An Exploration in Storing Telemetry in Cloud Object Storage

SRECon EMEA, 2024

Observability Challenges

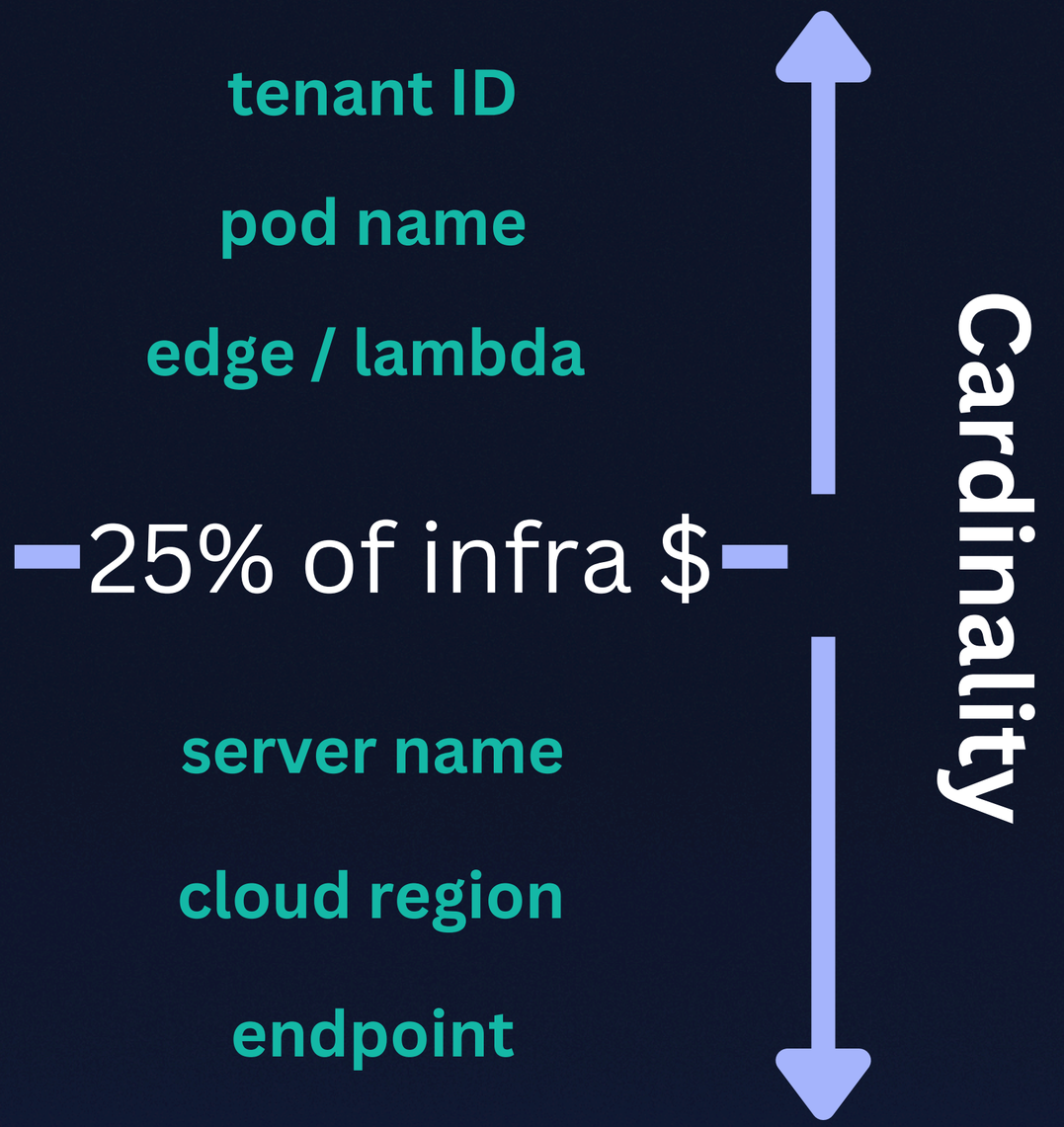
Compound growth rate of data ~20-30%

Cost is a persistent concern

Vendors have coupled storage and analysis

Hard to drive insights beyond operations

Cardinality struggle



Expanding beyond operations

31.56.96.51 [01-22-24:03:56:16] “GET /product/8284?utm_source=twitter” HTTP/1.1 390ms 200 “Mozilla/5.0 (Linux; Android 6.0)”

Operations

Latency distribution
Error counts

31.56.96.51 [01-22-24:03:56:16] “GET /product/8284?utm_source=twitter” HTTP/1.1 390ms 200 “Mozilla/5.0 (Linux; Android 6.0)”

Product

Engagement tracking
Feature use

Operations

Latency distribution
Error counts

31.56.96.51 [01-22-24:03:56:16] “GET /product/8284?utm_source=twitter” HTTP/1.1 390ms 200 “Mozilla/5.0 (Linux; Android 6.0)”

Security

WAF/DDoS
Geo analysis

Marketing

Campaigns
Traffic sources

Design/UX

Browser versions
Mobile/Desktop

State of Observability

Fundamental changes in technology and architectures are needed to meet tomorrow’s observability challenges

An Exploration in Storing Telemetry in Cloud Object Storage

Mike
Heffner

Ray
Jenkins

streamfold

Outline

What are telemetry data lakes?

Naive Approach: JSON

File Formats: Parquet

Table Formats: Iceburg

Future Challenges

Telemetry data lakes

All of your telemetry, in one place

The good

Separate storage from compute

Decouple telemetry ownership from vendors

Freely query and join telemetry data

Meet compliance requirements for long retention

Not so good

Requires careful planning and design

Lack of standardized query interfaces

Must consider UI and alerting

Security and data governance requirements

How would we build this?

