#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 resource profiles?

- Collection of Spark configurations, bundling settings & optimizations
- 3 **default** workloads
  - writeHeavy
  - readHeavyForSpark
  - readHeavyForPBI
- **Custom** workloads can be defined

Default on new workspaces = **writeHeavy**

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

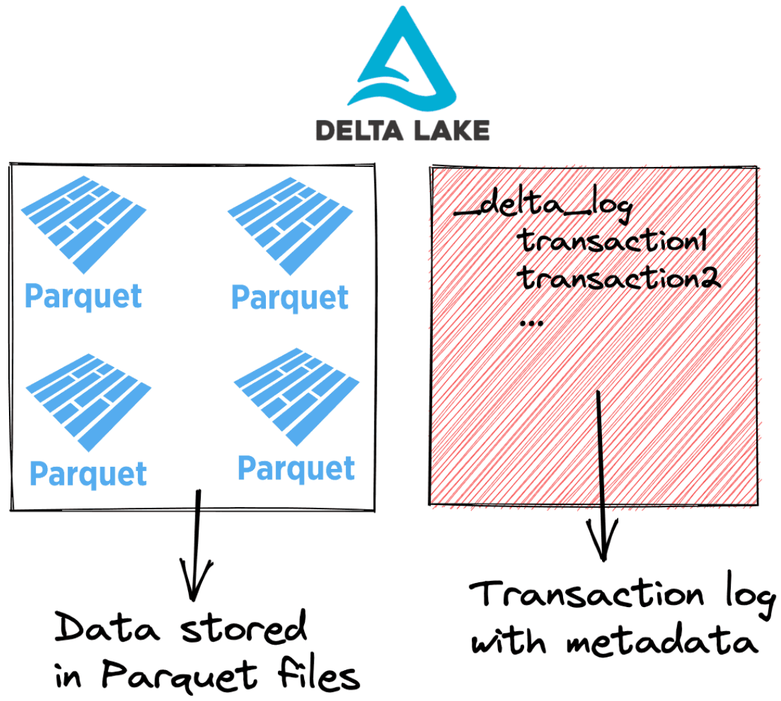
But profiles can evolve..

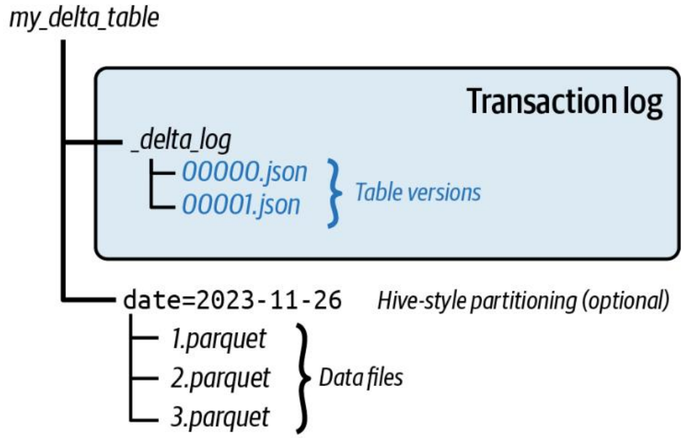
... and workloads do not

⇒ Learn which configs matter for your workloads

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

Underneath a lakehouse (table)





