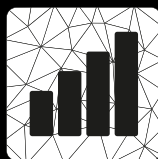




Conference Partners

Data Point Prague 2025



**Data
Brothers**



Microsoft



BYTECA

GORDON & WEBSTER



CONSULTING AND INVESTMENT INC.



**DATA
TALK**

Komunita
datových
profesionálů

Dive into the **DELTA LAKE**

Johan Ludvig Brattås
Director, Deloitte AI&Data

Agenda

- What are lakehouse files?
- The anatomy of Delta Lake
- Parquet 101
- Loading and querying data with Delta Lake
- Optimizing Delta Lake

What is lakehouse format?

- Wrapper around binary datafiles – most often parquet
- Provides RDBMS-like functionality for data lake
- All 4 formats are OSS
- Delta Lake originated at Databricks, and now Microsoft are all in
- Iceberg originated at Netflix. Snowflake are all in
- Hudi originated at Uber. Not gained much traction
- Paimon originated at Alibaba. Mostly used in Chinese companies



Lakehouse format

- Metadata stored in metastores

- Unity Catalog
- Hive
- Polaris (new)
- Internal to Fabric

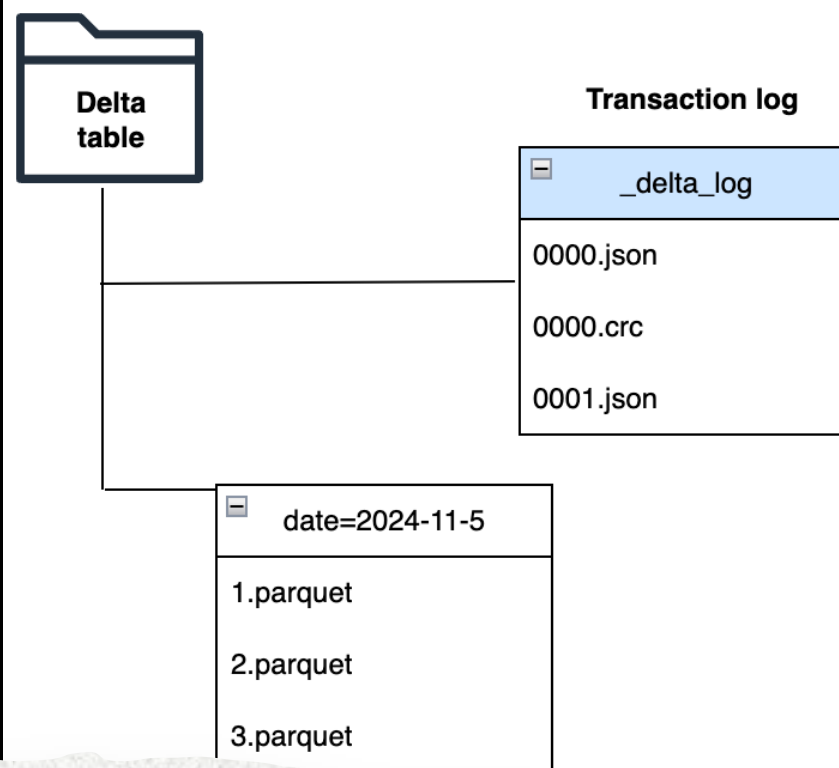
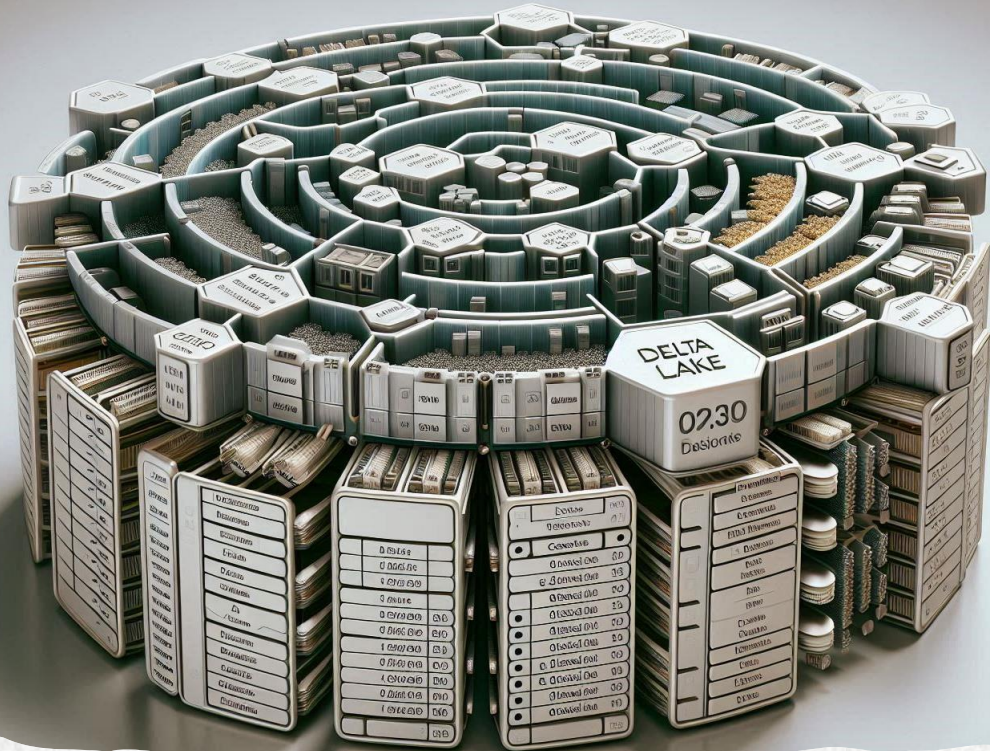
- Transaction logs