Dataset

Web logs from ecommerce site

900k unique request paths

260k unique clients

28k unique user agents

98% GET, 1% POST, 0.5% HEAD

Zaker, Farzin, 2019, "Online Shopping Store - Web Server Logs", https://doi.org/10.7910/DVN/3QBYB5, Harvard Dataverse, V1

Query

1. Select by time range
2. Filter where method is GET or POST
3. Group by (method, response status)
4. Count total requests

Stage 1: JSON

JSON

10M web log entries

NDJSON structured

4.4GB uncompressed


```
{
 "client_ip": "31.56.96.51",
 "method": "GET",
 "status_code": 200,
 "timestamp_nanos": 1548116776000000000,
 "resp_bytes": 5667,
 "path": "/image/60844/productModel/200x200",
 "http_version": "1.1",
 "referrer": "https://www.zanbil.ir/m/filter/b113",
 "agent": "Mozilla/5.0 (Linux; Android 6.0; ....)"
}
```

JSON

Go CLI tool using encoding/json

Read log entries logs.json{.gz}

Scan entries, matching time range and method

Record result in Go map

Use instrumented io.Reader to track bytes read

Result

Uncompressed (4.4GB):

Query time: 57 secs

Analyzed 150MB, scanned 4.4GB - 3.3%

Compressed (285MB):

Query time: 71 secs

Analyzed 150MB, scanned 285MB - 50%

Stage 2: Efficient File Formats

Parquet

Released 2013 by Twitter and Cloudera

Open source and widely supported

File format based on PAX

Enables efficient use of CPU Cache

Optimized compression and encoding

Natassa Ailamaki, David DeWitt, Mark Hill, and Marios Skounakis 2001, "Weaving Relations for Cache Performance", https://www.vldb.org/conf/2001/P169.pdf