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

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

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

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

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

Data layout tuning

spark.sql.parquet.vorder.default

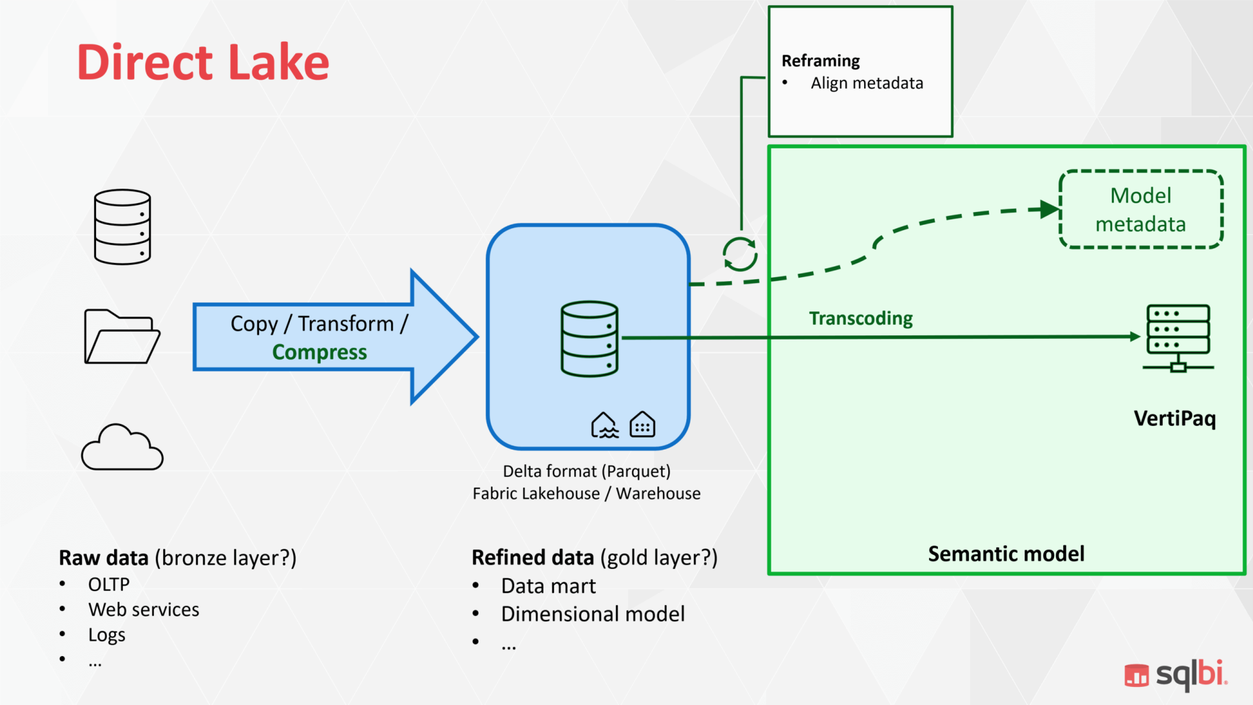
V-Order

- **Write-time** optimization at the Parquet file level
- **Vertipaq-style** sorting and encoding in Parquet files
- **Fabric specific** feature, but 100% compatible with open-source Parquet
- *Lightning-fast reads for Direct Lake*, 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

How V-Order benefits Direct Lake



- Loading data from OneLake when queried (*transcoding*)
- Helps Parquet files become more "*Vertipaq-ready*"
- **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
  - Will incur approx. 10% faster reads

File size tuning

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

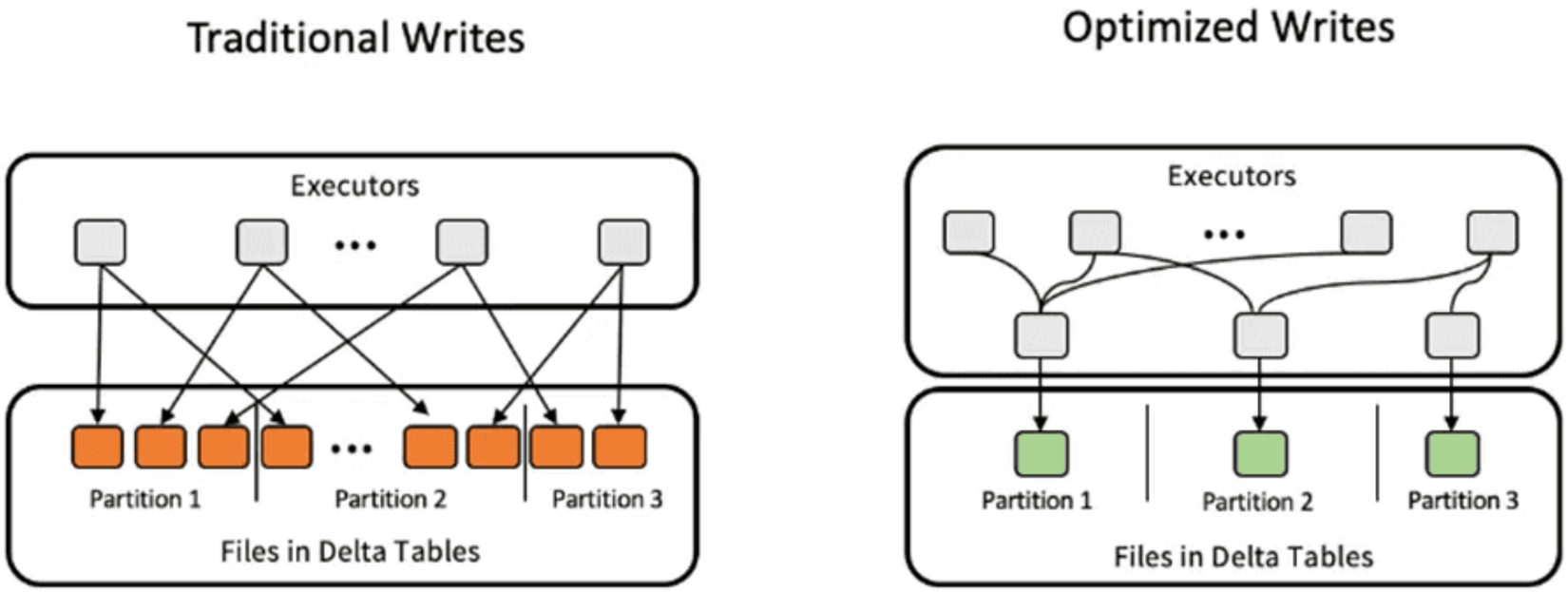
- More small files = more problems
  - Metadata & processing overhead
  - Inefficient I/O
  - Slow query planning

