#FABCONSQLCON2026

FABCON
Microsoft Fabric
COMMUNITY CONFERENCE

SQLCON
Microsoft SQL
COMMUNITY CONFERENCE

ATLANTA MARCH 16 - 20, 2026

Demystifying Spark Profile Optimizations in Microsoft Fabric

Thibauld CROONENBORGHS, AE NV, Belgium

What are Spark profiles?

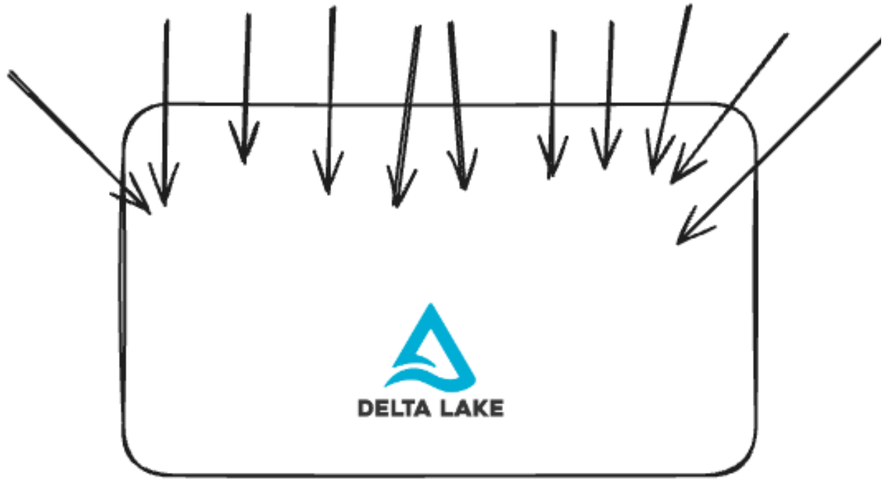
What are Spark resource profiles?

- Collection of Spark configurations, bundling settings & optimizations
- Applying tuning best practices for your specific workload
- 3 default workloads
- Custom workloads can be defined

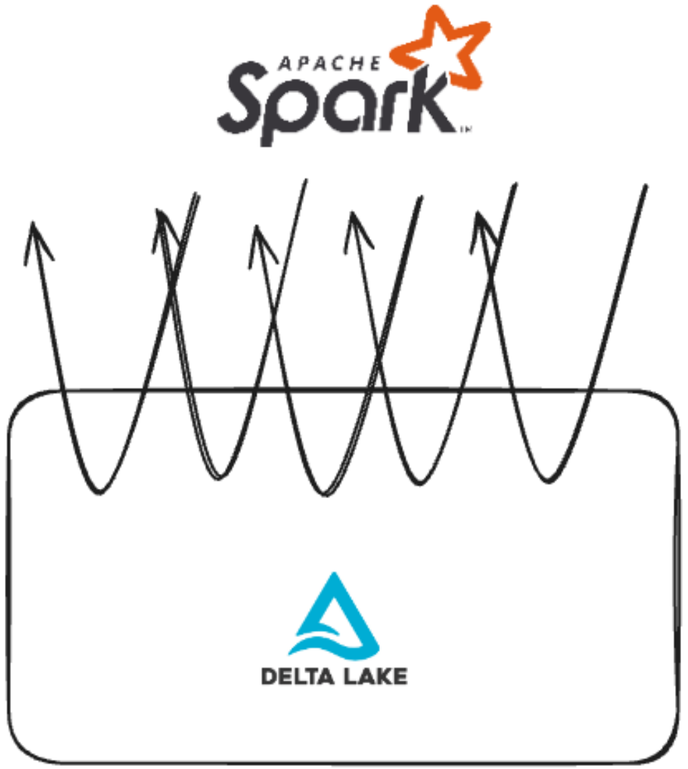
Default on new workspaces = **writeHeavy**

```
spark.conf.set("spark.fabric.resourceProfile", "writeHeavy")
```

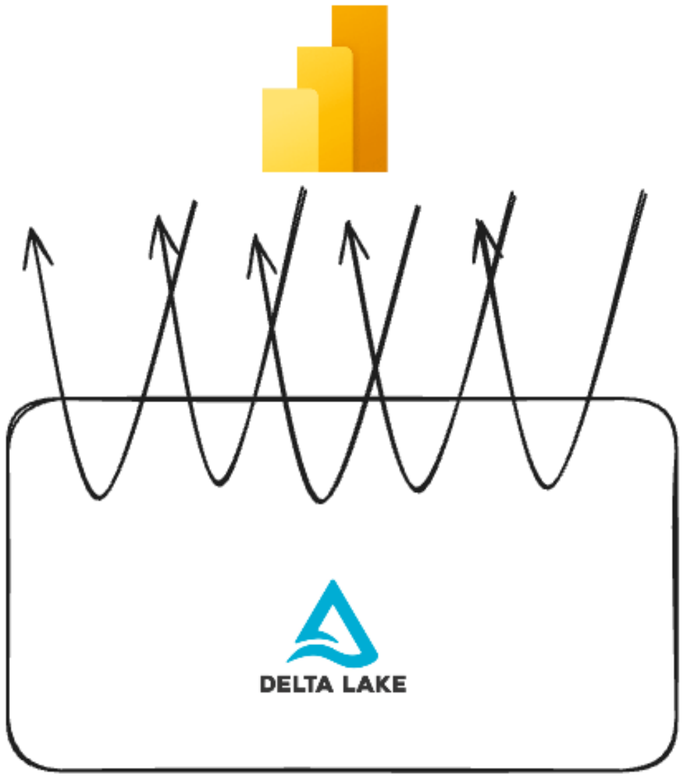
Which Spark profiles are available?



