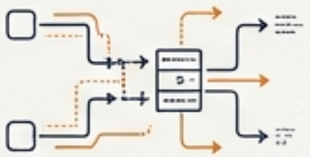
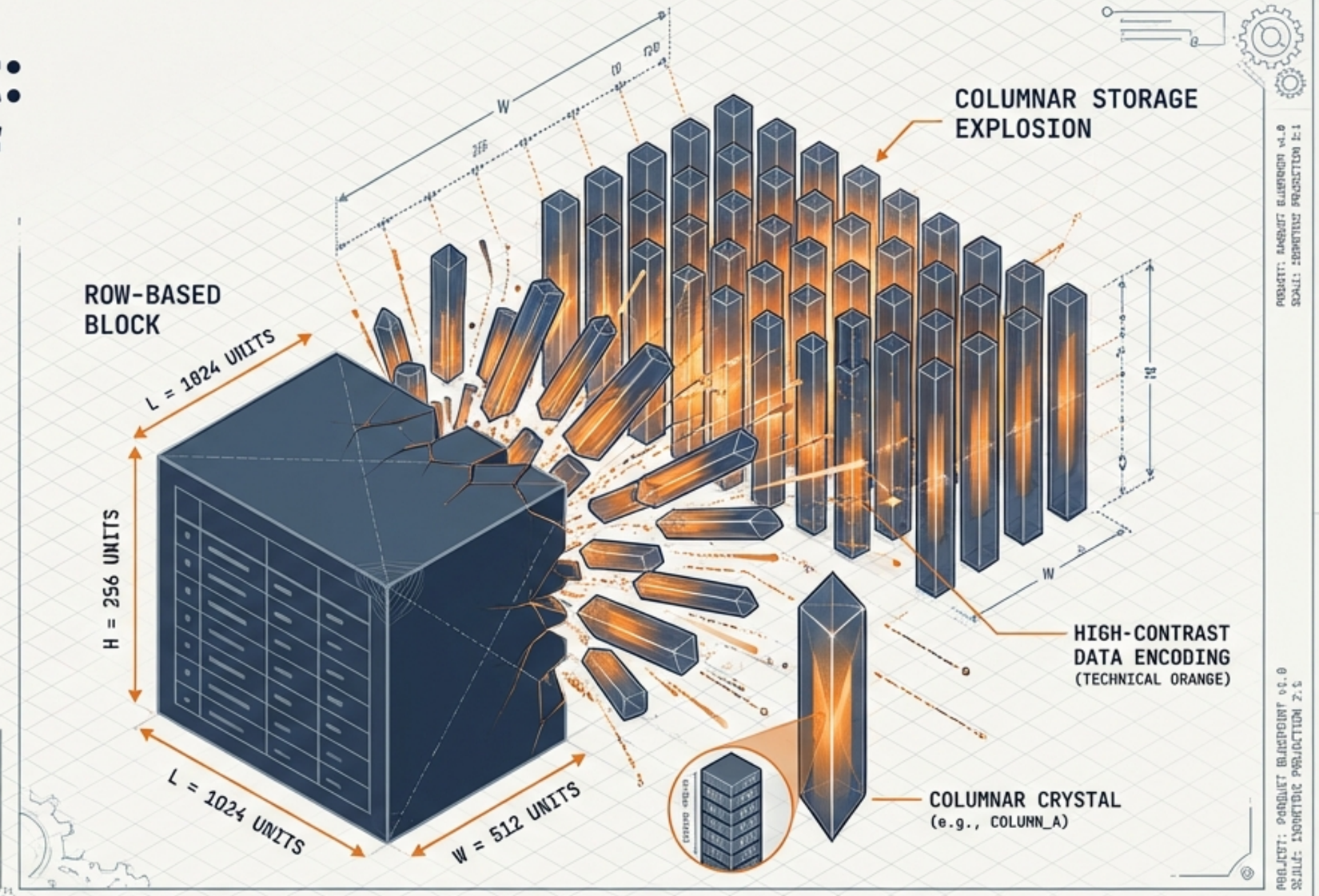
Apache Parquet: The Anatomy of Efficiency

A Deep Dive into the Standard for Modern Data Analytics



- Optimised for the Data Lakehouse
- Engineered for Speed
- The Industry Standard



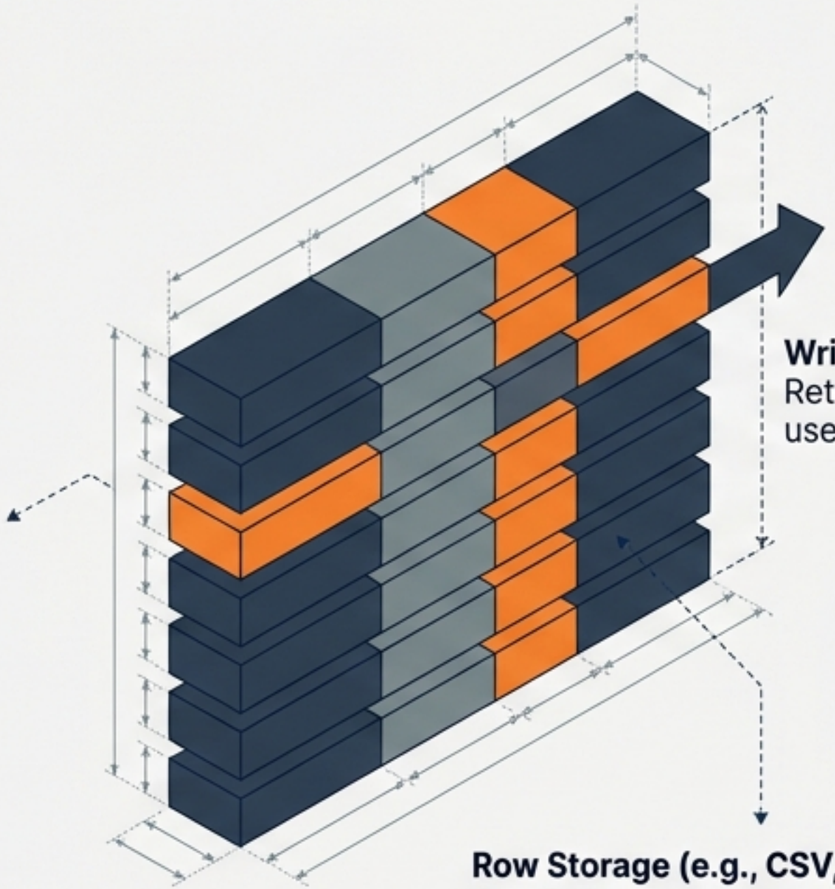
PRECISION ENGINEERING BLUEPRINT

PROJECT: PARQUET BLUEPRINT v1.0	Helvetica Now Display
SCALE: ISOMETRIC PROJECTION 2:1	Helvetica Now Display