Parquet

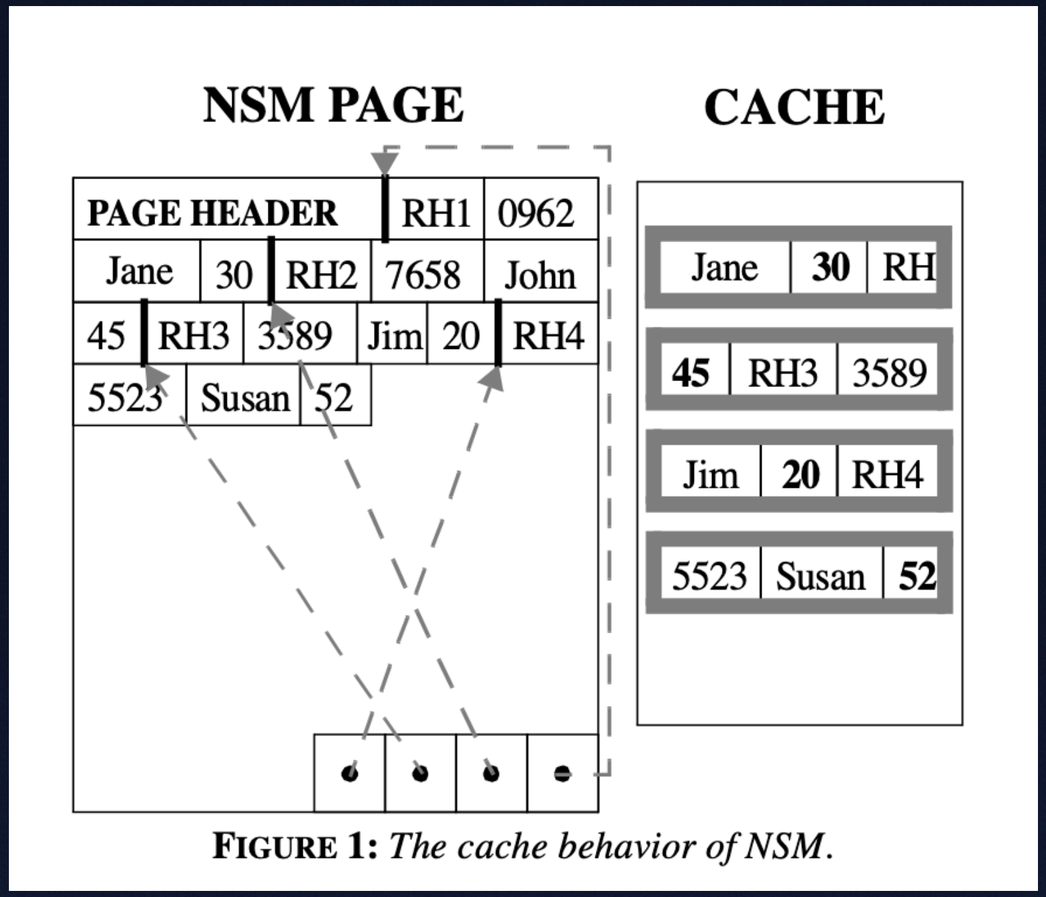


FIGURE 1: *The cache behavior of NSM.*

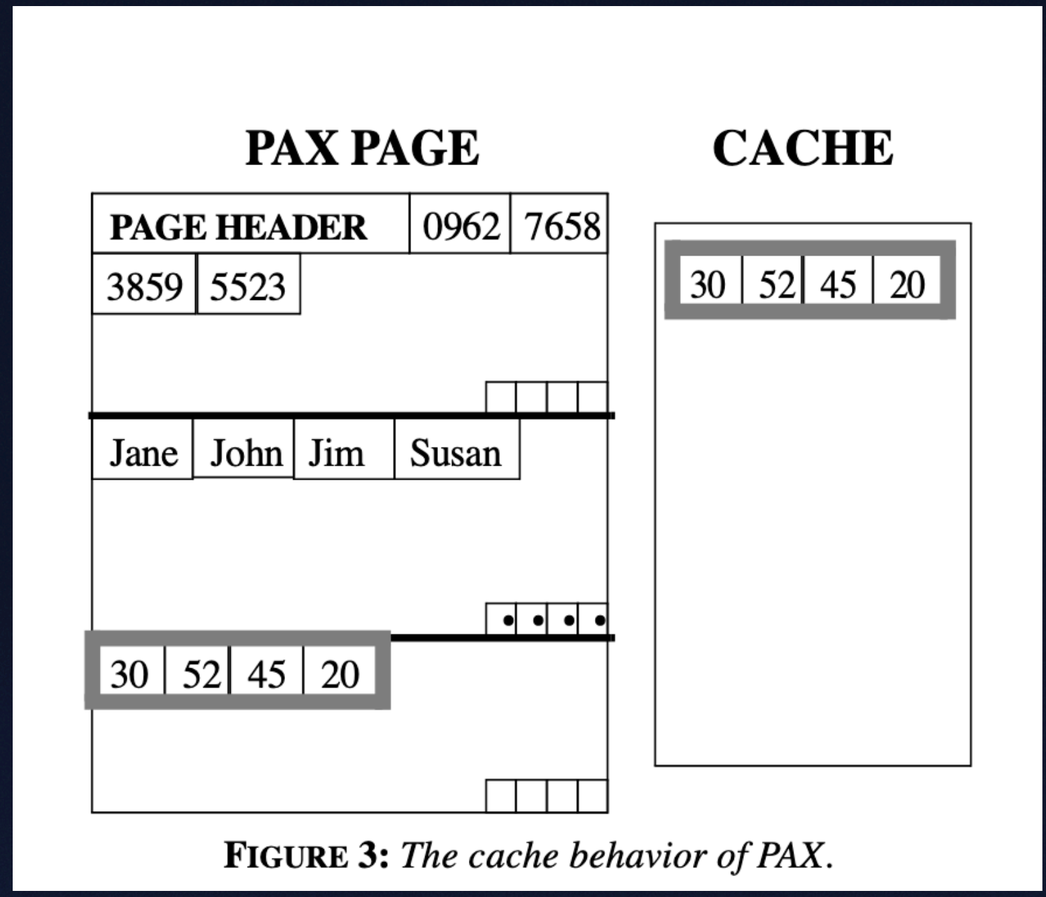
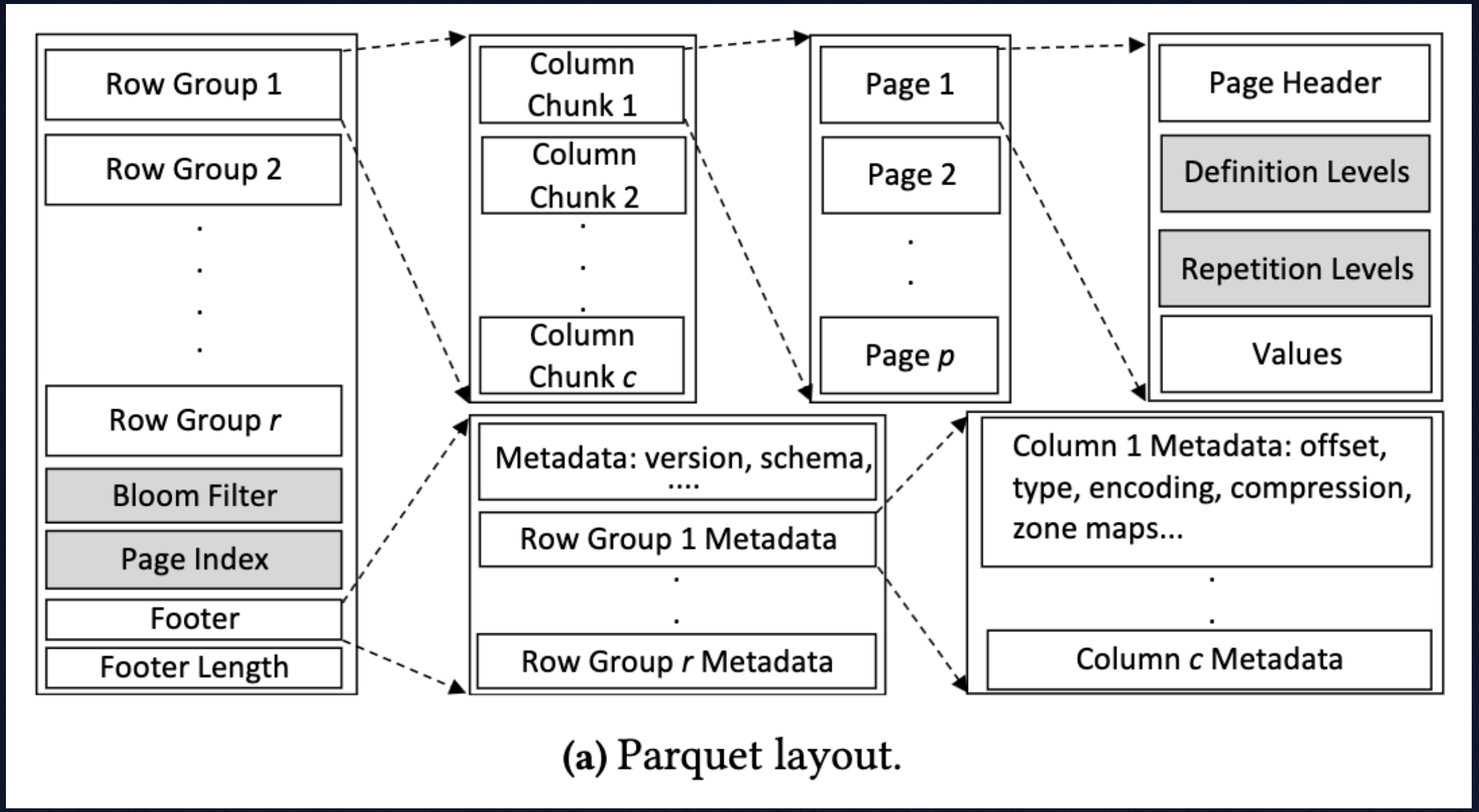