writeHeavy



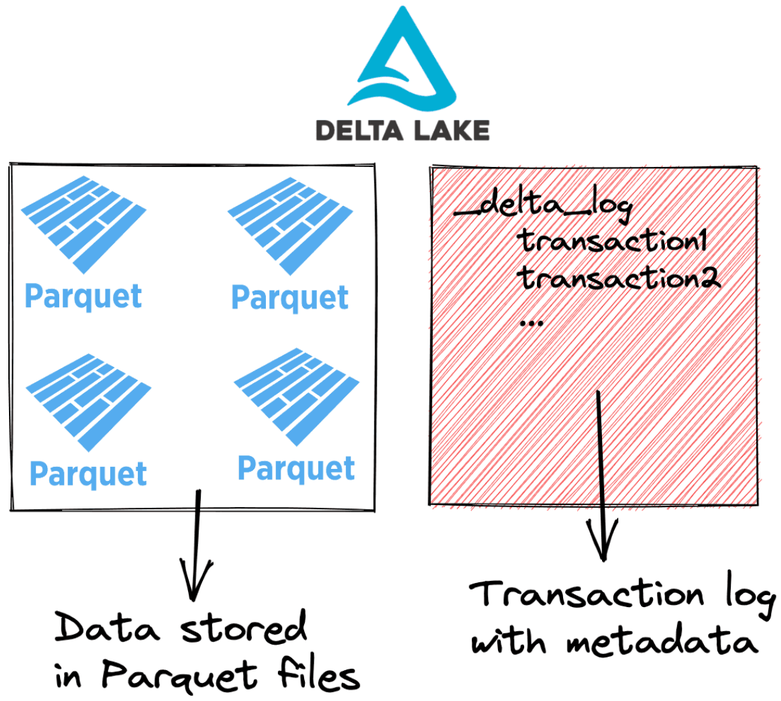
readHeavyForSpark

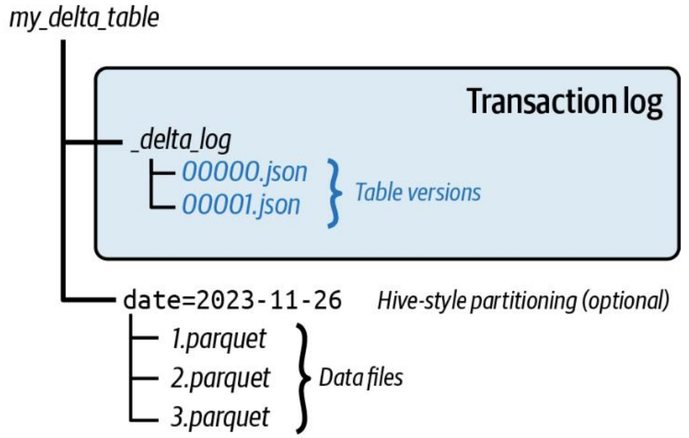


readHeavyForPBI

Taking a (small) step back: lakehouses and Parquet file format

Underneath a lakehouse (table)





Row oriented storage

VendorID	tpep_pickup	passenger_count	trip_distance	total_amount
1	2018-01-01 00:21:05	2	2.7	15.8
1	2020-06-07 10:20:50	4	0.5	5.8
1	2021-12-13 08:12:00	1	0.8	8.3
2	2021-08-30 05:50:00	3	10.3	34.3
2	2022-08-15 06:50:00	1	2.5	16.55
2	2022-06-29 13:20:00	4	4.7	24.65

Credit: https://data-mozart.com/parquet-file-format-everything-you-need-to-know/

The Parquet storage format: column-oriented

VendorID	tpep_pickup	passenger_count	trip_distance	total_amount
1	2018-01-01 00:21:05	2	2.7	15.8
1	2020-06-07 10:20:50	4	0.5	5.8
1	2021-12-13 08:12:00	1	0.8	8.3
2	2021-08-30 05:50:00	3	10.3	34.3
2	2022-08-15 06:50:00	1	2.5	16.55
2	2022-06-29 13:20:00	4	4.7	24.65

- Column-oriented (vs row-oriented)
- Compressed
- Enables column skipping

```
SELECT VendorId, trip_distance FROM dbo.taxi
```

The Parquet storage format: row groups

- Row groups as additional structure
- Extra metadata contained in those row groups

```
SELECT VendorId, trip_distance FROM dbo.taxi
WHERE VendorID = 1
```

Good data layout can have a huge impact on the performance of your queries.

What optimizations are included in the profiles?

V-Order

- **Write-time** optimization at the Parquet file level
- **Vertipaq (Power BI)-like** storage in Parquet files, achieving in-memory performance
- **Fabric specific** feature, but 100% compatible with open-source Parquet
- *Lightening-fast reads*, but hit during write performance
- ⚠ **disabled by default** in new Fabric workspaces (to favor ingestion speed)

What does V-Order achieve on the Parquet level?

