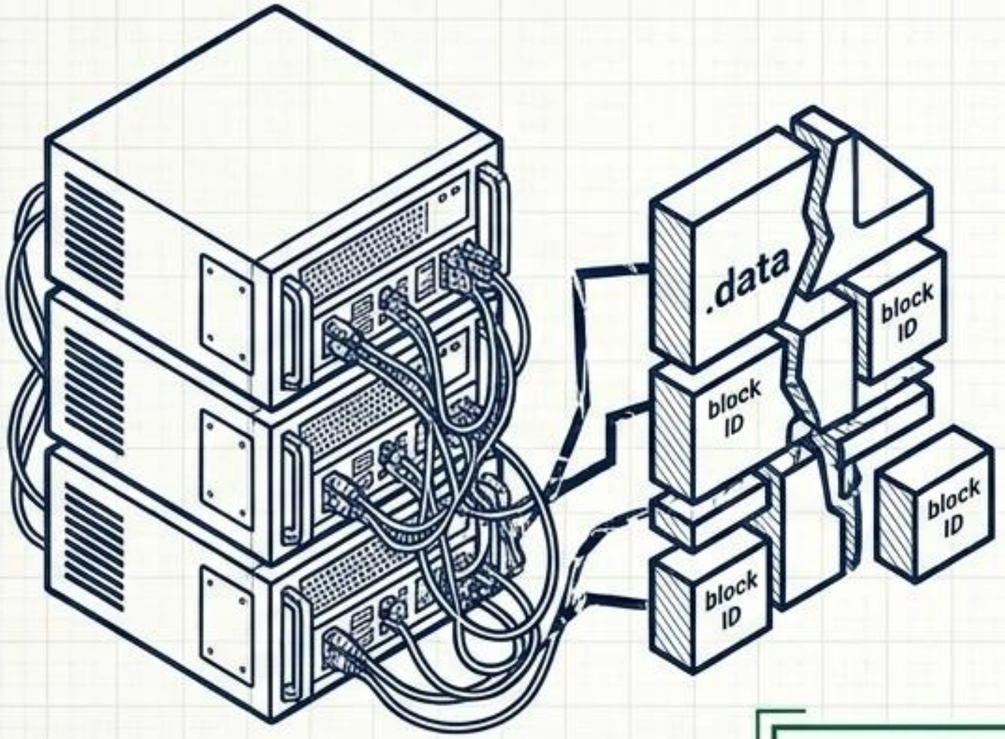
MODERN DATA LAKE



A data lake stores files first. Tables are created later on top of these files.

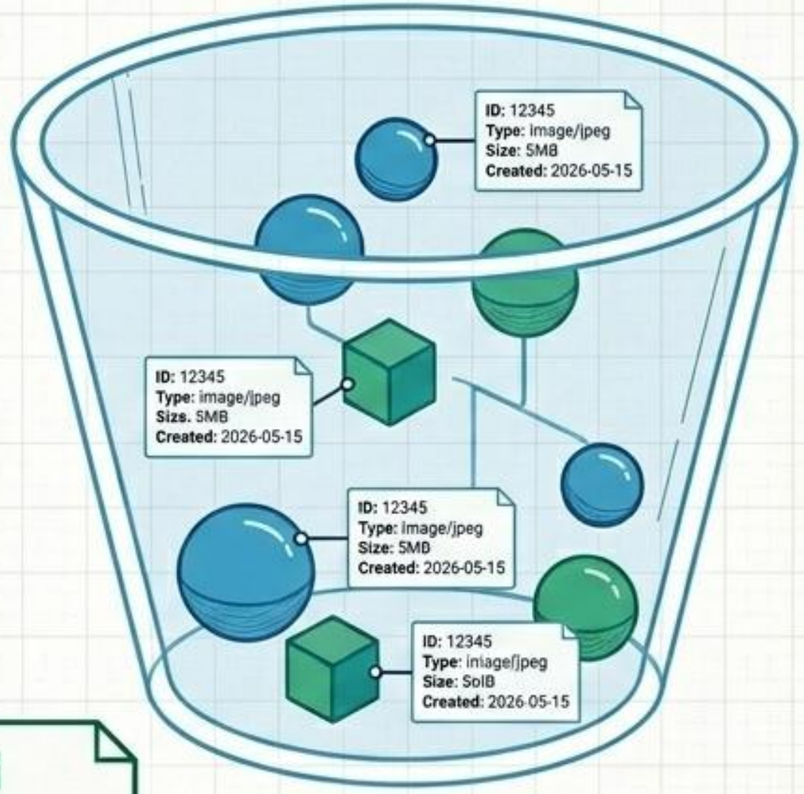
Modern data lakes rely on scalable object storage platforms