spark.databricks.delta.optimizeWrite.enabled

File size tuning: optimized write

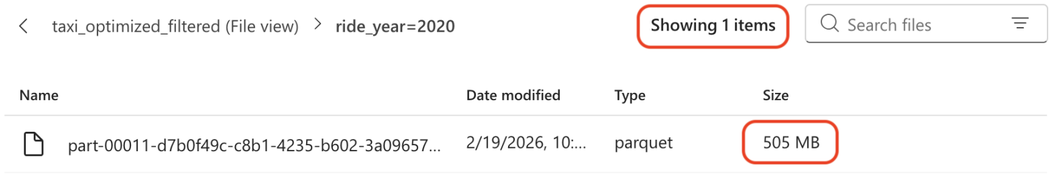
- File size, the number of files, #Spark workers and its configurations => impact on **performance**
- **Reduces # files** written & aims to **increase file size**
- Performs **pre-write** compaction
- Read performance & resource usage improvement, but hit during write performance (extra shuffle)
- Works best on **partitioned tables**
- Delta Lake specific feature

How optimized write works

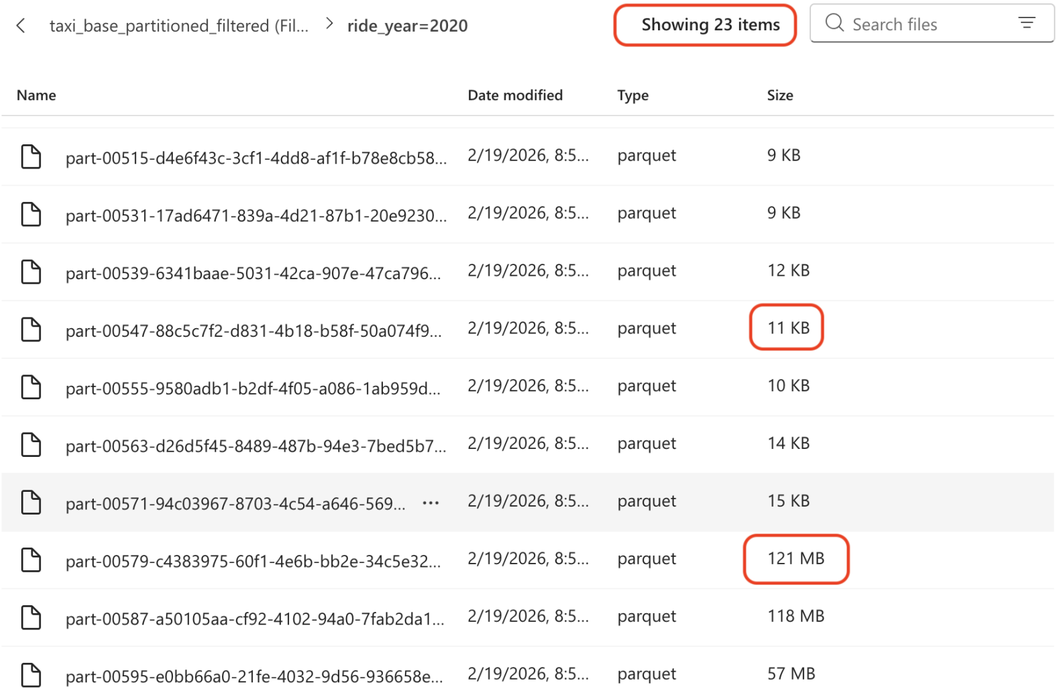


https://delta.io/blog/delta-lake-optimize/

Optimized write



Non optimized write



spark.databricks.delta.autoCompact.enabled

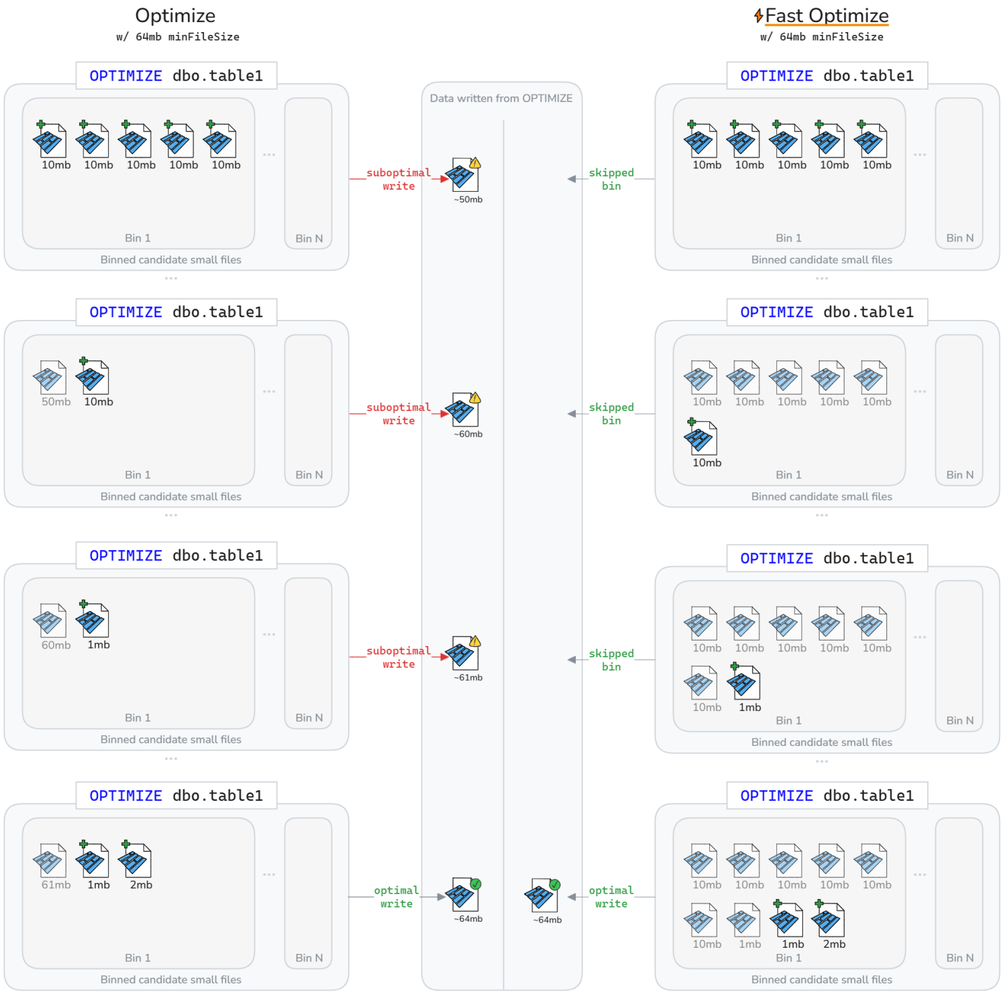
File size tuning: Auto compaction

- Evaluates partition health after each write operation
- **Post-write** compaction (synchronous **OPTIMIZE** operation)
- Thresholds can be tuned (min file size, max file size, min number of files)
- Can be used **together with optimized write**