- **Special sorting**: physical ordering on write based on one or multiple columns
- **Row-group distribution**: Distributing rows in a way that minimizes read times.
- **Dictionary encoding**: values are replaced with shorter, unique codes (Vertipaq implementation)
- **Run-length encoding**: consecutive repeated values with a single value and a count of its repetitions

Result: **Better compressed** Parquet files with clustered values which improves **data skipping** and **cache locality**

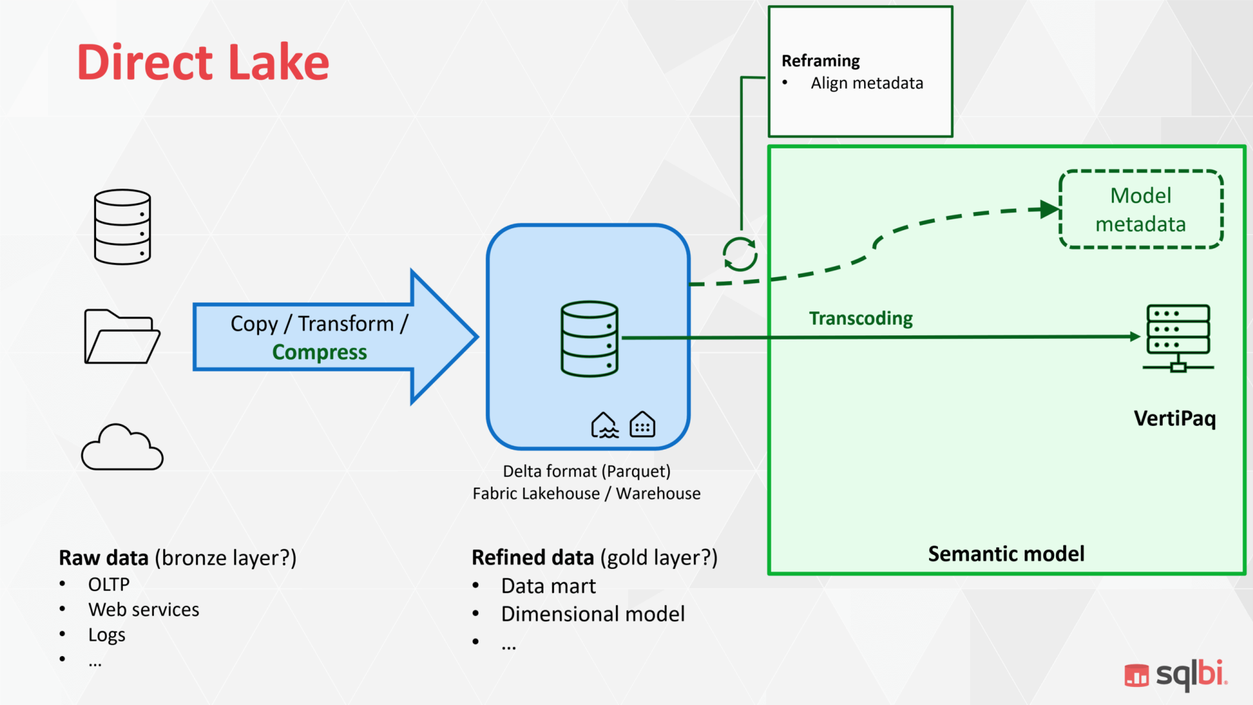
V-Order comparison on taxi data

	V-Order	Non V-Order
Write time (workspace default Spark cluster)	13min44s	8min50s
Total # Parquet files	225	225
Total # row groups	296	225
Total file size	12767.75 MB	24050.96 MB
Read time (heavy query)	58s	1min11s

46% size reduction!

962695813 rows

How V-Order benefits Direct Lake



- Loading data from OneLake when queried (*transcoding*)
- Helps Parquet files become more "*Vertipaq-ready*"
- Better compression = **fast data loading**

Credit: https://www.sqlbi.com/blog/marco/2025/05/13/direct-lake-vs-import-vs-direct-lakeimport-fabric-semantic-models-may-2025

When to use V-Order

- **Direct Lake**
  - Valuing query performance over data timeliness
- Extensive use of **Fabric warehouse or SQL endpoint**
  - Read heavy analytical tables
  - Valuing query performance over data timeliness
- **Import** mode
  - Valuing query performance over data timeliness
  - Small read performance

Conclusion: V-Order is not just about file size. It **improves how data is physically laid out** and how **metadata** can be used for pruning.