FIGURE 3: *The cache behavior of PAX.*

Natassa Ailamaki, David DeWitt, Mark Hill, and Marios Skounakis 2001, "Weaving Relations for Cache Performance", https://www.vldb.org/conf/2001/P169.pdf

Parquet Layout



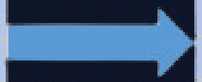
(a) Parquet layout.

- Table partitioned into Row Groups
- Tuples of columns-by-column or “chunks”
- Chunks further divided into pages (smallest unit), where encoding and compression is applied.
- Row Group and Column Chunk metadata and zone maps.
- Bloom Filters and supports predicate pushdown

Product	Quantity	OrderDate
Product A	1	06/12/2021 19:01:15.000
Product A	1	07/12/2021 19:01:16.000
Product B	1	08/12/2021 19:01:16.231
Product C	2	09/12/2021 19:01:17.000
Product A	1	10/12/2021 19:01:18.000
Product B	1	11/12/2021 19:01:19.565
Product B	2	12/12/2021 19:01:20.000
Product A	2	13/12/2021 19:01:20.876
Product A	2	14/12/2021 19:01:21.500
Product C	1	15/12/2021 19:01:22.000

Dictionary Encoding

Product
Product A
Product A
Product B
Product C
Product A
Product B
Product B
Product A
Product A
Product C

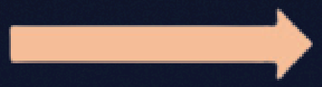


Product
0
0
1
2
0
1
1
0
0
2

Dictionary (Product)	
0	Product A
1	Product B
2	Product C

Run Length Encoding (RLE)

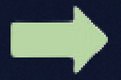
Quantity
1
1
1
2
1
1
3
3
3
3



Quantity (RLE)
3
1
1
2
2
1
4
3

Delta Encoding

OrderDate
06/12/2021 19:01:15
07/12/2021 19:01:16
08/12/2021 19:01:18
09/12/2021 19:01:23
10/12/2021 19:01:25
11/12/2021 19:01:29
12/12/2021 19:01:31
13/12/2021 19:01:32
14/12/2021 19:01:35
15/12/2021 19:01:38



delta (s)
0
1
2
5
2
4
2
1
3
3

1st OrderDate
06/12/2021 19:01:15

https://dataninjago.com/2021/12/07/databricks-deep-dive-3-parquet-encoding/

Parquet Conversion

	JSON	Parquet	%
Uncompressed	4.4 GB	391 MB	-91%
Compressed	285 MB	160 MB	-43%

Parquet / DuckDB results

```
SELECT method, status_code, count(*) FROM logs WHERE
method IN ('GET', 'POST') and timestamp_nanos >= 1548176400
000000000 and timestamp_nanos <= 154843560000000000
    GROUP BY method, status_code;
```

Result

- Compressed:
  - Query time: 0.0337s
- Demo

Scale

How do we manage many files and petabytes or exabytes of data in a higher level abstraction like a “table”?

Lakes aren’t enough

We need to analyze the data in real-time.

Unbundling of the Cloud Data Warehouse will change everything

Stage 3: Table Formats

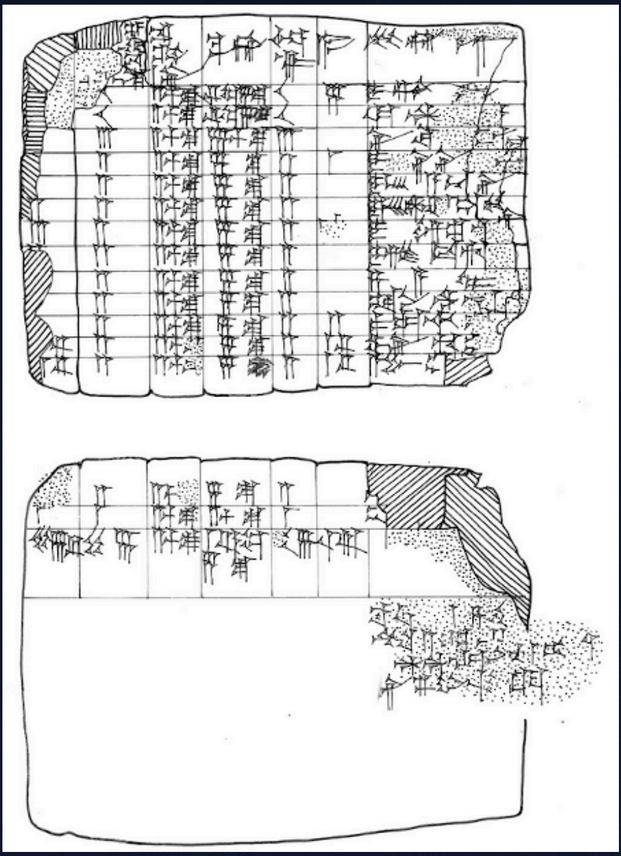
File Formats vs Table Formats

File formats like Parquet help you work efficiently with a single file, i.e. pruning, skipping, modifying etc.