https://milescole.dev/data-engineering/2025/02/26/The-Art-and-Science-of-Table-Compaction.html

spark.microsoft.delta.optimize.fast.enabled

File size tuning: Fast optimize



- Intelligently analyzes Delta table files
- Extra checks before bins are compacted
- Skips non-performance enhancing compaction operations
- N/A to liquid clustering and Z-Order

File size tuning: Adaptive target file size

spark.microsoft.delta.targetFileSize.adaptive.enabled
spark.microsoft.delta.optimize.fileLevelTarget.enabled

- One size does not fit all
  - table sizes can grow
  - Already mentioned file size tuning techniques use different file size settings (can also use **defined target file size** setting)
  - Wrong file size can have serious performance issues

- Use Delta table telemetry to determine ideal file size & applies
  - During table creation operations
  - OPTIMIZE operations

- Automatically updates as conditions change

- Provides one **unified file size setting** for all file size tuning strategies

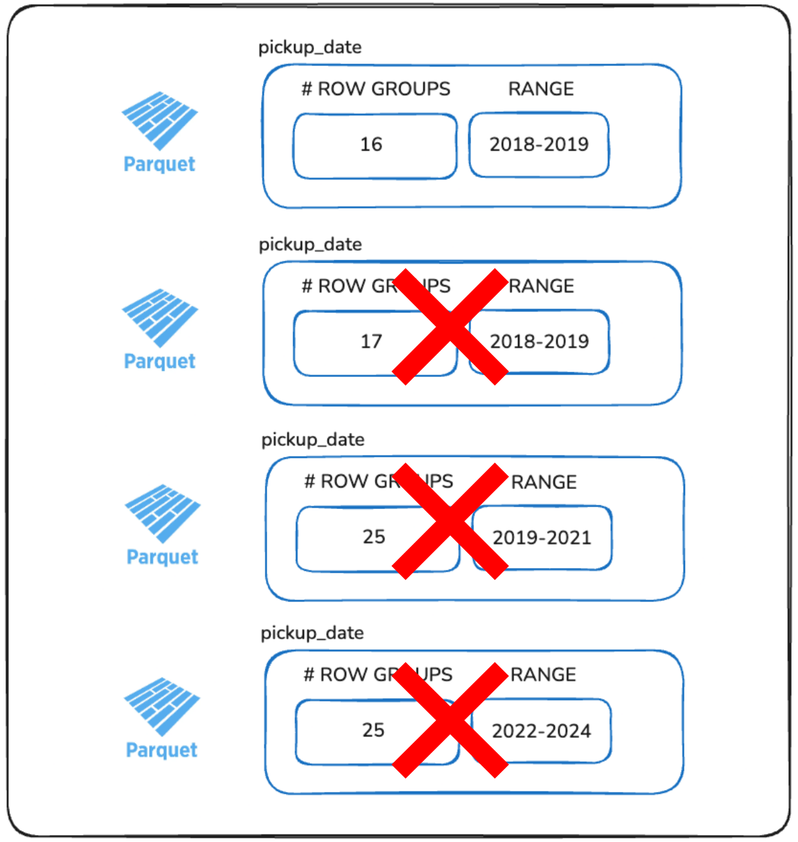
- Set **file level compaction targets** to avoid rewrite of previously compacted data

Z-Order

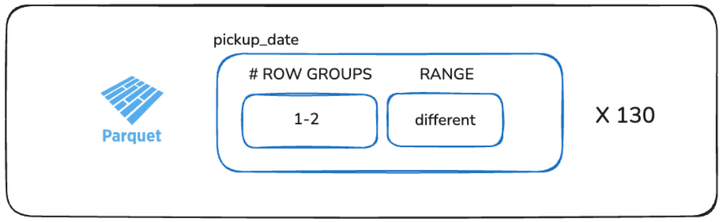
- Optimization at the Parquet level
- **Open-source Delta Lake feature** (not Fabric-specific)
- **Co-locating** related information in the same set of files based on specified columns
- Provides **data skipping**
- Applied during OPTIMIZE

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

Z-Order



Non Z-Order




```
SELECT SUM(total_amount) AS total_trip_amnt, pickup_date
FROM dbo.taxi
WHERE pickup_date >= '2018-06-29' AND pickup_date <= '2018-07-15'
GROUP BY pickup_date
```

When to use what?