Small file problem

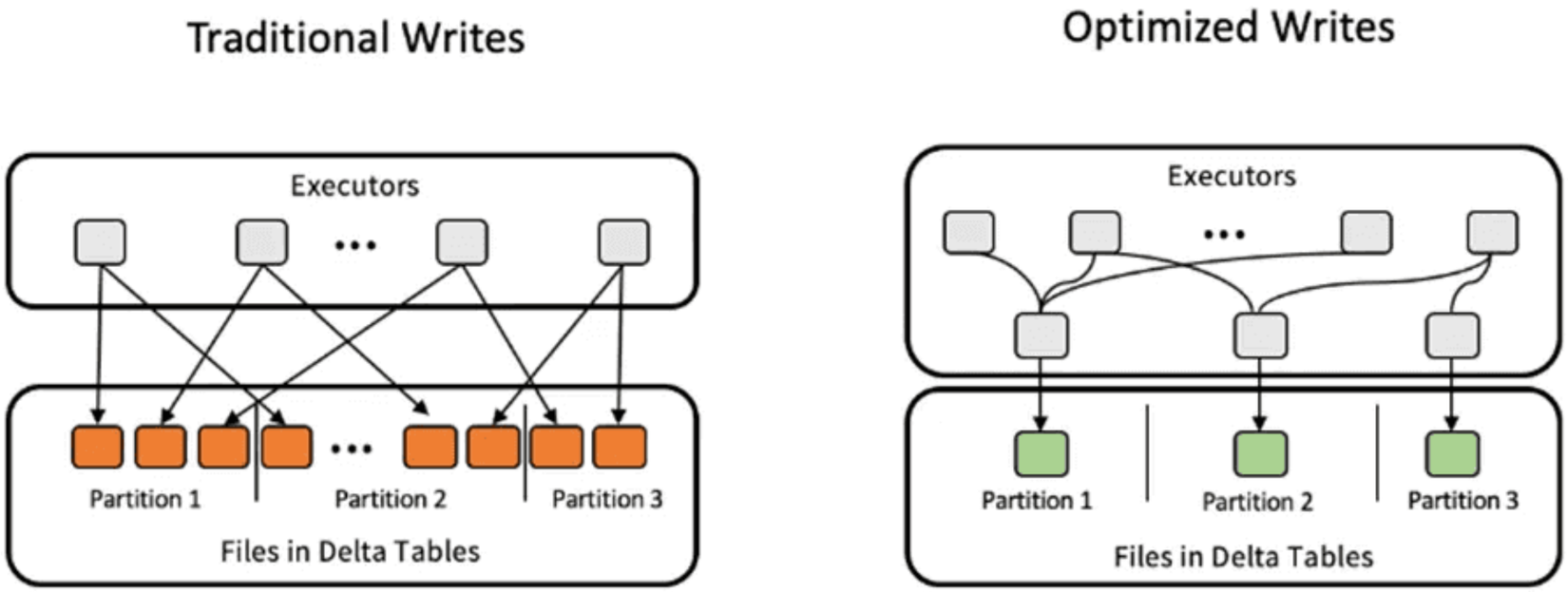
- Small data files can be caused by:
  - **User error:** for example, when repartitioning datasets
  - **Partitioning:** Partitioning a table on a high-cardinality column
  - **Frequent incremental updates:** Tables with frequent, small updates

- More small files = more problems
  - Metadata & processing overhead
  - Inefficient I/O
  - Slow query planning

Optimized write

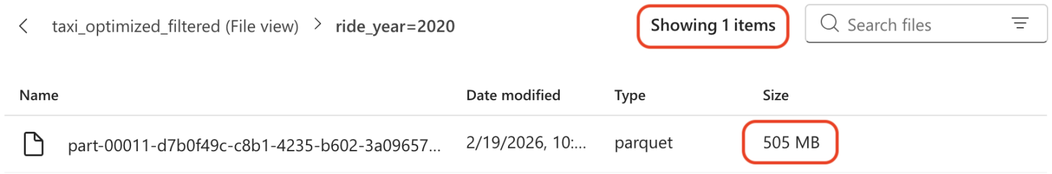
- File size, the number of files, #Spark workers and its configurations => impact on **performance**
- **Reduces # files** written
- Aims to **increase file size** (default 128Mb)
- Read performance & resource usage improvement, but hit during write performance (extra shuffle)
- Works best on **partitioned tables**
- Delta Lake specific feature (so not Fabric specific)

How optimized write works

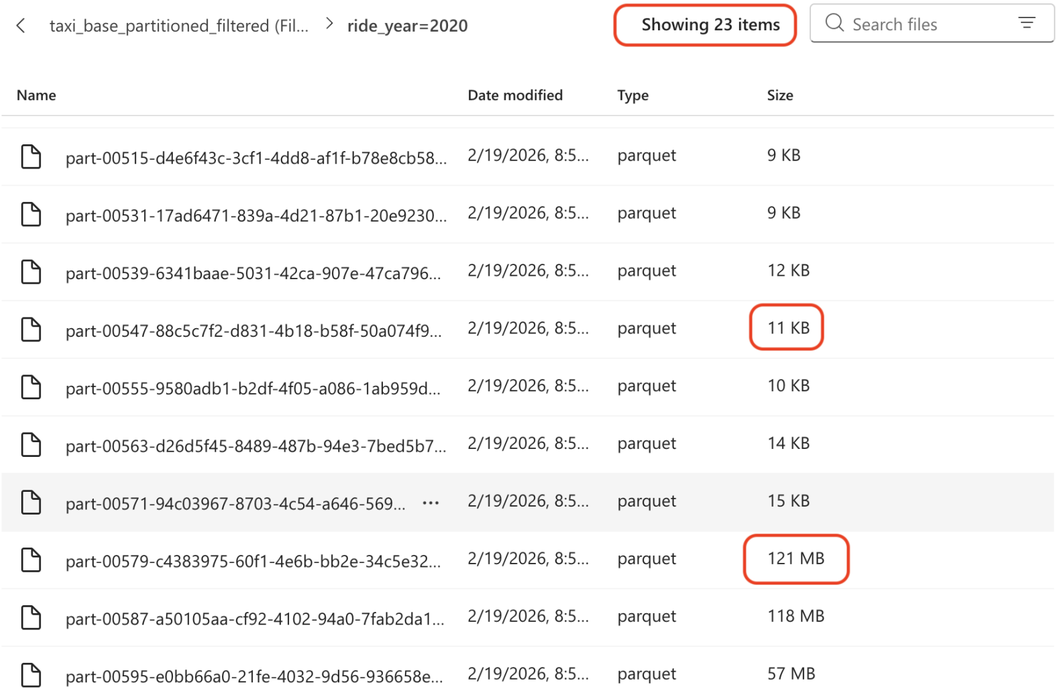


https://delta.io/blog/delta-lake-optimize/

Optimized write



Non optimized write



When to use optimized write?