- Various mechanisms for optimizations, indexing, partitioning etc
(this is what we will look at today)



But Microsoft says Delta-Parquet?

- Microsoft marketing...
- Fabric actually uses  **DELTA LAKE**
- Though they add a custom V-Order at write time (more on that later).



Anatomy of the Delta Lake

- Data in Parquet files
- Transaction logs
- Metadata
- Schema
- Checkpoints

Parquet

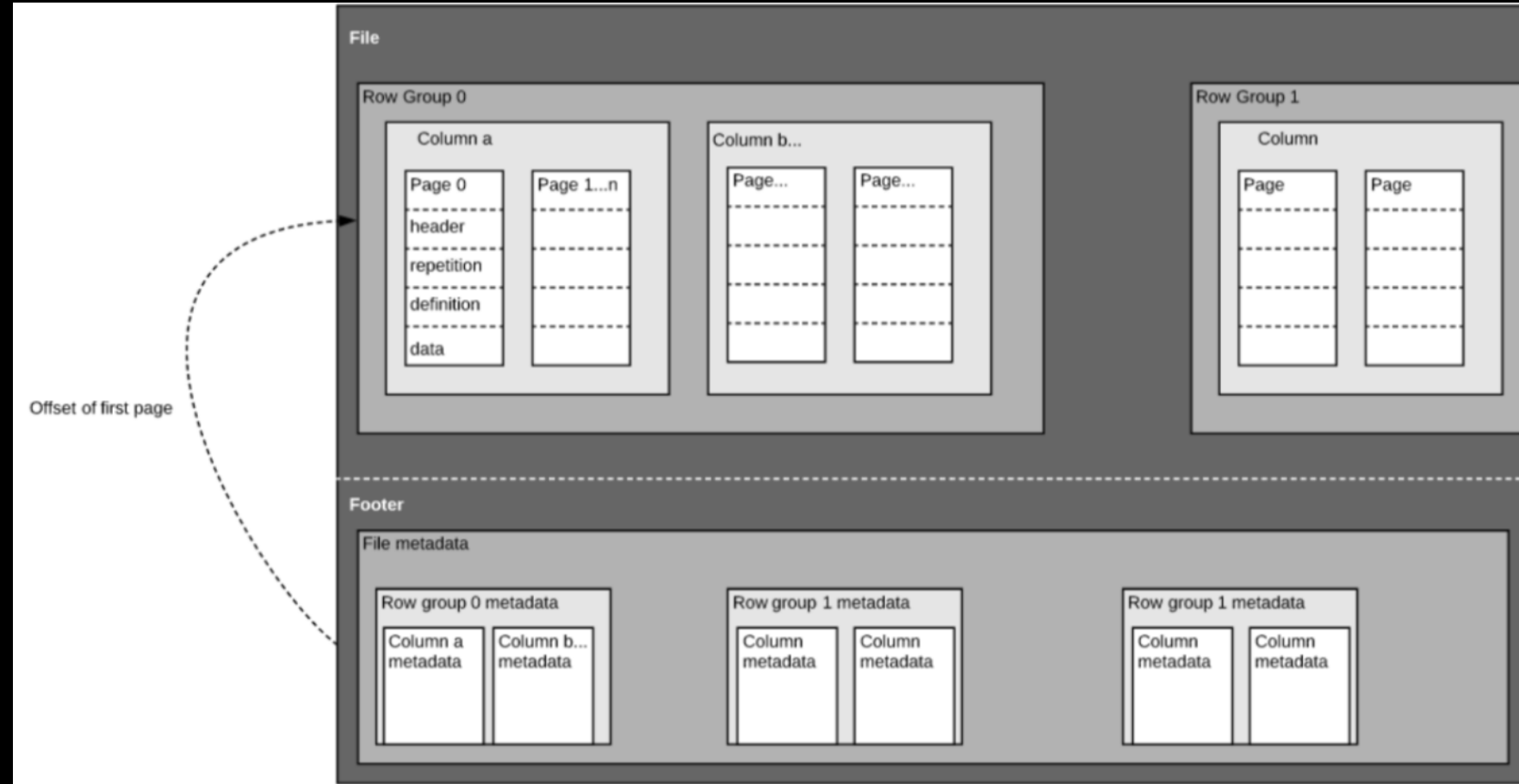
- Columnar
- Self-describing
- Optimized for read performance
i.e. Write Once, Read Many
- Compressed
 - Supports 7 different algorithms
 - Default Snappy