APPROVED FOR DATA ANALYTICS

The Paradigm Shift: From Transactions to Analytics

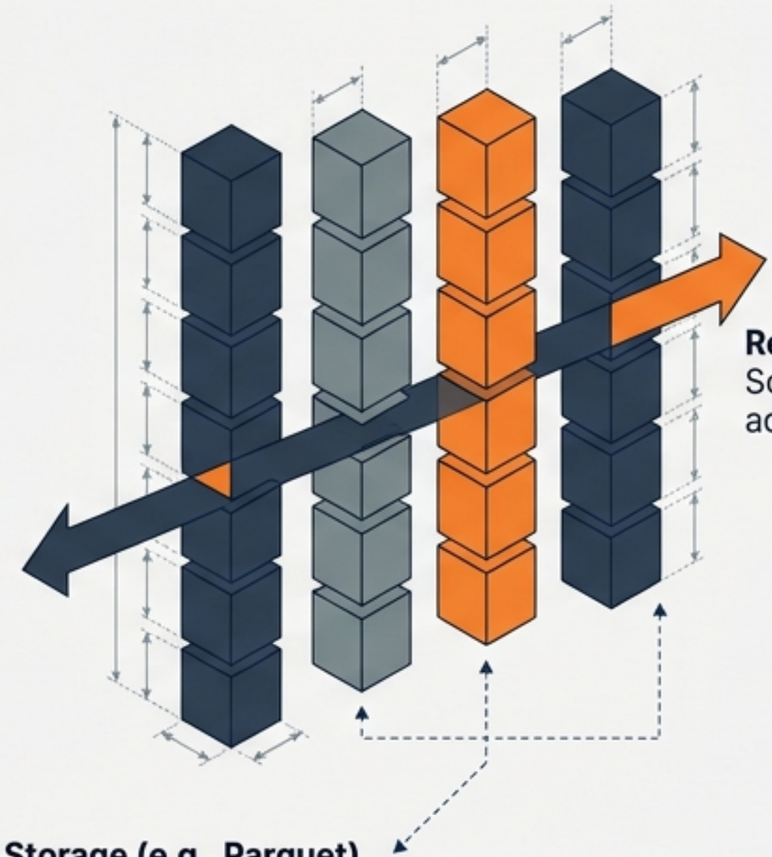
Row-Oriented / OLTP



Write-Optimised:
Retrieve one user's full profile.

Row Storage (e.g., CSV, Avro)
Writes are fast and sequential.
Best for transactional workloads where you need to retrieve one specific record's entire set of attributes.

Column-Oriented / OLAP



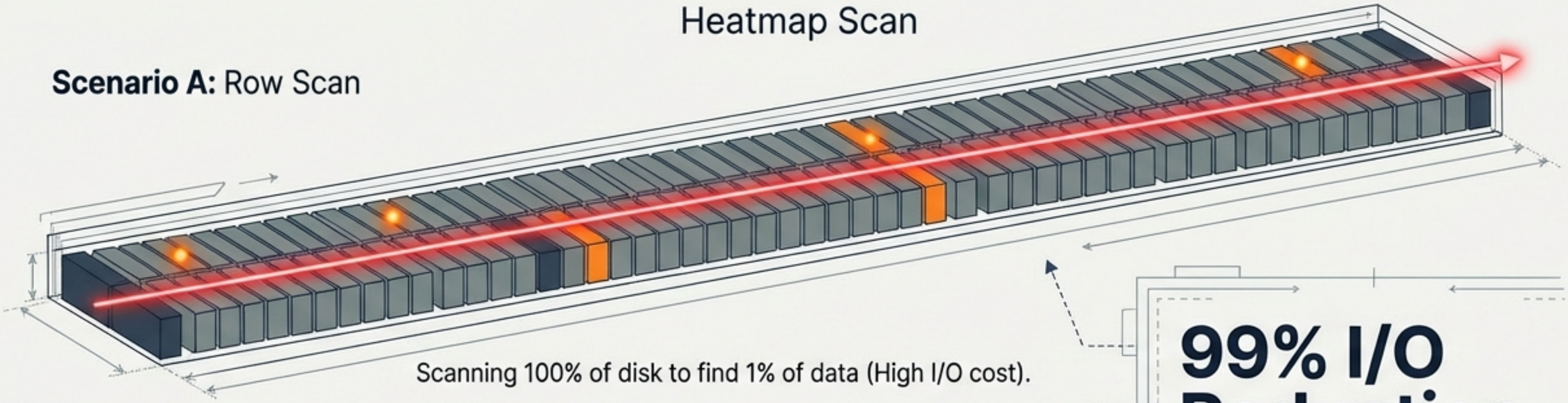
Read-Optimised:
Scan one metric across billions.

Column Storage (e.g., Parquet)
Reads are selective.
Best for analytical workloads where you need to compute a metric (like Average Age) across millions of users without loading their Names or Addresses.