- Partitioned tables with write patterns with **suboptimal file size** (batch write, repartitioning, streaming)
  - Improves read performance or resource usage, also for Spark

- **Direct Lake**
  - Valuing query performance over data timeliness

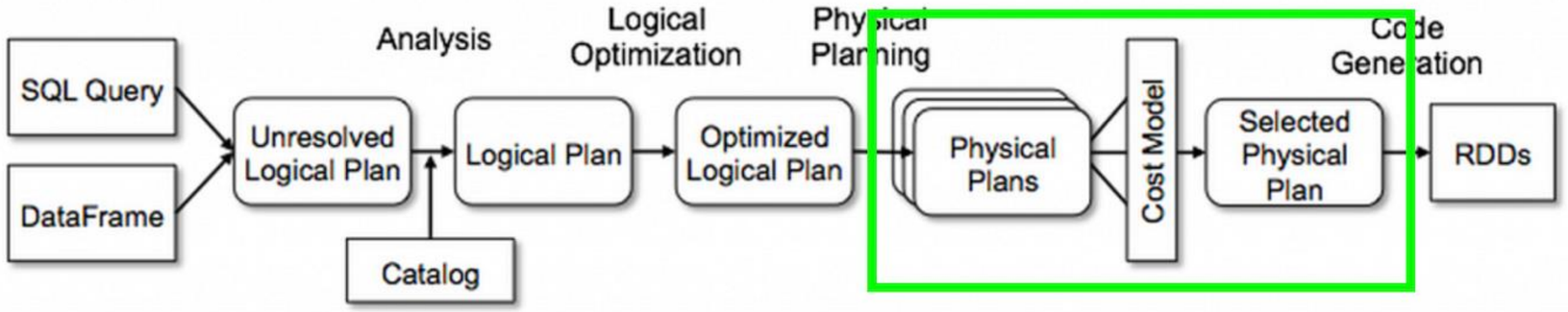
Other table-level optimizations

What are automated table statistics?

- Automatically collect detailed **table-level metrics**
- Auto-compute set of extended stats for the first 32 columns of the table
- Improved **query planning** for joins, filters, aggregations, and partition pruning
- Output in separate **metadata** file (Parquet format,*_stats* folder in *_delta_log)*
- No more *ANALYZE TABLE COMPUTE*
- Stats are collected **only at write time and not continuously updated**

How automated table statistics work

- Column stats calculated upon write (NULL count per column, DISTINCT count, etc.)
- Injected into Spark's cost-based query optimizer
- Purpose: Inform Spark's cost-based optimizer for choosing best strategies
  - Partition pruning
  - Choosing join strategy
  - Optimize aggregations
  - Etc.



Currently only leveraged by Spark

Z-Order

- Read-time optimization at the Parquet level, often combined with partitioning
- **Open-source Delta Lake feature** (not Fabric-specific)
- **Co-locating** related information in the same set of files based on specified columns
- Provides **data skipping**
- Applied during OPTIMIZE

```
OPTIMIZE table_name ZORDER BY col1, col2 VORDER
```

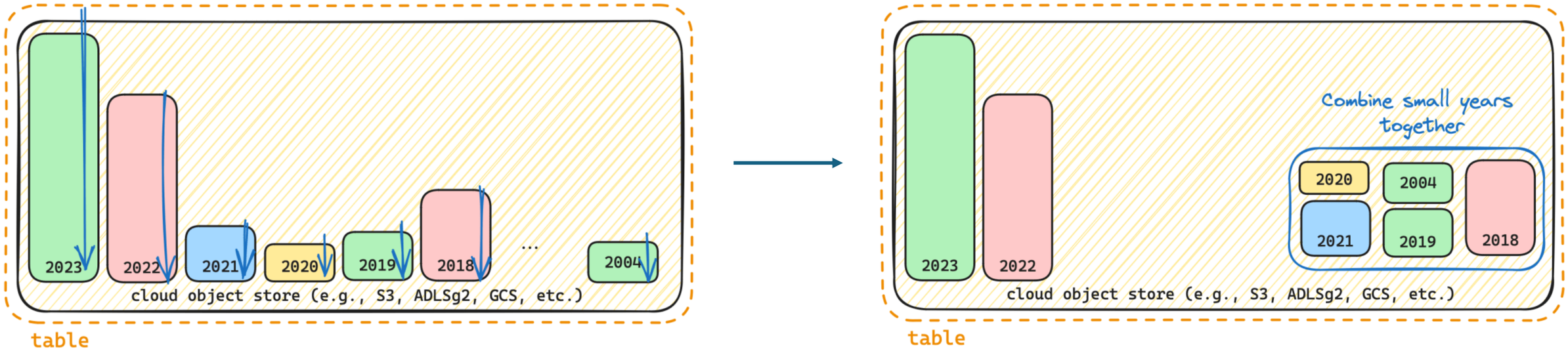
What is Liquid Clustering?

- **Flexible data layout** optimization technique for Delta tables
- **Replaces** traditional partitioning and Z-ordering with a **simpler, more adaptive** approach
- **Incremental clustering**: new data is clustered without having to recompute entire layout (making use of Delta Lake metadata)
- **Multidimensional data locality**: engine groups rows with similar clustering key values into same files
- **Open-source Delta Lake feature** (not Fabric-specific)

Result: **Better data skipping** with **lower maintenance** and **no partition rewriting**

How does Liquid Clustering work?