Parquet

Self describing?

- Header
 - Row groups
 - Columns
 - Pages
- Footer
 - Metadata
 - Row groups
 - Columns
- Optimal row group size 1GB



Transaction log

Also called Delta Log

- Series of JSON files
- Each transaction stored in new JSON file
 - Type of operation
 - Files added or removed
 - Schema at time of transaction
- Each transaction also has a CRC file
 - Key statistics of table version



Metadata

- Stored in the transaction log
- Contains information on schema, partitioning and configuration
- Important for managing and optimizing tables
- Can be queried with SQL, Python & Rust



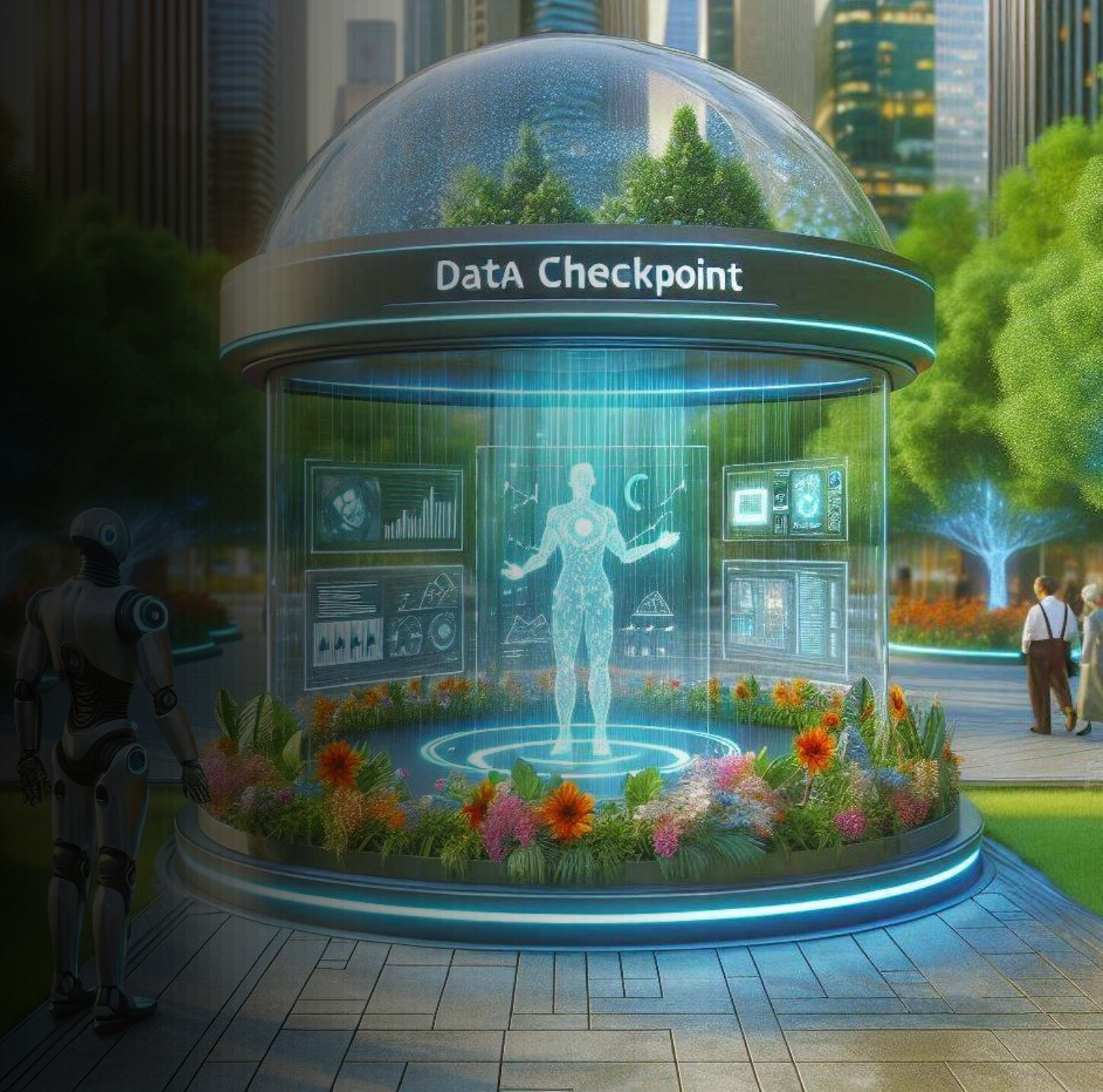
Schema

- Enforces Schema on Write
- Allows for schema evolution
- As expected – defines the table structure



Checkpoints

- Periodic snapshots of transaction log
- Speeds up read- and recovery
- Checkpoint created automatically every 10 transactions
- Also created during optimization
- Stored as parquet files





Demo

Working with DELTA LAKE

- As mentioned, you can use SQL or several other languages
- CREATE
- INSERT
- SELECT
- UPDATE
- DELETE
- MERGE

Maintaining your delta lake

- Table properties
- Optimizing
- Table management
- Repairing



Table properties

- TBLPROPERTIES stored with metadata
- Enables automatic maintenance
 - Cleaning
 - Tuning
 - Repairing
- Enables better control over tables