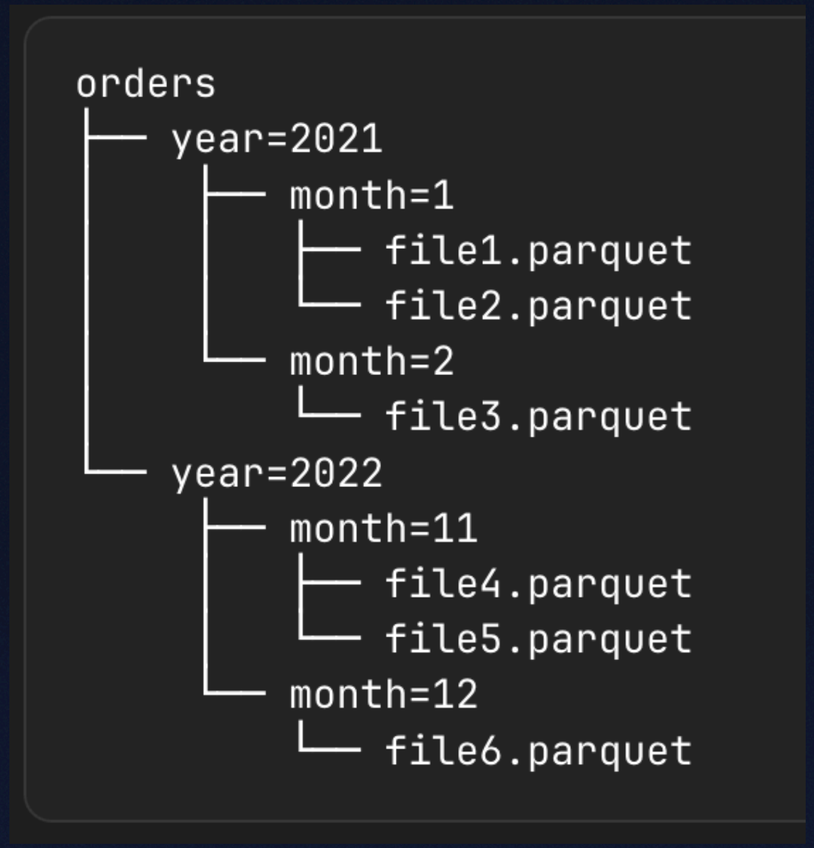
Table formats do this for a group or set of many files.

Table Formats

- Really old, ~3500 years, E.F. Codd: Relational Model 1970
- Logical abstraction and layer of indirection over real files stored on disk, while providing a unified two-dimensional tabular view of data
- Traditionally bundled into RDBMS
- Helps enable nice things like schema evolution, hidden partitioning, and serializable isolation



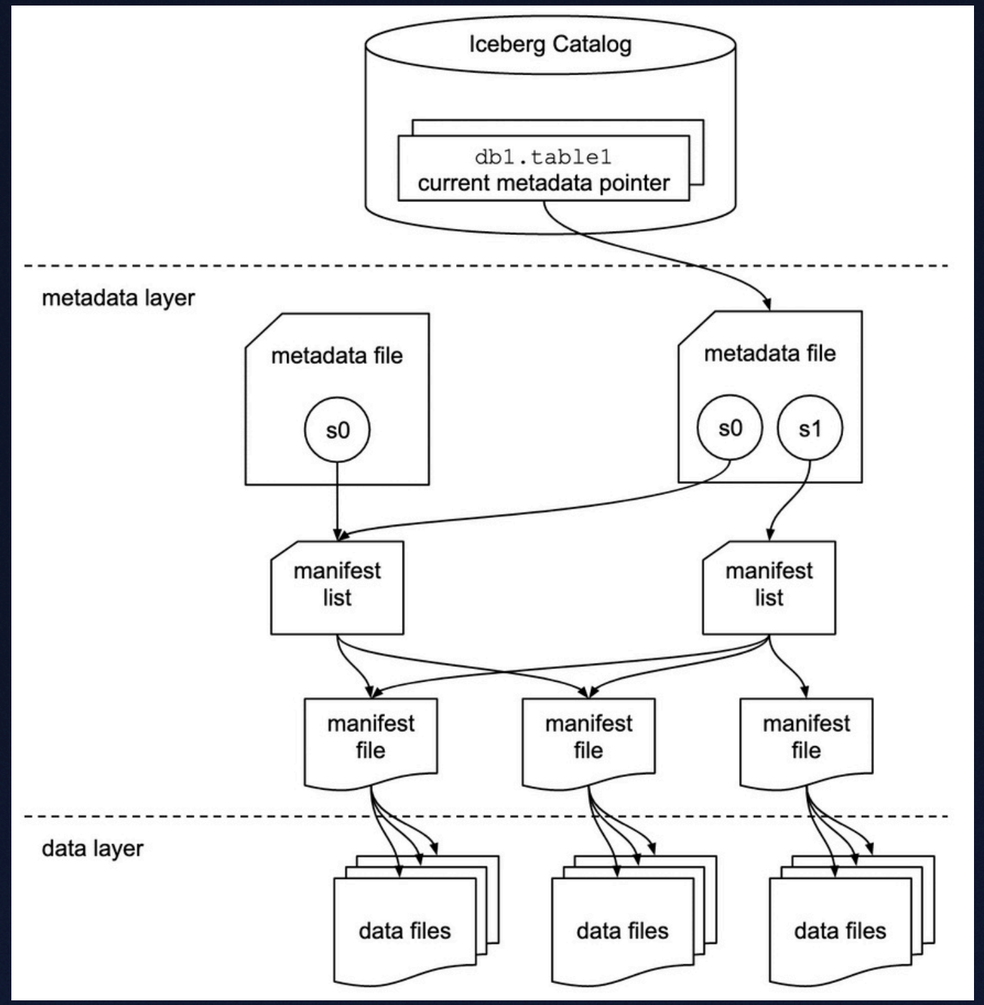
Hive Table Format



- Organized into a directory tree
- Additional metastore to track partitions
- Pros:
  - Simplicity, wide adoption, de-facto standard
  - File-format agnostic, Parquet, ORC, etc
- Cons:
  - Too much directory listing
  - State in Metastore and FS
  - Atomicity only at partition, no atomic file+metastore writes
  - Poor concurrent writer support