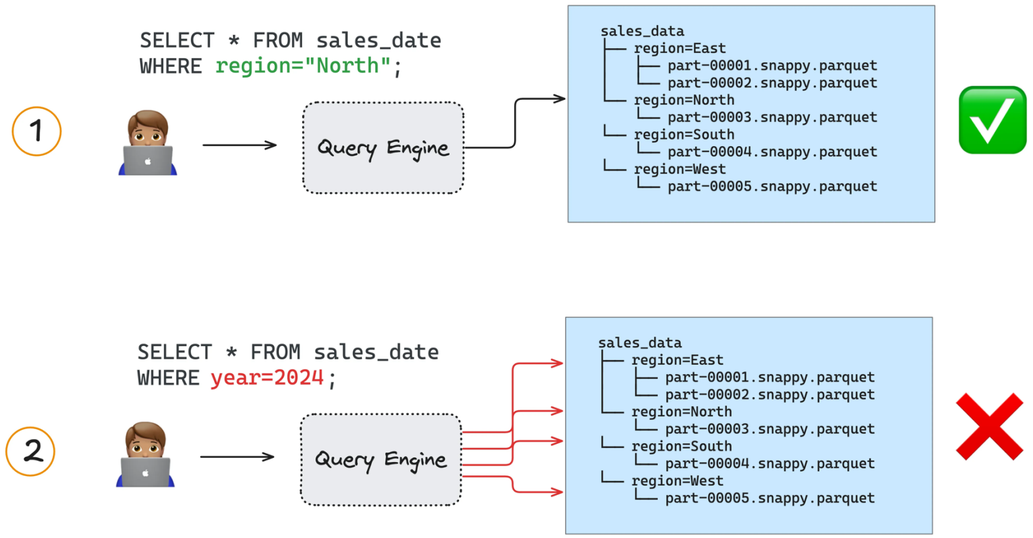
CREATE TABLE table_name CLUSTER BY col1, col2 VORDER



https://dennyglee.com/2024/02/06/how-delta-lake-liquid-clustering-conceptually-works/

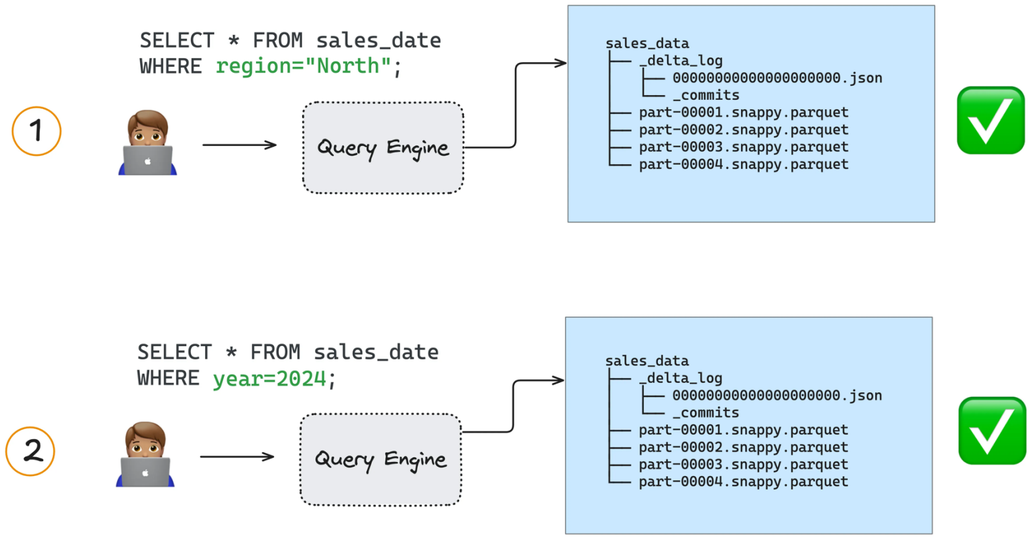
Z-order & partitioning vs Liquid Clustering

Partitioning



--> complete rewrite required 😰

Liquid Clustering



--> optimal data layout without rewrites 🙌

https://delta.io/blog/liquid-clustering

Partitioned Z-Ordering

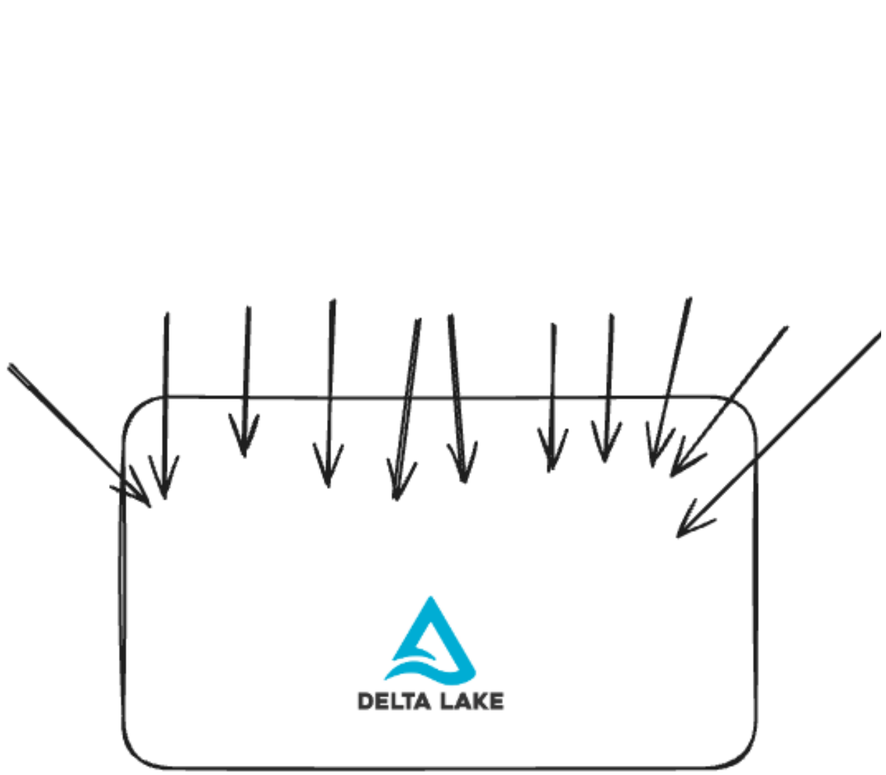
- Stable and dominant query patterns
- You have a very strong, low-cardinality partition key
- Max 1-2 extra non-partition columns on which to filter
- You have time for rewrites