The I/O Arbitrage

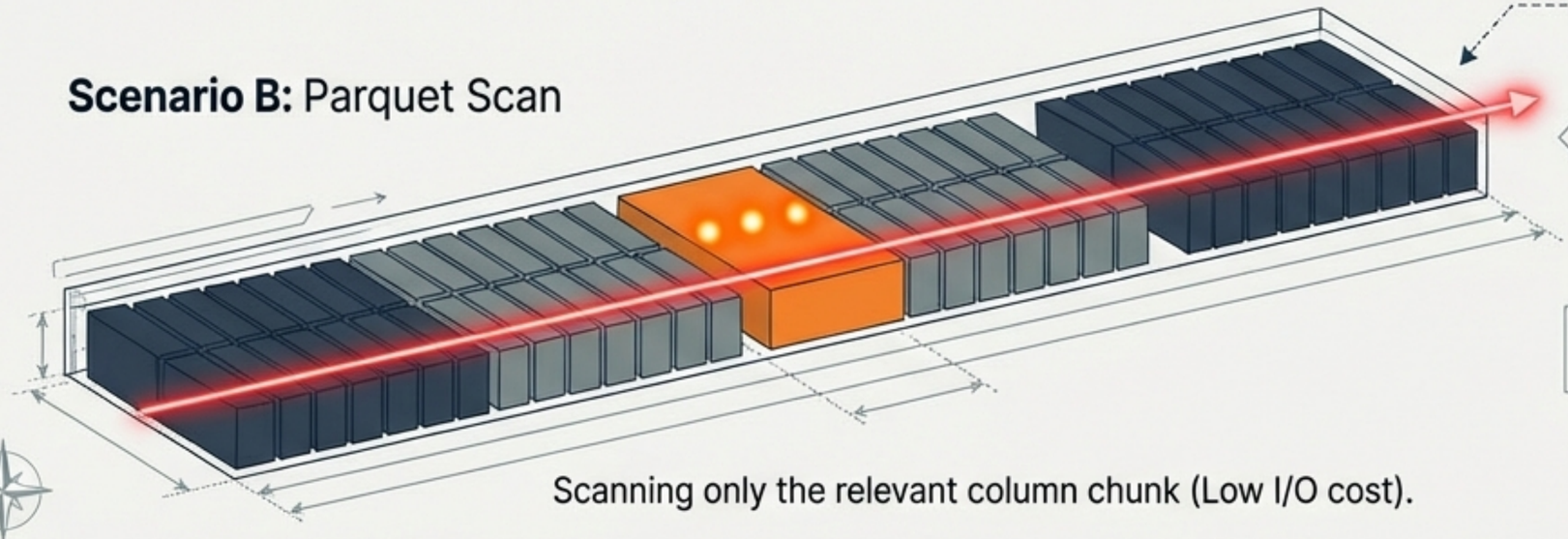
Heatmap Scan

Scenario A: Row Scan



Scanning 100% of disk to find 1% of data (High I/O cost).

Scenario B: Parquet Scan



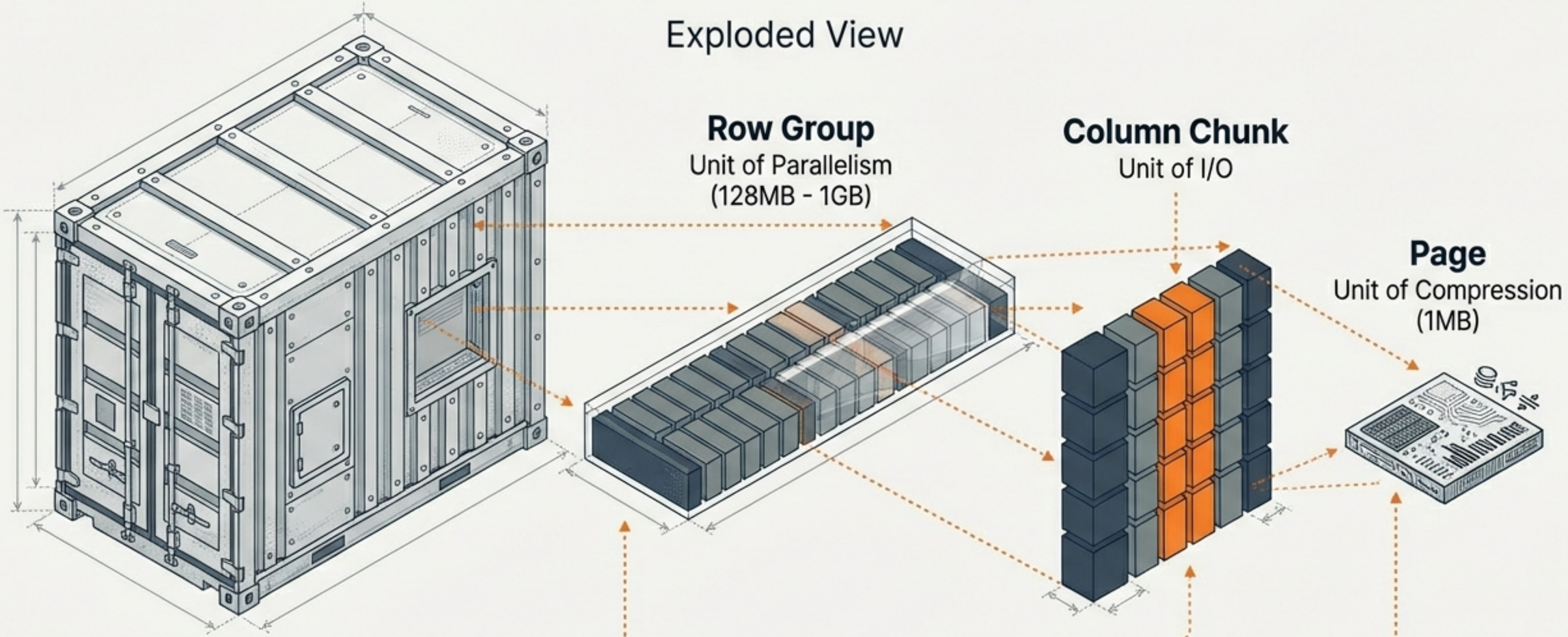
Scanning only the relevant column chunk (Low I/O cost).

99% I/O Reduction

Column Pruning: In wide tables with hundreds of columns, analytic queries often touch only 3-4 metrics. Parquet physically skips the storage blocks of unrequested columns, meaning queries process `gigabytes` instead of `terabytes`.