	When is it performed?	When to use	When not to use
Optimized write	During write operation	Partitioned tables, batch ingestion, frequent small inserts, MERGE / UPDATE / DELETE	Latency-sensitive writes where extra shuffle hurts too much
Auto compaction	After write operation	Streaming / micro-batch ingestion, frequent small writes (but is fine for all workloads)	Latency-sensitive writes
Fast optimize	During separate OPTIMIZE	You already run scheduled/manual OPTIMIZE and want to skip low-value rewrites	Does not help with autocompaction
Adaptive target file size	All file size strategies	You use different strategies together	You love control
Z-Order	During separate OPTIMIZE	Same columns commonly used in query & high cardinality	All other scenarios

Metadata

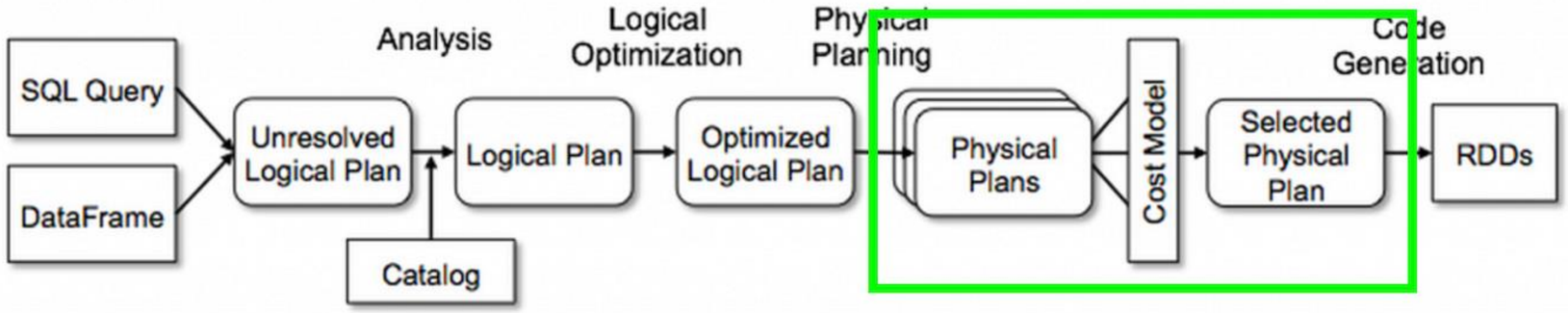
spark.microsoft.delta.stats.collect.extended
spark.microsoft.delta.stats.injection.enabled

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
- *BUT* +- 25% slower writes
- No more need for manual *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, value histogram, 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.