Legacy: Distributed File System (HDFS)



Files are divided into hardware blocks and distributed across physical on-premise machines.

Modern: Cloud Object Storage



Data is stored as objects inside highly scalable cloud buckets (e.g., Amazon S3, Azure Data Lake, Google Cloud Storage).

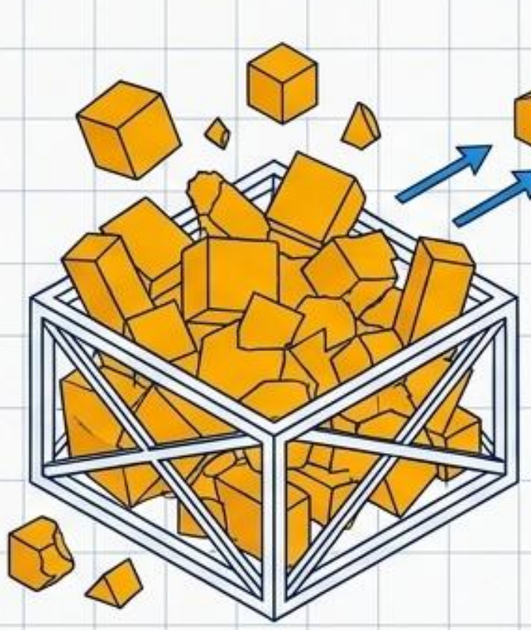
The Modern Standard

Amazon S3 acts as a common foundation, providing infinitely scalable storage, automated security, and native analytics integration without the need to manage physical clusters.

Data matures progressively through structured refinement zones

The Medallion Architecture

Bronze Layer (Raw Zone)

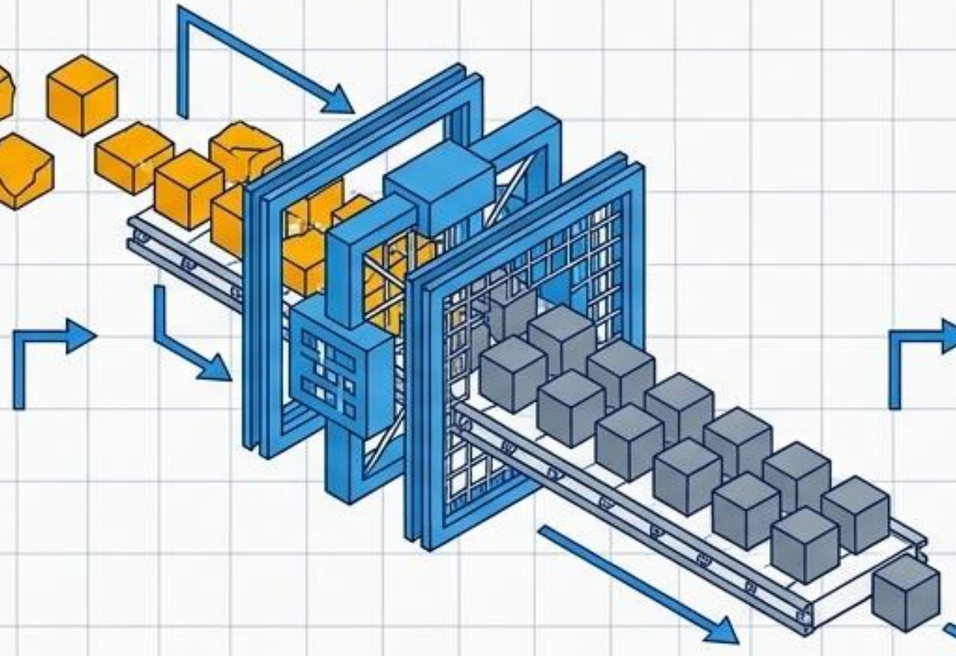


Raw ingested data. Exactly as received from the source.