The Blueprint: Parquet File Hierarchy

Exploded View



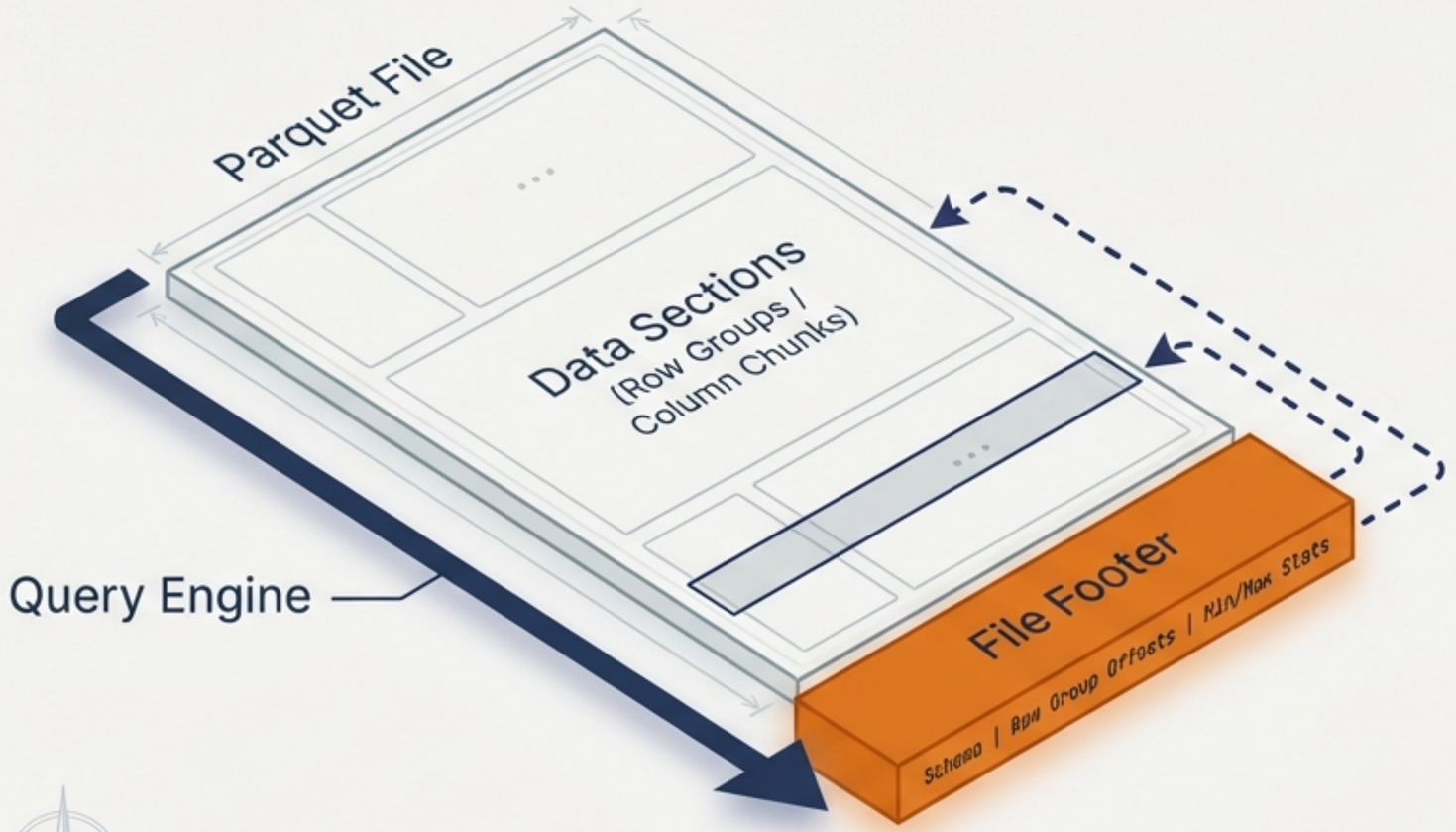
File (The Container)

Row Group
Allows distributed engines (Spark/Trino) to split a single file across multiple CPU cores.

Column Chunk
Contains the data for a single column. This is what enables column pruning.

Page
The atomic unit. This is where encoding and compression happen.

Architecture: Metadata First



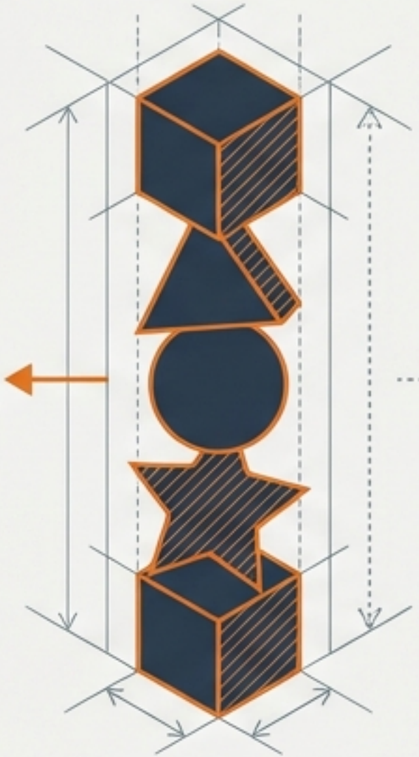
Smart Skipping

- Unlike CSV, Parquet is **self-describing**. The Footer contains the map. The query engine reads this map before processing any data.
- **Strategy**: If the metadata says a Row Group typically doesn't contain the requested data range, the engine never touches those bytes.

The Art of Compression

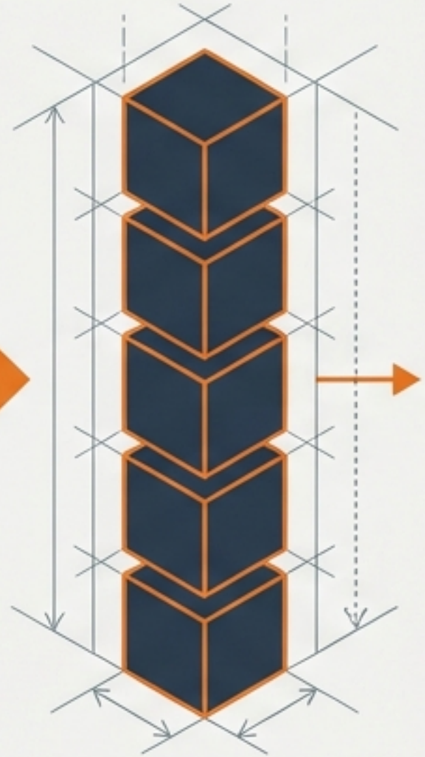
Row / High Entropy

Column / Low Entropy



Hard to Compress
(Heterogeneous Types)

Up to 75%
Storage Reduction



Easy to Compress
(Homogeneous Types)

The Physics of Homogeneity: Storing similar data types together maximizes the efficiency of compression codecs.