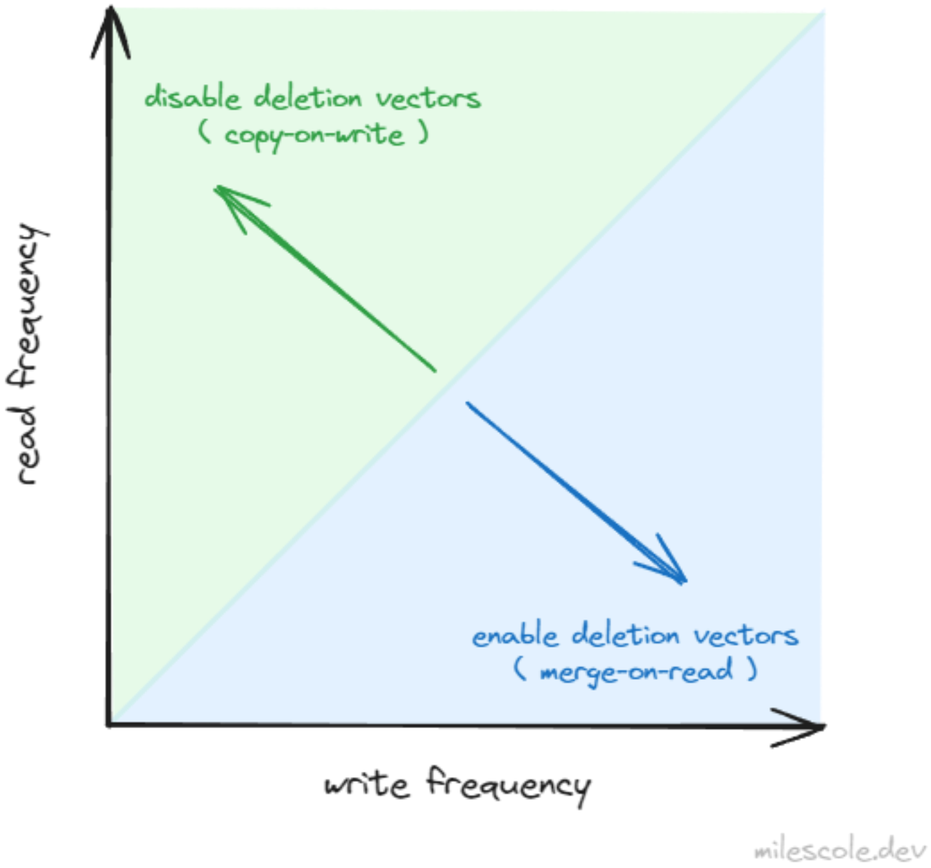


Currently only leveraged by Spark

Deletion vectors

- **Storage optimization** feature on Delta tables
- **Copy-on-write vs merge-on-read**
- Mark deleted rows as removed **without rewriting the entire Parquet file** (soft-deletes)
- Changes applied only on certain commands
  - OPTIMIZE
  - DML command with deletion vectors disabled
  - REORG TABLE … APPLY PURGE
- ! Enabling it will **permanently increase** minReaderVersion and minWriterVersion

When to use deletion vectors?



- Strong choice for **most workloads** (with merge patterns / frequent updates)
- **High write** frequency
- Maintenance strategy in place
- **NOT** in read-heavy scenarios with infrequent writes

https://milescole.dev/data-engineering/2024/11/04/Deletion-Vectors.html

Fabric Spark Profiles (finally!)

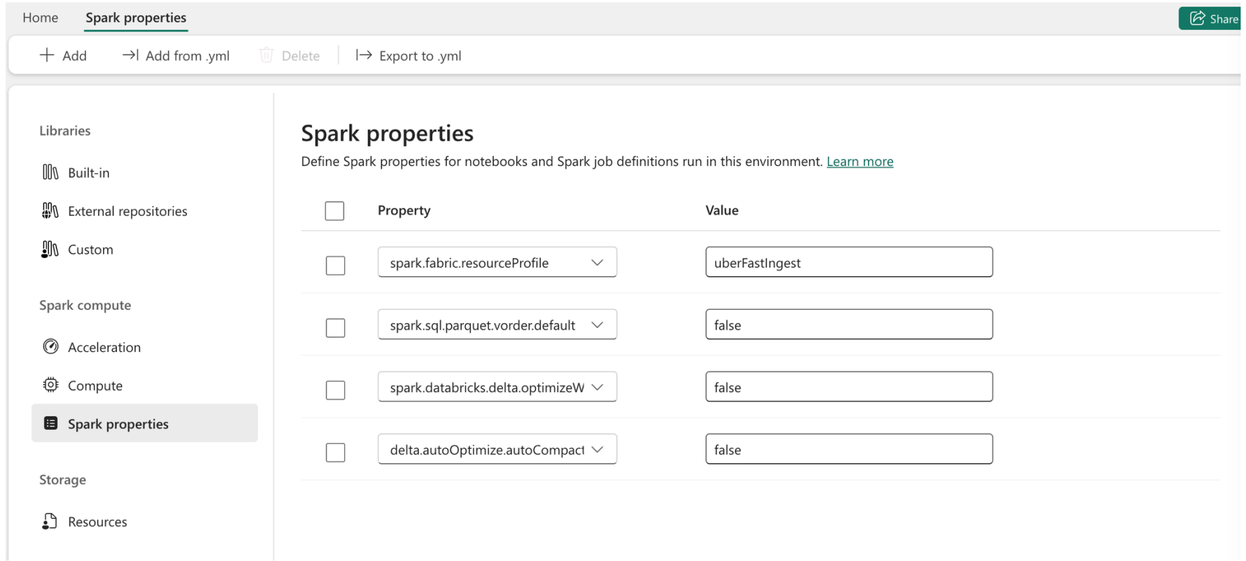
What is currently included?