University: Original JSON Canvas logs, raw SIS registration files.

Used by: Data Engineers

Silver Layer (Cleaned Zone)

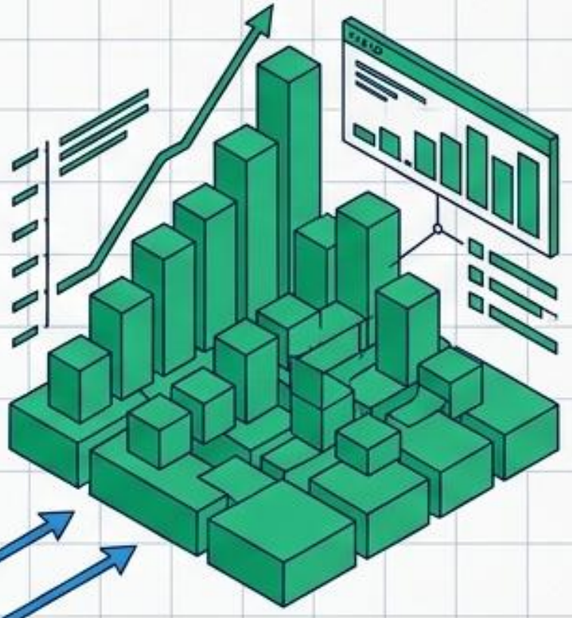


Cleaned, joined, and standardized data. Invalid records removed.

University: Standardized student-course-attendance table.

Used by: Data Scientists

Gold Layer (Curated Zone)



Business-ready, aggregated data prepared for reporting.

University: Dashboard showing attendance rate and risk indicators.

Used by: Business Analysts & BI Tools

SELECT AVG(Grade) FROM students;

Row-Based Storage (e.g., CSV)

StudentID	Name	Course	Grade
101	Ali	Big Data	85
102	Sara	Big Data	92
103	Omar	AI	78

Row-based engines must read every single field in a record to answer the query, creating massive I/O overhead.

Columnar Storage (e.g., Parquet)

StudentID	Name	Course	Grade
101	Ali	Big Data	85
102	Sara	Big Data	92
103	Omar	AI	78

Columnar engines isolate and extract only the required columns, completely bypassing irrelevant data.

Key Takeaway: Columnar storage is hyper-efficient for massive datasets where analytical queries target a fraction of the available columns.

Analytical engines require machine-optimized binary file formats

Human-Readable

Format	Type	Good For	Weakness
CSV	Row / Text	Simple, universal data exchange	No schema, highly inefficient for big data workloads.
JSON	Semi-structured / Text	Web logs, APIs, nested data structures	Massive file sizes, results in very slow analytical queries.

Machine-Optimized

Format	Type	Good For	Weakness
Avro	Row-based Binary	Data streaming, continuous schema evolution	Less ideal for deep analytical column queries.
Parquet	Columnar Binary	Heavy Analytics, Spark, Amazon Athena	Not directly human-readable without processing.
ORC	Columnar Binary	Hadoop/Hive ecosystem analytics	Less common outside of specific Hadoop ecosystems.