Codec Selection:

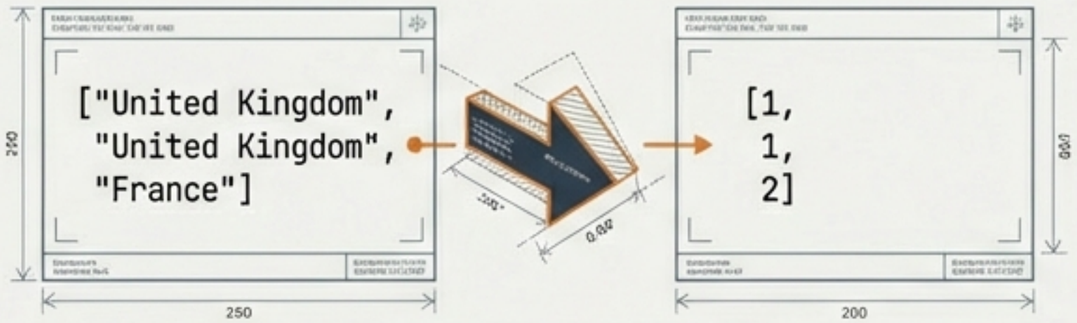
- **Snappy**: Optimised for speed (Hot Data).
- **Gzip/ZSTD**: Optimised for storage ratio (Cold Data).
- **ZSTD**: The modern gold standard for balancing both.

Precision Engineering Blueprint

Encoding: The Mathematics of Reduction

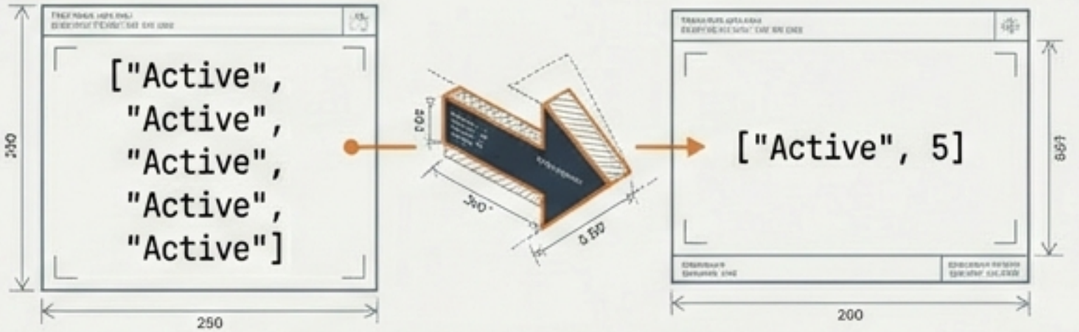
Logical reduction before physical compression.

Dictionary Encoding



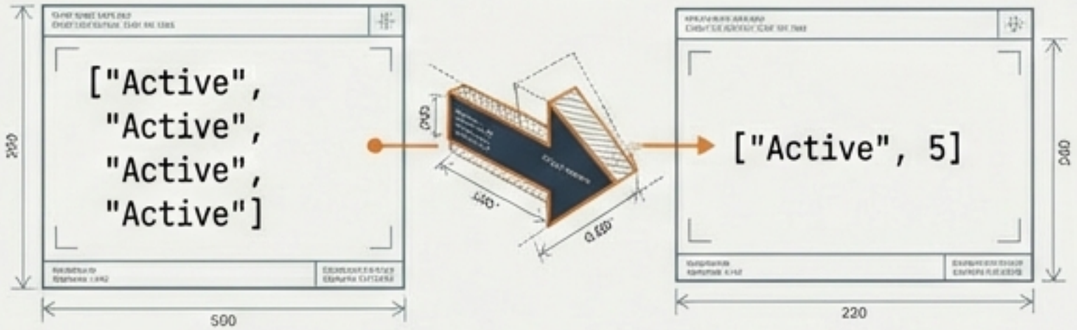
Replaces long strings with tiny integers.

Run-Length Encoding (RLE)



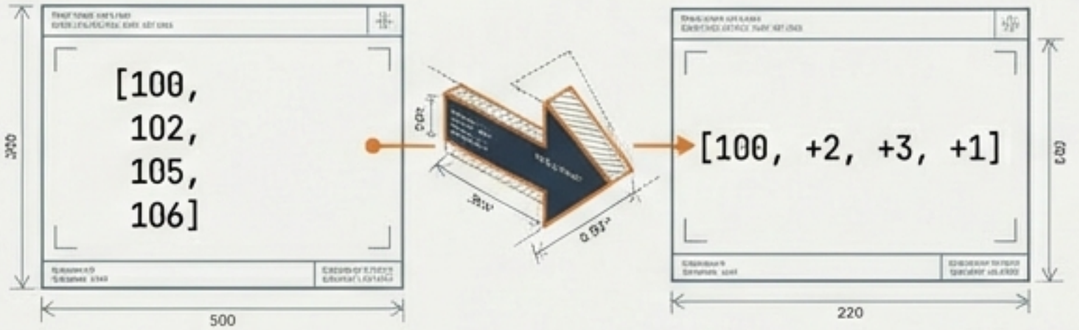
Collapses repetitive sequences.

Eudon Encoding



Collapses repetitive sequences.