Liquid clustering

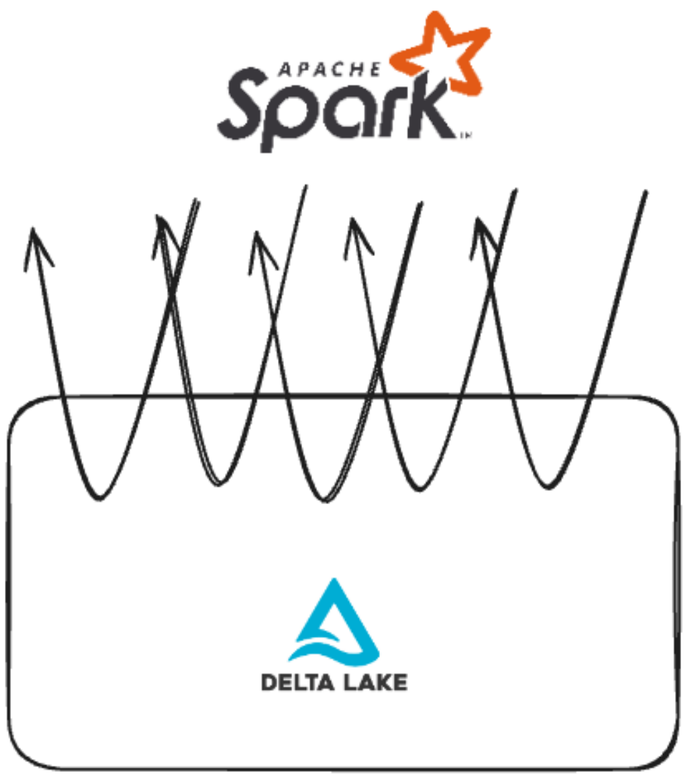
- **Flexibility** needed, query patterns may evolve over time
- Want to avoid eager clustering
- Skewed data, weird cardinalities, lots of new data is ingested
- **Strong default choice**
- **BUT no automatic recompute**

Fabric Spark Profiles (finally!)

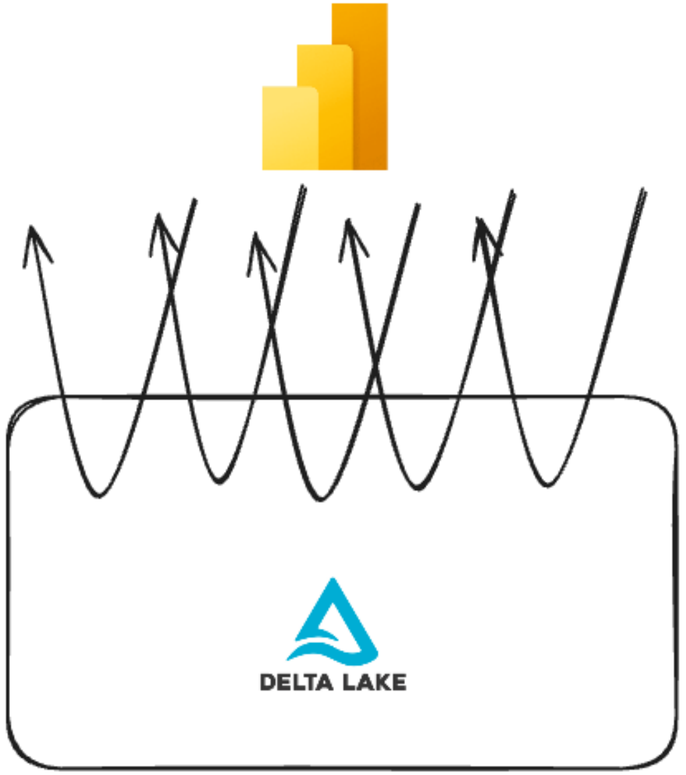
Which Spark profiles are available?



writeHeavy



readHeavyForSpark



readHeavyForPBI

writeHeavy (default)

```
{
        "spark.sql.parquet.vorder.default": "false",
         "spark.databricks.delta.optimizeWrite.enabled" : "null",
        "spark.databricks.delta.optimizeWrite.binSize": "128",
         "spark.databricks.delta.optimizeWrite.partitioned.enabled": "true"
}
```

OPTIMIZED WRITE 

V-ORDER 

OW PARTITIONED TABLE 

writeHeavy (default)

```
{
        "spark.sql.parquet.vorder.default": "false",
         "spark.databricks.delta.optimizeWrite.enabled" : "null",
        "spark.databricks.delta.optimizeWrite.binSize": "128",
         "spark.databricks.delta.optimizeWrite.partitioned.enabled": "true"
}
```

OPTIMIZED WRITE 

V-ORDER 

OW PARTITIONED TABLE 

readHeavyForSpark

```
{
        "spark.sql.parquet.vorder.default": "false",
         "spark.databricks.delta.optimizeWrite.enabled" : ”true",
        "spark.databricks.delta.optimizeWrite.binSize": "128",
         "spark.databricks.delta.optimizeWrite.partitioned.enabled": "true"
}
```

OPTIMIZED WRITE 

V-ORDER 

OW PARTITIONED TABLE 

readHeavyForPBI

```
{
        "spark.sql.parquet.vorder.default": ”true",
         "spark.databricks.delta.optimizeWrite.enabled" : ”true",
        "spark.databricks.delta.optimizeWrite.binSize": "1g"
}
```

OPTIMIZED WRITE 

V-ORDER 

OW PARTITIONED TABLE 

How to fit it into your medallion architecture?