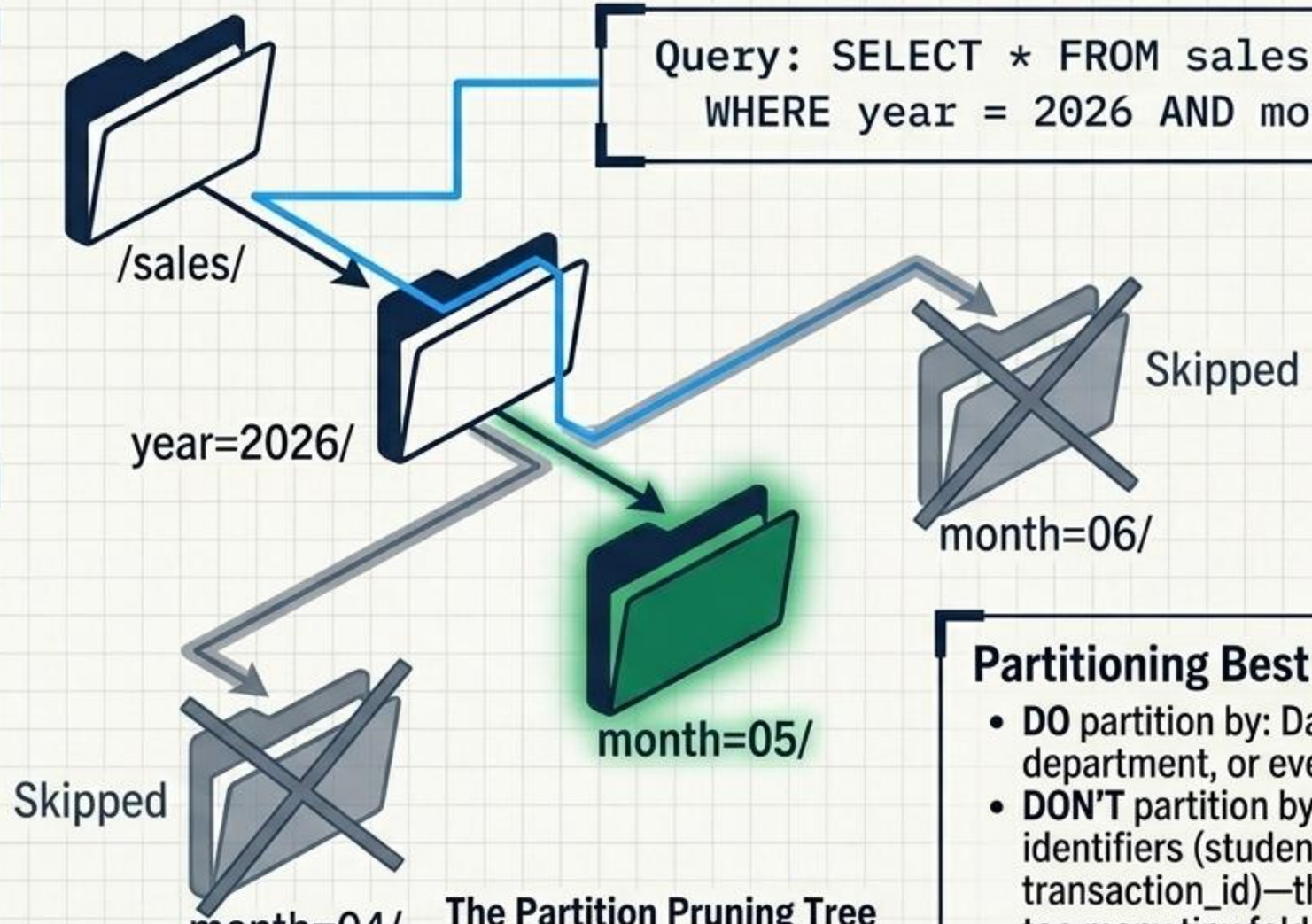
Synthesis: CSV and JSON are easy to read but fail at scale. Parquet and ORC are compressed, schema-aware, and purpose-built for massive parallel processing.