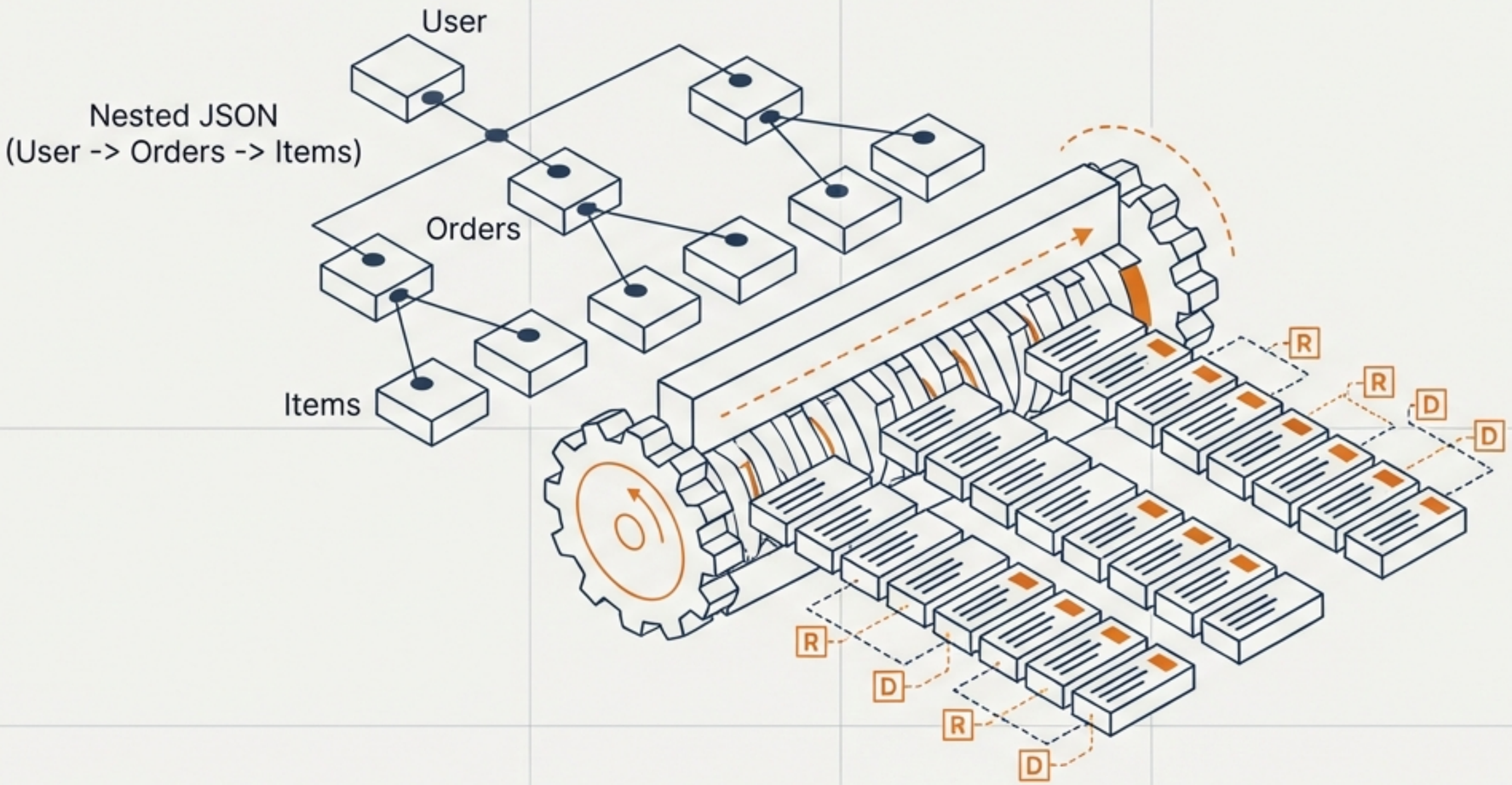
Delta Encoding



Stores small differences rather than large absolute values.