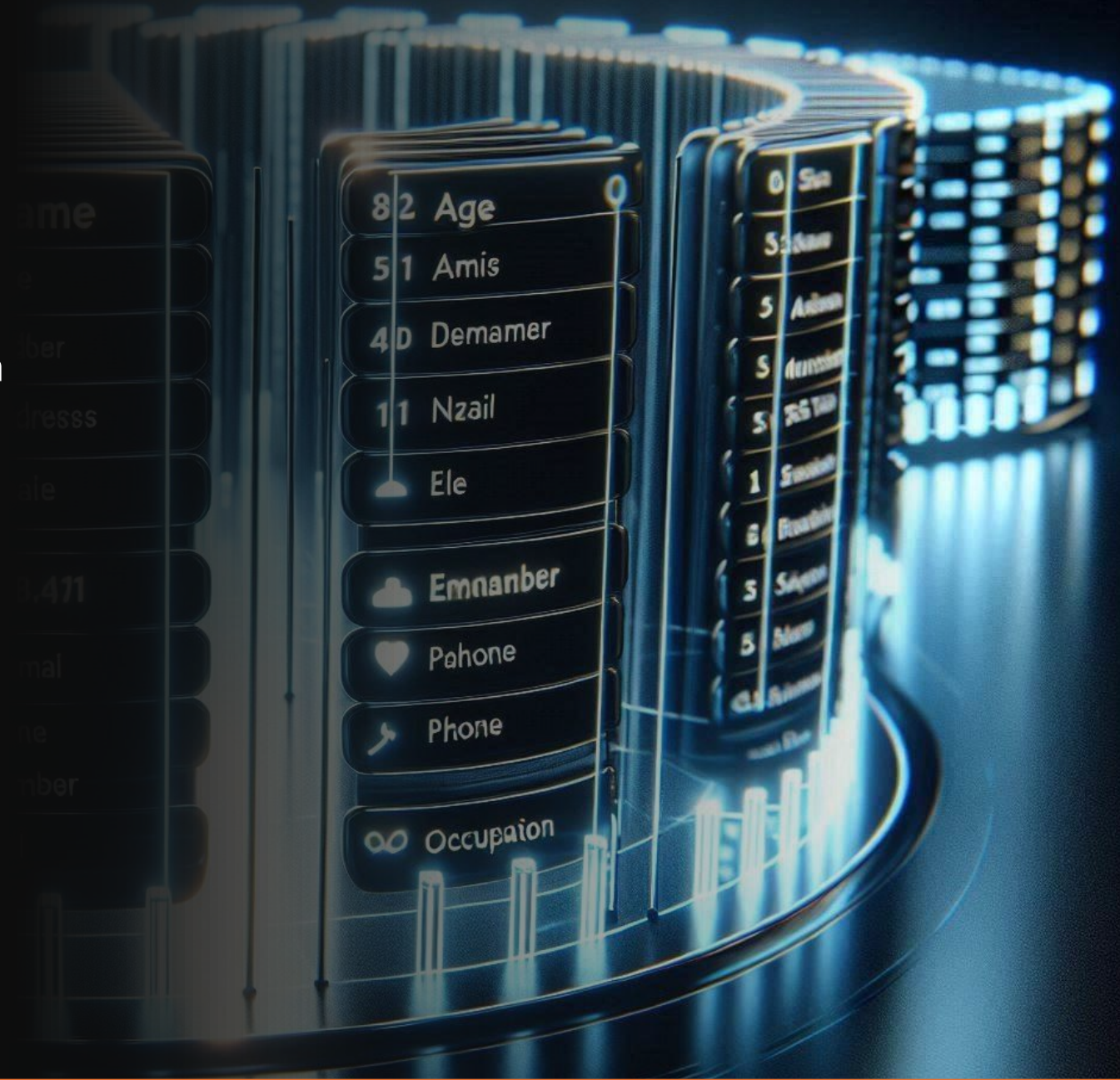


Table properties

Property	Data type	Use with	Default
<i>delta.logRetentionDuration</i>	CalendarInterval	Cleaning	<i>interval 30 days</i>
<i>delta.deletedFileRetentionDuration</i>	CalendarInterval	Cleaning	Interval 1 week
<i>delta.setTransactionRetentionDuration</i>	CalendarInterval	<i>Cleaning, Repairing</i>	none
<i>delta.targetFileSize*</i>	String	Tuning	none
<i>delta.tuneFileSizesForRewrites*</i>	Boolean	Tuning	none
<i>delta.autoOptimize.optimizeWrite*</i>	Boolean	Tuning	none
<i>delta.autoOptimize.autoCompact</i>	Boolean	Tuning	none
<i>delta.dataSkippingNumIndexedCols</i>	Int	Tuning	32
<i>delta.checkpoint.writeStatsAsStruct</i>	Boolean	Tuning	none
<i>delta.checkpoint.writeStatsAsJson</i>	Boolean	Tuning	True
<i>delta.randomizeFilePrefixes</i>	Boolean	Tuning	False

* Exclusive to Databricks

Optimize

- The small file problem
- Performance hit
- OPTIMIZE
- Z-ORDER BY
- Table properties can be used to help
- File compression algorithm change?





V-ORDER (only Fabric)

- Write-time optimization
- Applies special
 - Sorting
 - Row group distribution
 - Dictionary encoding
 - Compression
- Requires less disk, CPU and bandwidth on read
- 15% more time on write
- 50% better compression
- Applied on parquet files so compatible with Delta level functions such as Z-Order

VACUUM

- Dead files caused by OPTIMIZE
- VACUUM helps clean up dead files
- Uses transaction log
- `deletedFileRetentionDuration` property sets boundaries
- VACUUM USING INVENTORY
- VACUUM LITE



Table Management

- Partitioning
- Don't use if table < 1 TB
- Choose the right column
 - Low cardinality
 - Partition files > 1GB
- Partition at table creation
- Partitioning existing table forces rewrite



Liquid Clustering

- Similar to partitioning – but also not...
- Replaces Z-ORDER and partitioning
- Can be implemented after table creation
- Better at full scans, or across multiple partitions
- Use on medium sized tables or with high cardinality columns