Partitioning isolates relevant data to accelerate query execution

The Mechanism

Organizing files into logical folder hierarchies based on frequently filtered columns reduces the total data scanned. Less scanning equals significantly faster queries and lower cloud computing costs.

```
Query: SELECT * FROM sales
  WHERE year = 2026 AND month = 5;
```



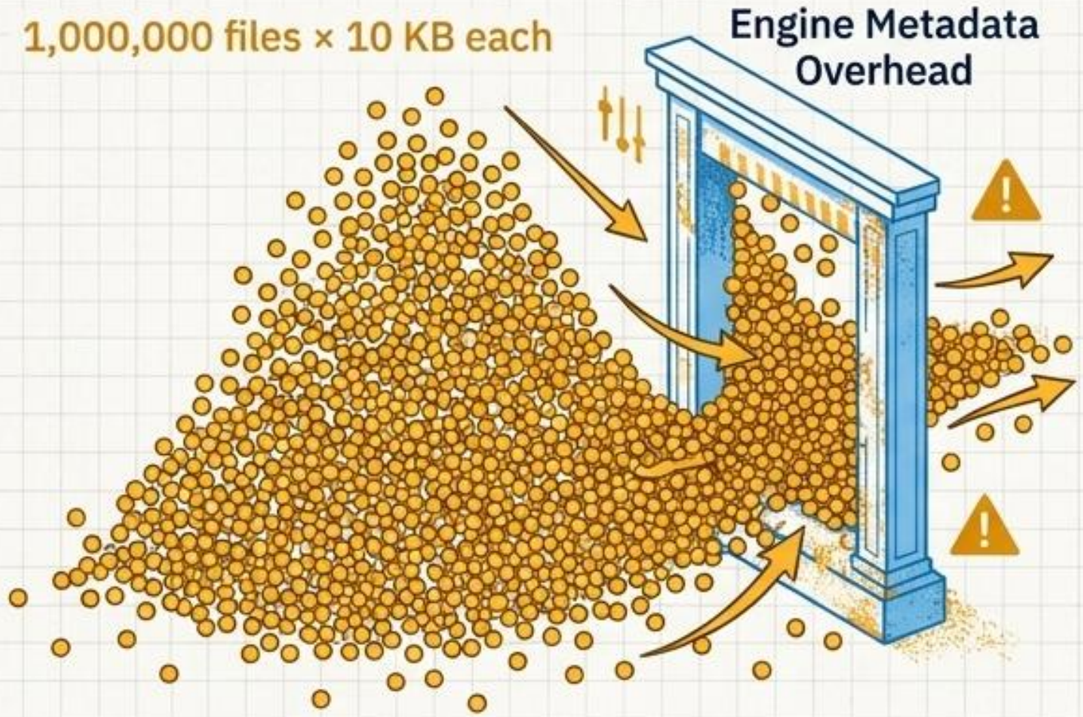
The Partition Pruning Tree

Partitioning Best Practices

- **DO** partition by: Date, region, department, or event_type.
- **DON'T** partition by: Unique identifiers (student_id, email, transaction_id)—this creates too many tiny folders.

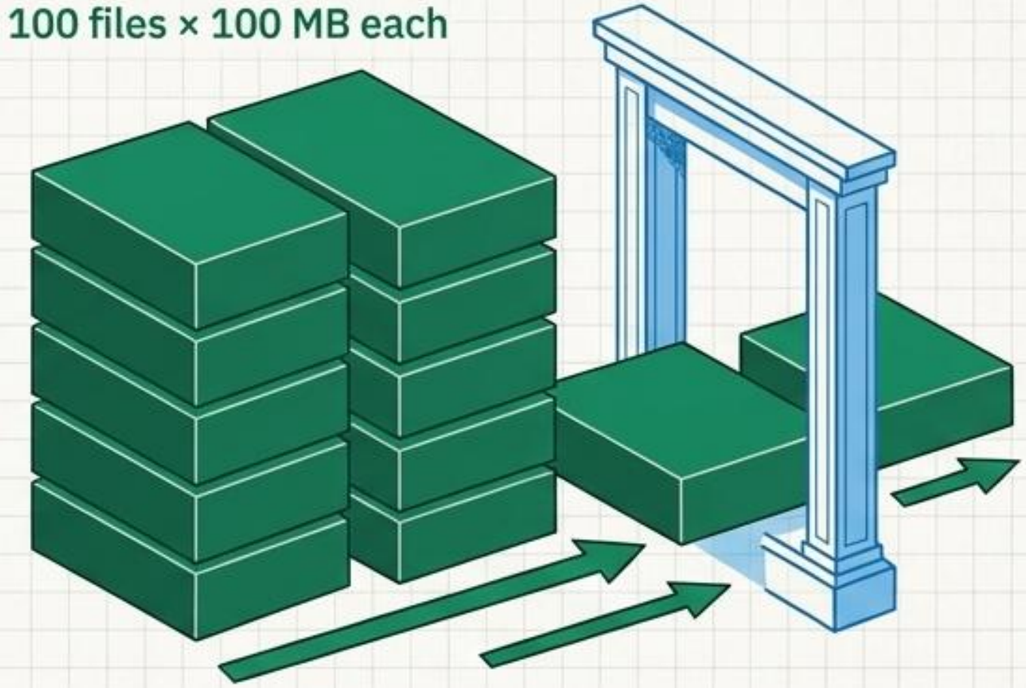
Millions of tiny files create catastrophic engine overhead