Precision Engineering Blueprint

Handling Complexity: The Dremel Algorithm

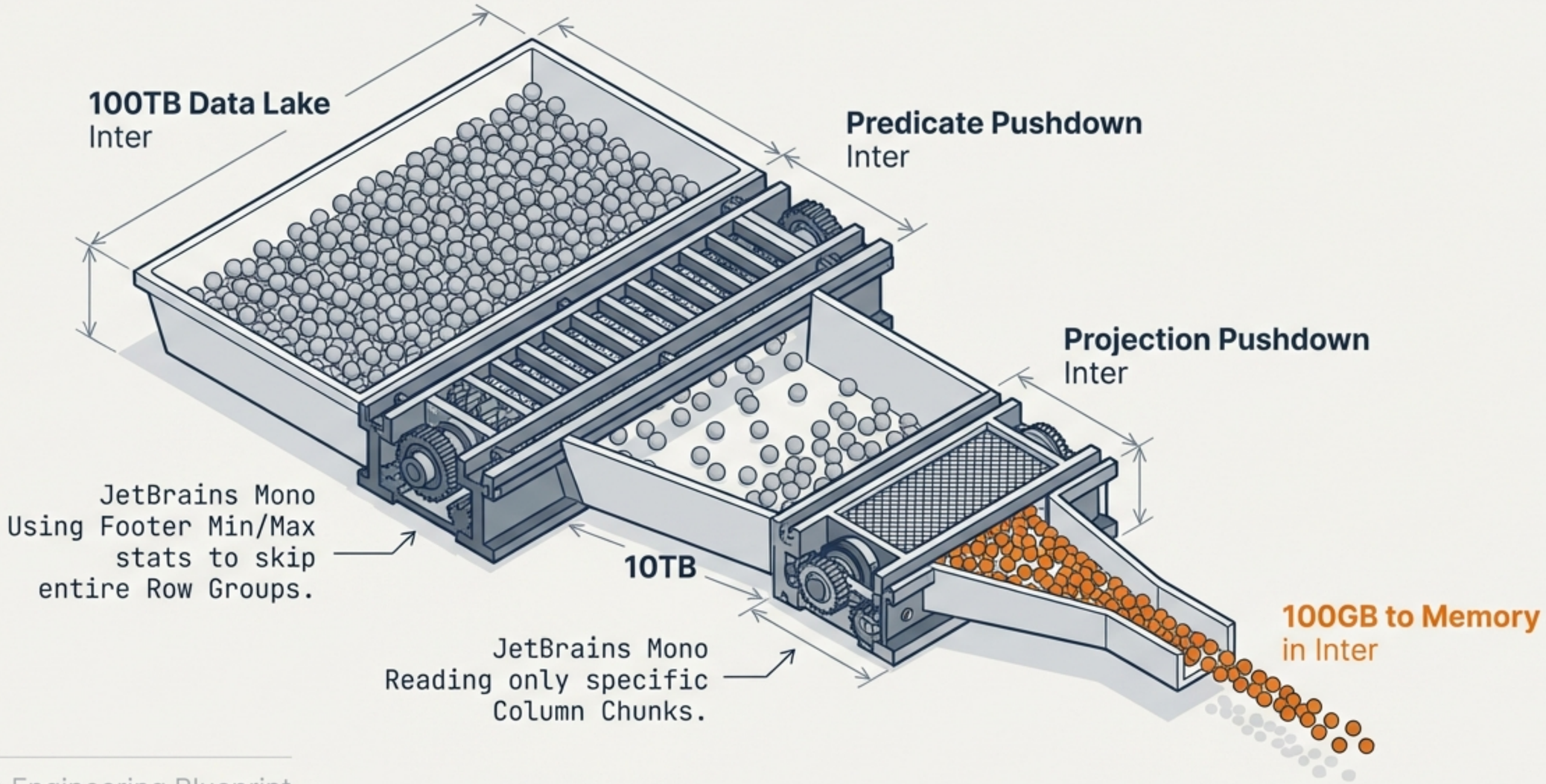


Precision Engineering Blueprint

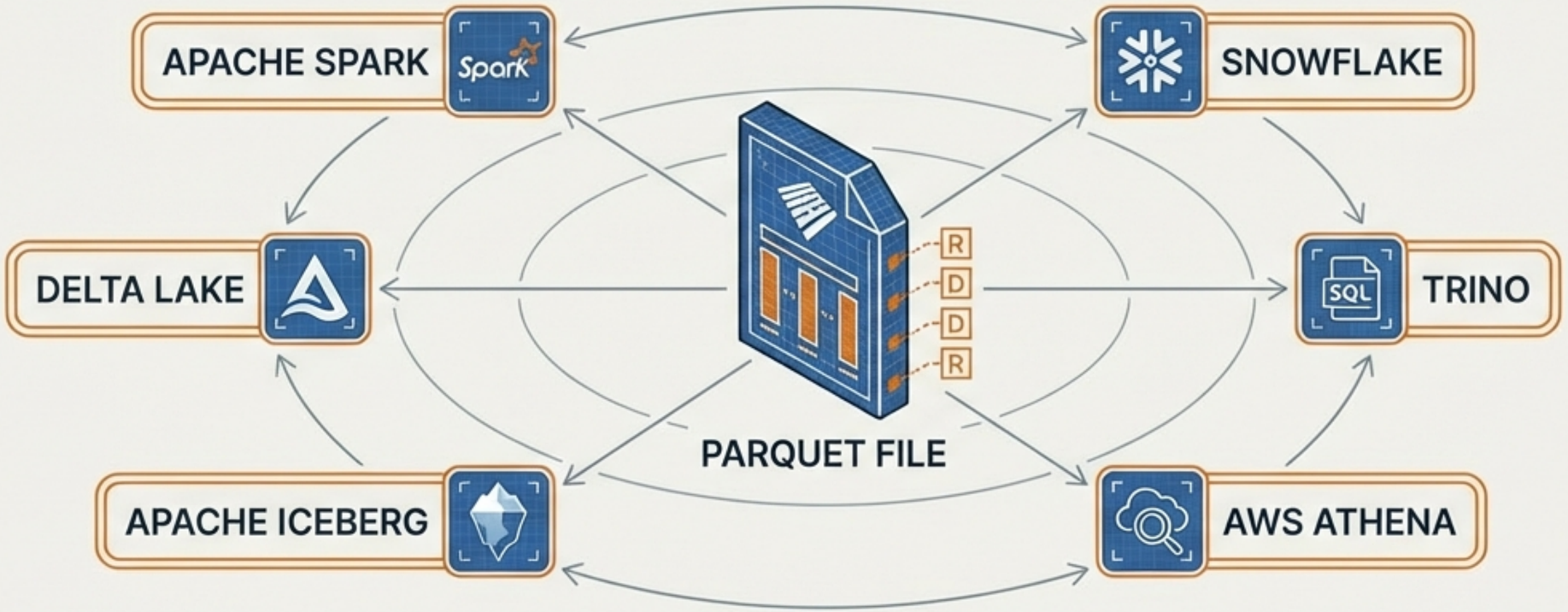
Record Shredding

Parquet flattens nested structures (arrays/maps) into columns without data duplication.
Metadata Tags (R & D Levels): Track depth and repetition, allowing the engine to reconstruct the original nested structure on read.
Benefit: Query a single field deep inside a complex object without parsing the entire tree.

Optimization: Don’t Pay for Data You Don’t Use



The Universal Currency of the Lakehouse

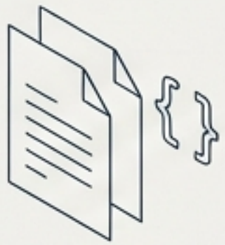


Decoupled Architecture: **Write data once** with **Spark**, read it later with **Snowflake** or **Trino**. No conversion required.

Interoperability: The foundational storage layer for modern table formats like **Iceberg** and **Delta Lake**.

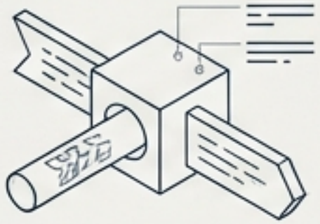
The Showdown: Row vs. Columnar

CSV / JSON



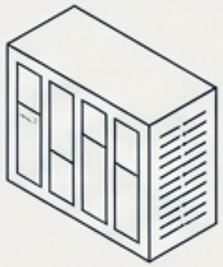
- **Use Case:** Ad-hoc, Small Data
- **Pros:** Human Readable
- **Cons:** Slow, No Schema, Bloated

Apache Avro (Row)



- **Use Case:** Streaming (Bronze Layer)
- **Pros:** Fast Writes, Schema Evolution
- **Cons:** Slow Analytical Reads

Apache Parquet (Column)



- **Use Case:** Analytics (Silver/Gold Layer)
- **Pros:** Fast Reads, High Compression
- **Cons:** Slower Writes (Overhead)

Ingest with Avro
(Capture with fidelity).

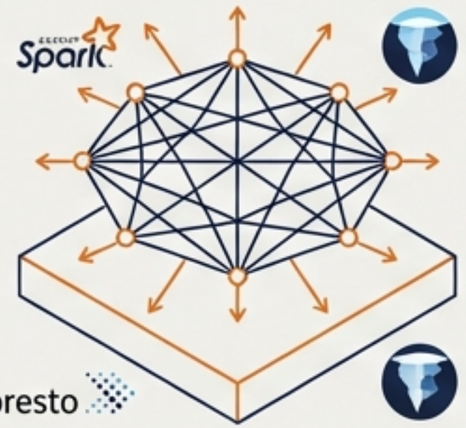
Analyse with Parquet
(Optimise for speed).

Precision Engineering Blueprint

The Columnar Landscape

ZONE 1

Parquet (The Standard)

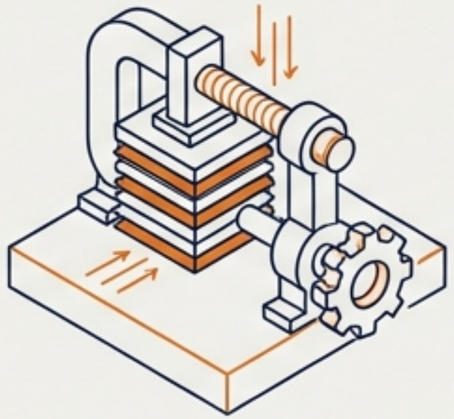


Universal ecosystem support (Spark/Presto/Iceberg).
Best all-rounder.