	writeHeavy	readHeavyForSpark	readHeavyForPBI
V-Order	No	No	Yes
Optimized write	Only when partitioning	Yes	Yes
Default bin size	128Mb	128Mb	1Gb

Each consumption engine benefits from its own settings

Consumption engine	Target file size	Row group size
SQL analytics endpoint	400 MB	2 million rows
Power BI Direct Lake	400 MB to 1 GB	8+ million rows
Spark	128 MB to 1 GB	1-2 million rows

How to create a custom profile



- Custom Spark environment
- Overwrite **spark.fabric.resourceProfile**
- Use in Spark workloads

Recommendations

	Fast low-latency (Bronze)	Mutable ingestion/CDC (Bronze/silver)	Heavy Spark transforms (Silver)	Interactive Spark analytics (Silver/Gold)	Power BI Direct Lake (Gold)
V-Order	avoid	avoid	avoid	avoid	recommended
Optimized write	use with care	use with care	use with care	recommended	recommended
Auto compaction	use with care	recommended	recommended	recommended	recommended
Fast optimize	recommended	recommended	recommended	recommended	recommended
Adaptive target file size	recommended	recommended	recommended	recommended	recommended
Deletion vectors	recommended	recommended	recommended	optional	optional
Table stats	avoid	avoid	optional	recommended	optional

autoUpdate is here!

spark.fabric.resourceProfile.readHeavyForPBI
spark.fabric.resourceProfile.readHeavyForPBIAutoUpdate
spark.fabric.resourceProfile.readHeavyForSpark
spark.fabric.resourceProfile.readHeavyForSparkAutoUpdate
spark.fabric.resourceProfile.writeHeavy
spark.fabric.resourceProfile.writeHeavyAutoUpdate

- Auto-updating resource profiles
- Versions change over time to include new features shipped in GA Runtime

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

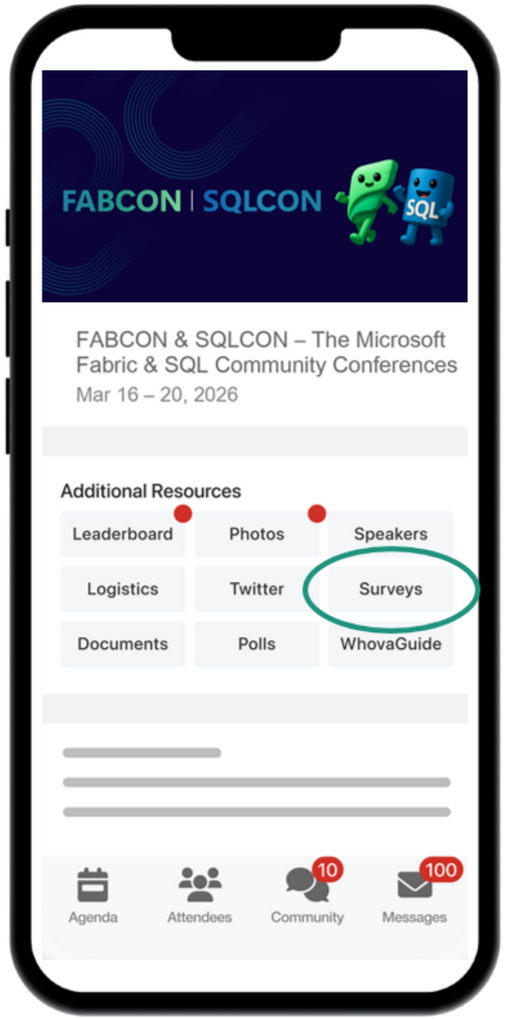
Join the SQL User Panel

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