Failing System Architecture



Engines like Spark or Presto must individually open, list, and schedule work for every single file. Too many files choke the system.

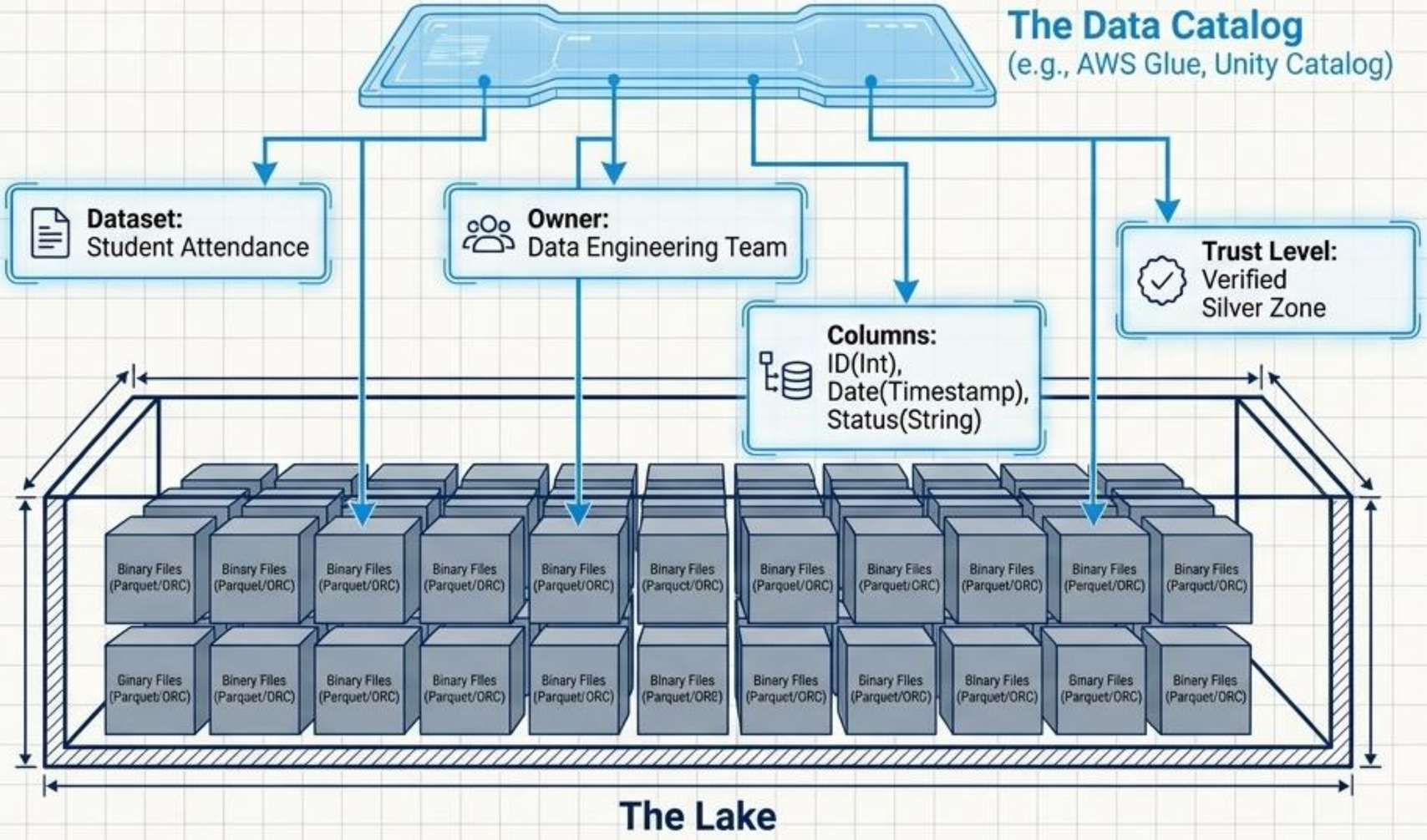
Optimized System Architecture



Larger, batched files minimize metadata overhead, allowing analytical engines to maximize throughput and processing speed.

The Golden Rule: In big data, the total NUMBER of files impacts engine performance just as much as the overall SIZE of the data.

Data catalogs bridge the gap between file storage and human usability



The Core Concept

The raw files are the physical data. The catalog is the map that explains the data.