Iceberg

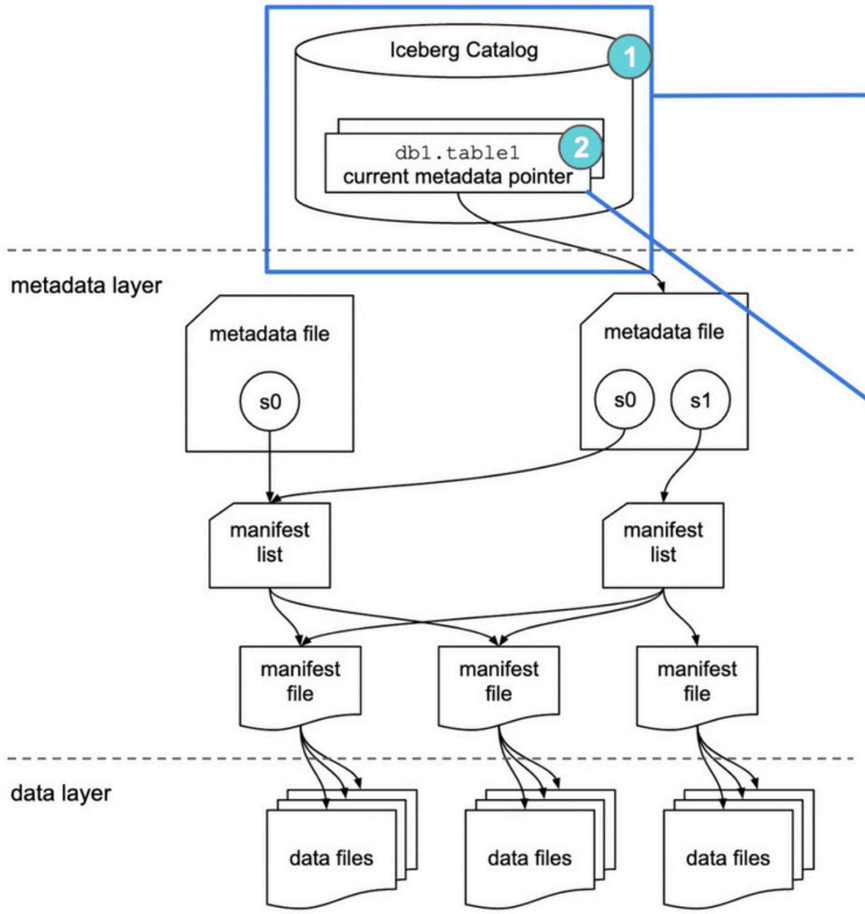
- Created in 2017 by engineers at Netflix
- Developed to address limitations with Hive
  - Lack of atomic transactions
  - File granularity of operations
  - Schema evolution
- Supports time travel
- Logical vs physical partitioning abstraction



Apache Iceberg

- Key Insight: Tracks all files in a table over time
- Snapshots contain complete list of files in table
- Writes commit and produce new snapshot
- Readers use current snapshot, Writers use OCC to create new snapshots and commit with serializable isolation.