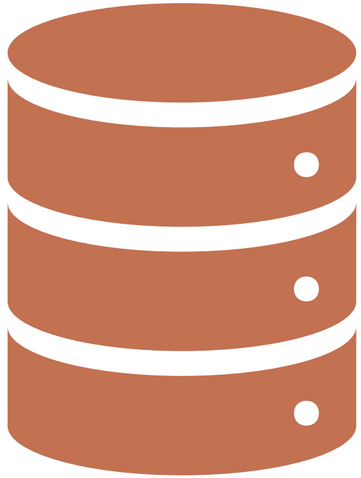
Bronze

LC/Z-ORDER ❌

OPTIMIZED WRITE ❌

V-ORDER ❌

writeHeavy



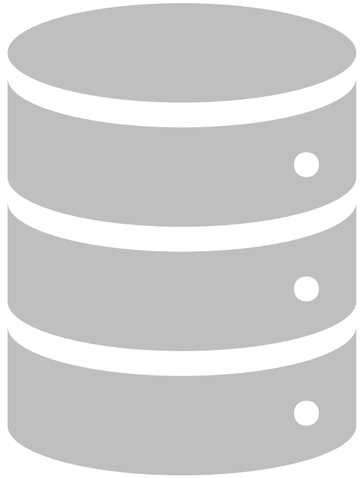
Bronze

LC/Z-ORDER  depends on use case!

OPTIMIZED WRITE  128-256Mb

V-ORDER 

readHeavyForSpark



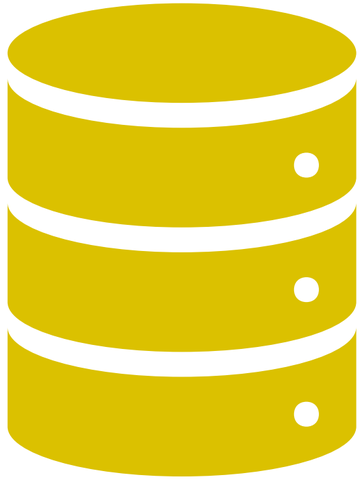
Gold

LC/Z-ORDER ✓

OPTIMIZED WRITE ✓ 400Mb-1Gb

V-ORDER ✓

readHeavyForPBI



Sound off.
The mic is all yours.
Influence the product roadmap.

Join the Fabric User Panel

Share your feedback directly with our Fabric product group and researchers.

https://aka.ms/JoinFabricUserPanel

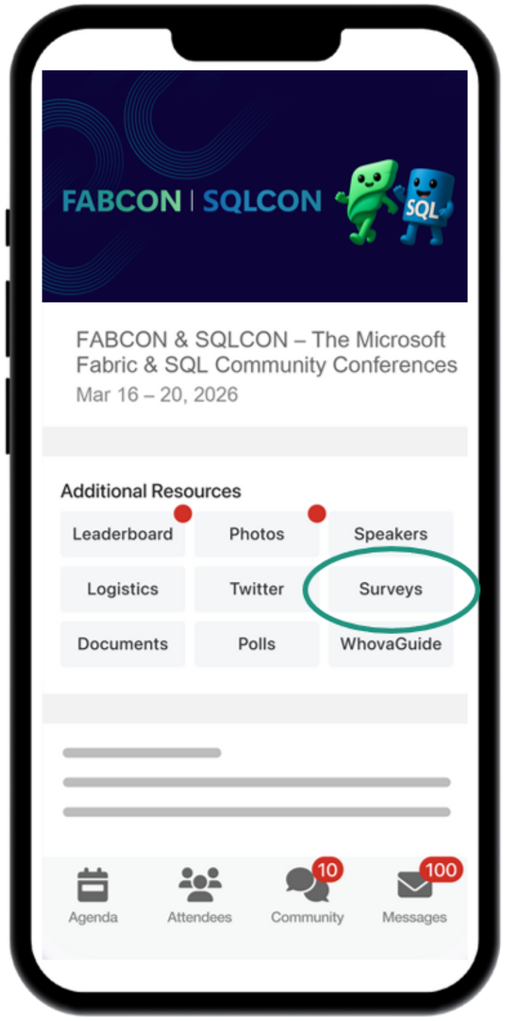
Join the SQL User Panel

Influence our SQL roadmap and ensure it meets your real-life needs

https://aka.ms/JoinSQLUserPanel

How was the session?

Complete Session Surveys in Whova for your chance to WIN PRIZES!



Get Two Fabric Certifications for FREE

Attendees of FABCON can take the Fabric Analytics Engineer or Fabric Data Engineer exam for free. Be part of the 2 fastest growing role-based certifications in Microsoft history.

Request your voucher by March 23, 2026.

https://aka.ms/fabcon/cert100