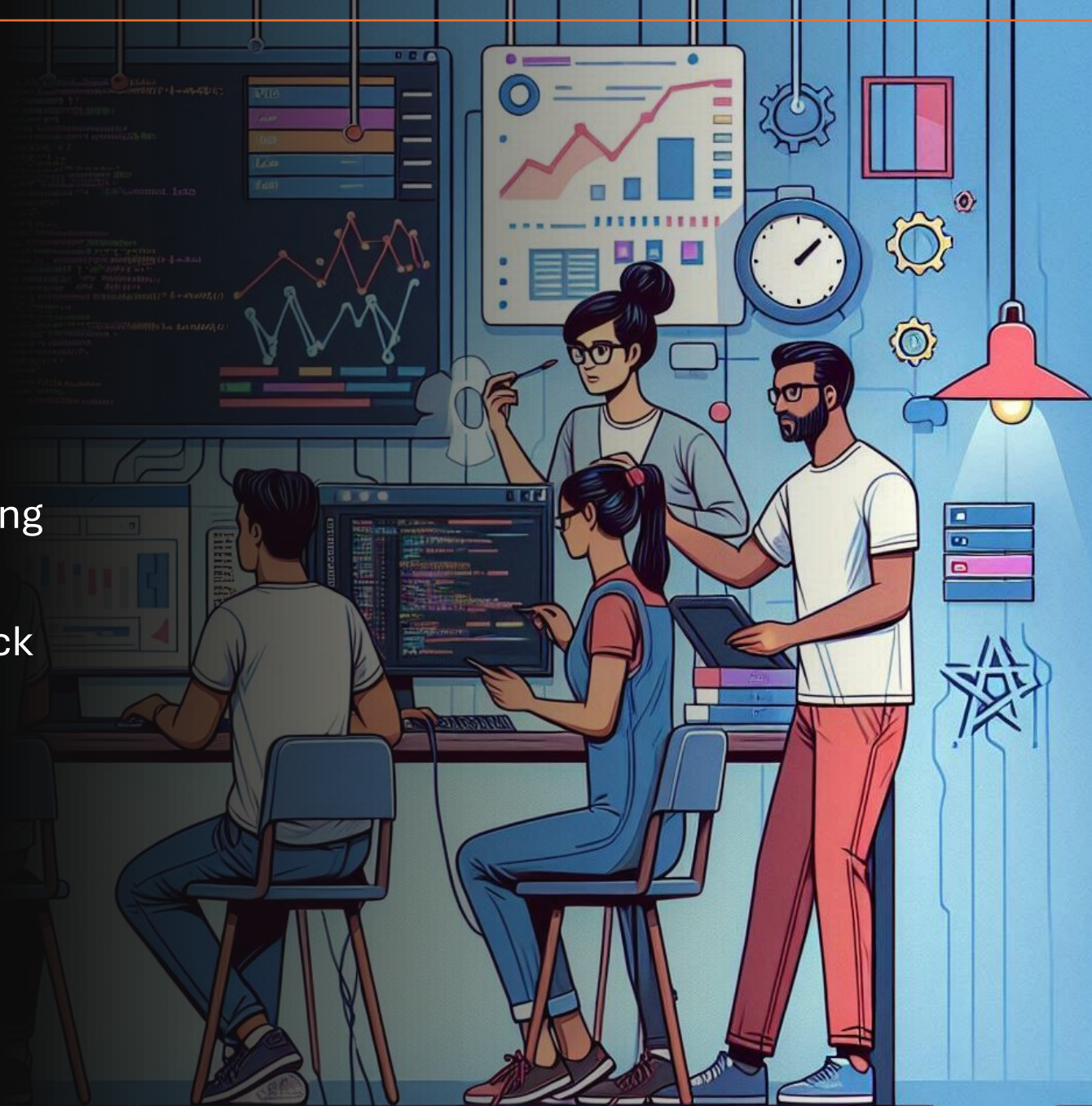


Worth mentioning?

- Fabric Lakehouse supports Auto Optimization
- Databricks has Predictive Optimization (not in all regions yet)
- Works well for smaller files and daily operations – for Liquid Clustering tables
- Worth using in combination with regular OPTIMIZE – especially if partitioned

Repairing

- Recover data with REPLACEWHERE
 - Replace missing or overwrite existing
- Delete data or remove partitions
 - Timetravel can help us get data back
- Dropping table
 - No undo button
- Keeping a clean house – or table
 - VACUUM to remove deleted files







Johan Ludvig Brattås

Director, Deloitte



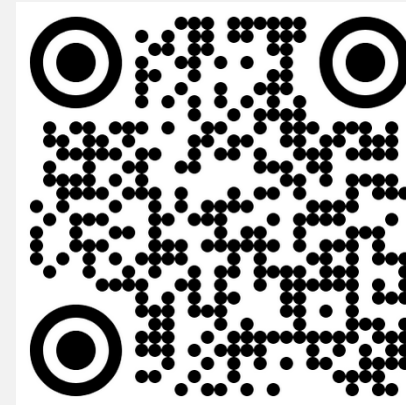
/johanludvig



@intoleranse



jbrattas@deloitte.com



GitHub

Chronic volunteer

Co-organizer – DataSaturday Oslo

President – MDPUG Oslo

Frequent volunteer in general

When not geeking out over new tech

Teaching coeliacs how to bake gluten free

Baking

Hiking

Gardening