Apache Iceberg



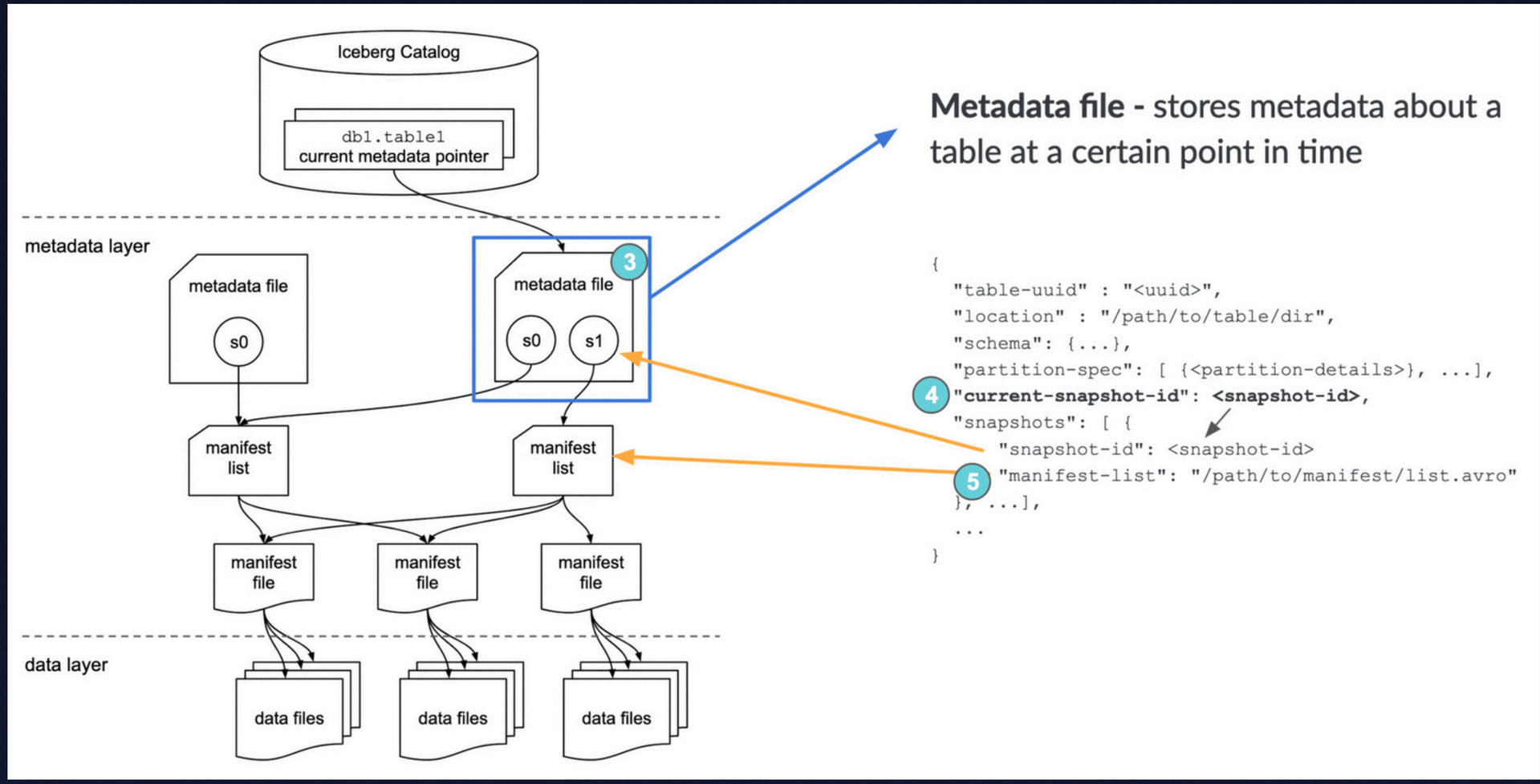
Iceberg Catalog

- A store that houses the current metadata pointer for Iceberg tables
- Must support atomic operations for updating the current metadata pointer (e.g. HDFS, HMS, Nessie)