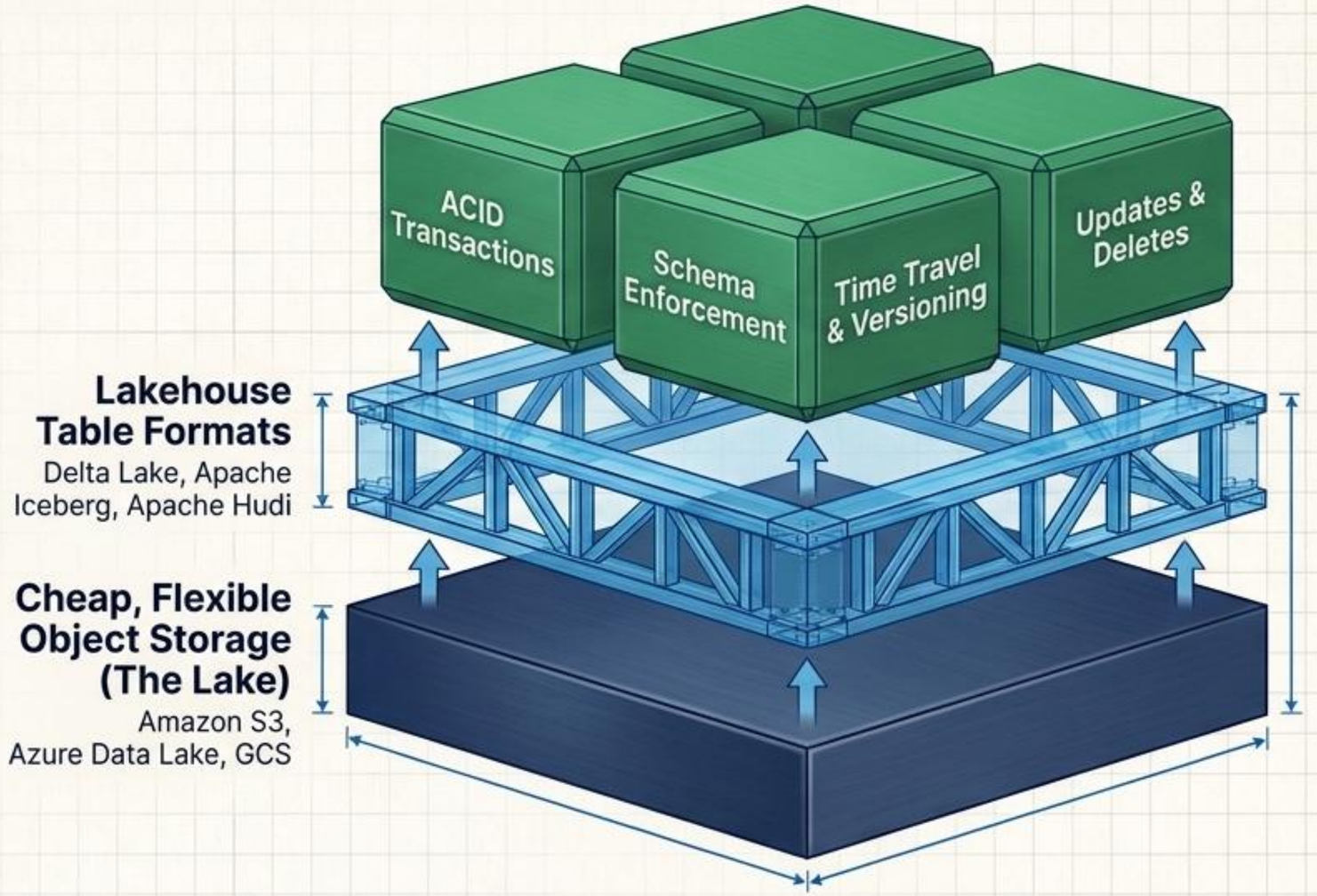
The Risk

Without an active metadata catalog, users cannot search, understand, or trust the data, turning an expensive lake into an unusable swamp.

The Lakehouse architecture merges storage flexibility with database reliability

The Synthesis

The Lakehouse resolves the historical tension. It retains the low-cost, infinitely scalable file storage of a data lake, but overlays the strict table management and reliability guarantees previously unique to traditional data warehouses.

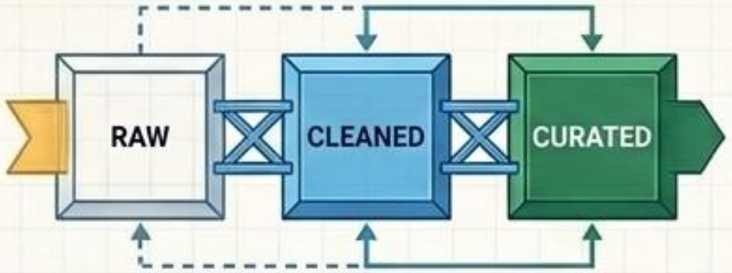


The Six Pillars of Modern Data Storage



Files First