JetBrains Mono, slate grey

ZONE 2

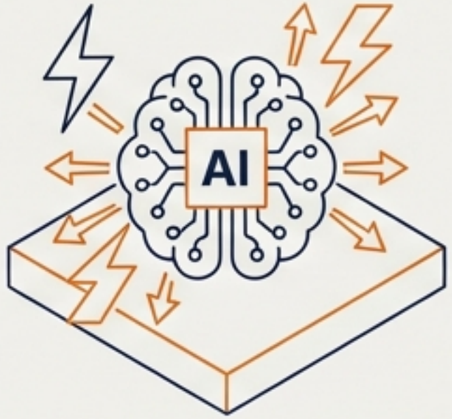
ORC (The Specialist)



Slightly better compression.
Native ACID support.
Highly optimised for Hive environments.

ZONE 3

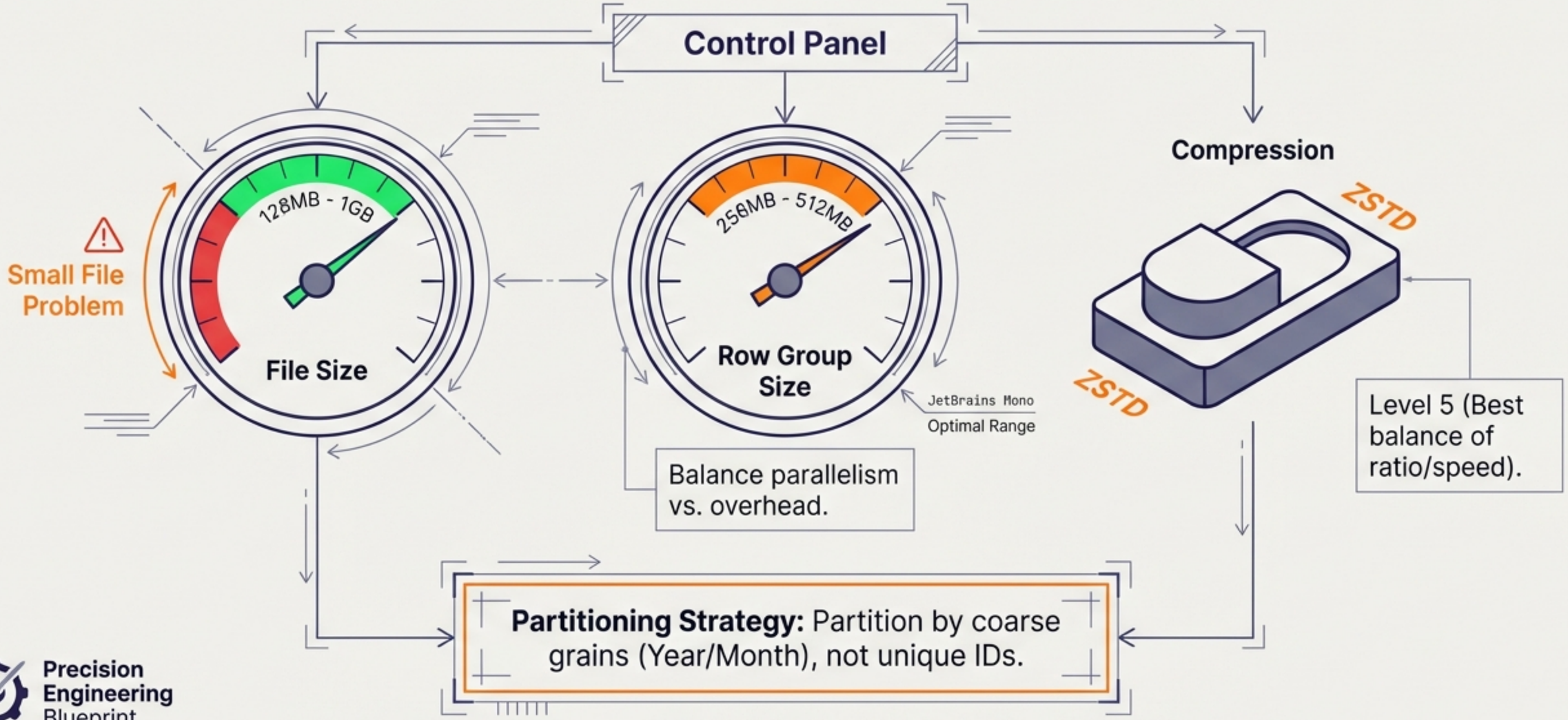
Lance / Vortex (The Challengers)



New generation. Optimised for AI/Vector Search and random access.
High performance, narrower ecosystem.

Precision Engineering Blueprint

Best Practices: Tuning for Performance



Precision Engineering Blueprint

When NOT to Use Parquet

Streaming Ingestion
High latency. Requires buffering rows before writing a column chunk.

Frequent Updates
Files are immutable. Changing one cell requires rewriting the whole Row Group.

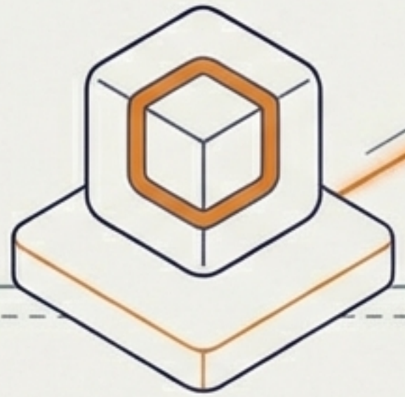
Unstructured Data
Not efficient for Images, Audio, or large text blobs.

JetBrains Mono, slate grey

Precision Engineering Blueprint

The Future of Parquet

Variant Type

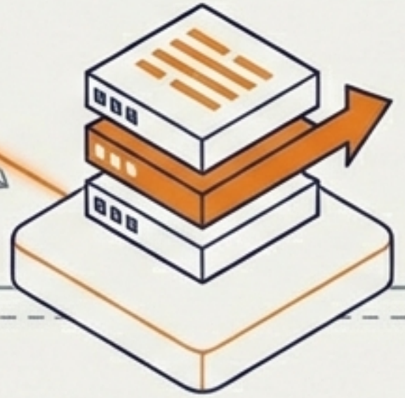


Native support for semi-structured JSON without string parsing overhead.

GeoParquet

Standardised encoding for geospatial data.

FlatBuffers



Accelerated metadata parsing.

The bedrock of modern data engineering.