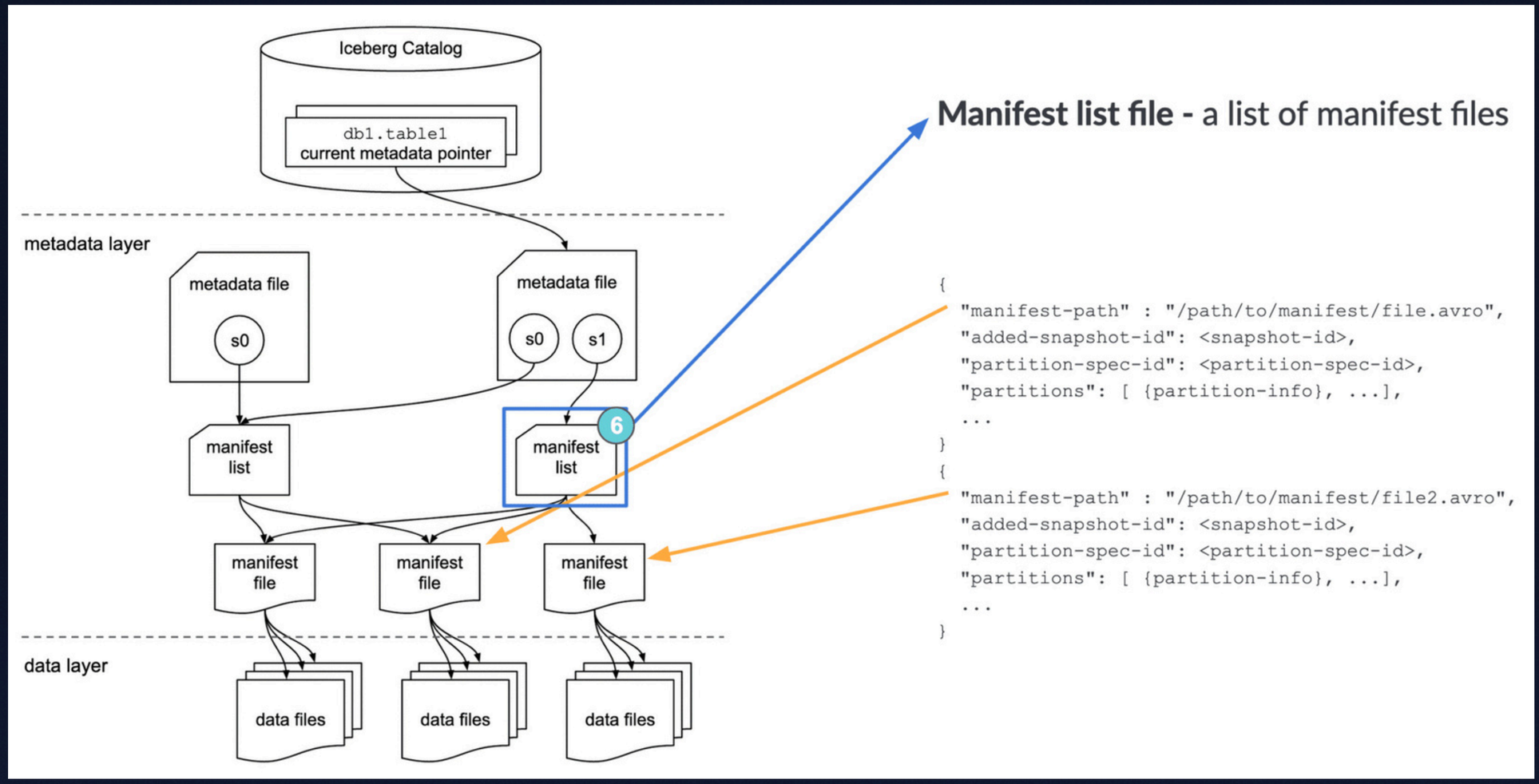
`table1`'s current metadata pointer

- Mapping of table name to the location of current metadata file

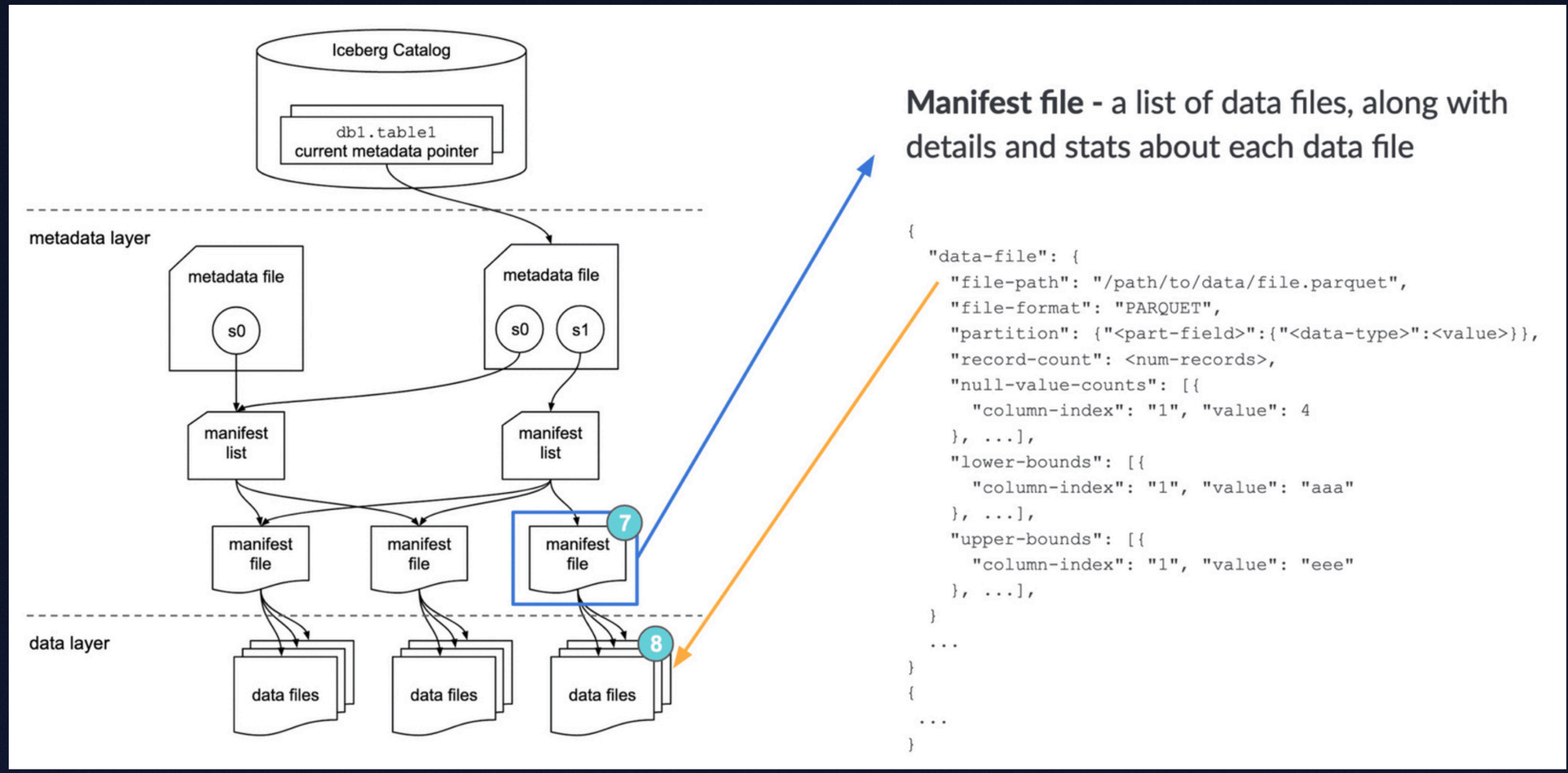
Apache Iceberg



Apache Iceberg



Apache Iceberg



Demo

Next: Future challenges

Storage I/O is fast

Traditional block-based compression methods restrict decoding speeds due to heavy CPU dependency.

Leverage the GPU

Variable run-length encoding and poor file layout can limit vector processing speed ups.

Metadata improvements

We need centralized metadata storage designed for cloud object storage and tuned for telemetry data.

Zeng, Xinyu, Yulong Hui, Jiahong Shen, Andrew Pavlo, Wes McKinney, and Huanchen Zhang. “An Empirical Evaluation of Columnar Storage Formats.” Proceedings of the VLDB Endowment 17, no. 2 (October 2023): 148–61. https://doi.org/10.14778/3626292.3626298.

Kuschewski, Maximilian, David Sauerwein, Adnan Alhomssi, and Viktor Leis. “BtrBlocks: Efficient Columnar Compression for Data Lakes.” Proceedings of the ACM on Management of Data 1, no. 2 (June 13, 2023): 1–26. https://doi.org/10.1145/3589263.

Thanks!

Mike Heffner

Ray Jenkins

streamfold.com

Fin