Big data is stored initially as files in scalable cloud buckets, not rigid tables.



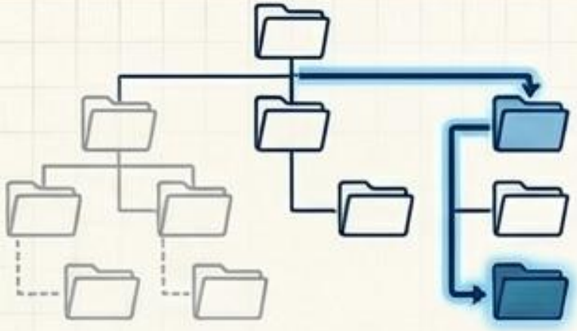
Zone Architecture

Data matures progressively through structured layers (Raw, Cleaned, Curated).



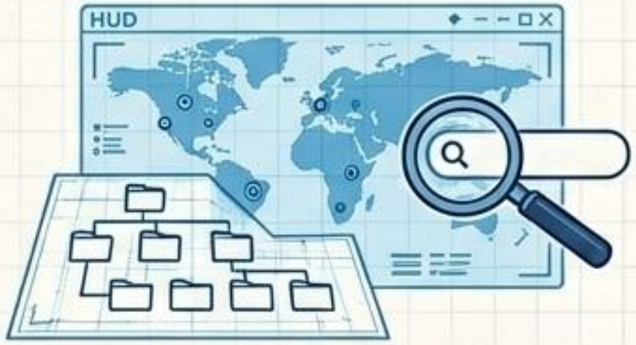
Format Efficiency

Machine-optimized columnar formats (Parquet/ORC) dictate analytical performance.



Smart Partitioning

Organizing files logically allows queries to bypass massive amounts of irrelevant data.



Active Catalogs

A metadata layer makes an unstructured lake visible, searchable, and usable to humans.



Strict Governance

Access control, ownership, and quality checks prevent the lake from degrading into a swamp.