17th edition
SQLDay Conference

12-14 May 2025, WROCŁAW + ONLINE

SQL Day

Platinum sponsors

DATUMO

Woodler

Gold sponsors

lingaro

VOLVO

TECHNOLOGY INNOVATION DATA KNOWLEDGE tidk

software one

Dataedo

Silver sponsors

Capgemini

DATA RISE LAB

elitmind

C&F

kursy FABRIC

Mastering Microsoft Fabric Data Warehouse Performance

Filip Popović
Senior Program Manager, Microsoft, Serbia

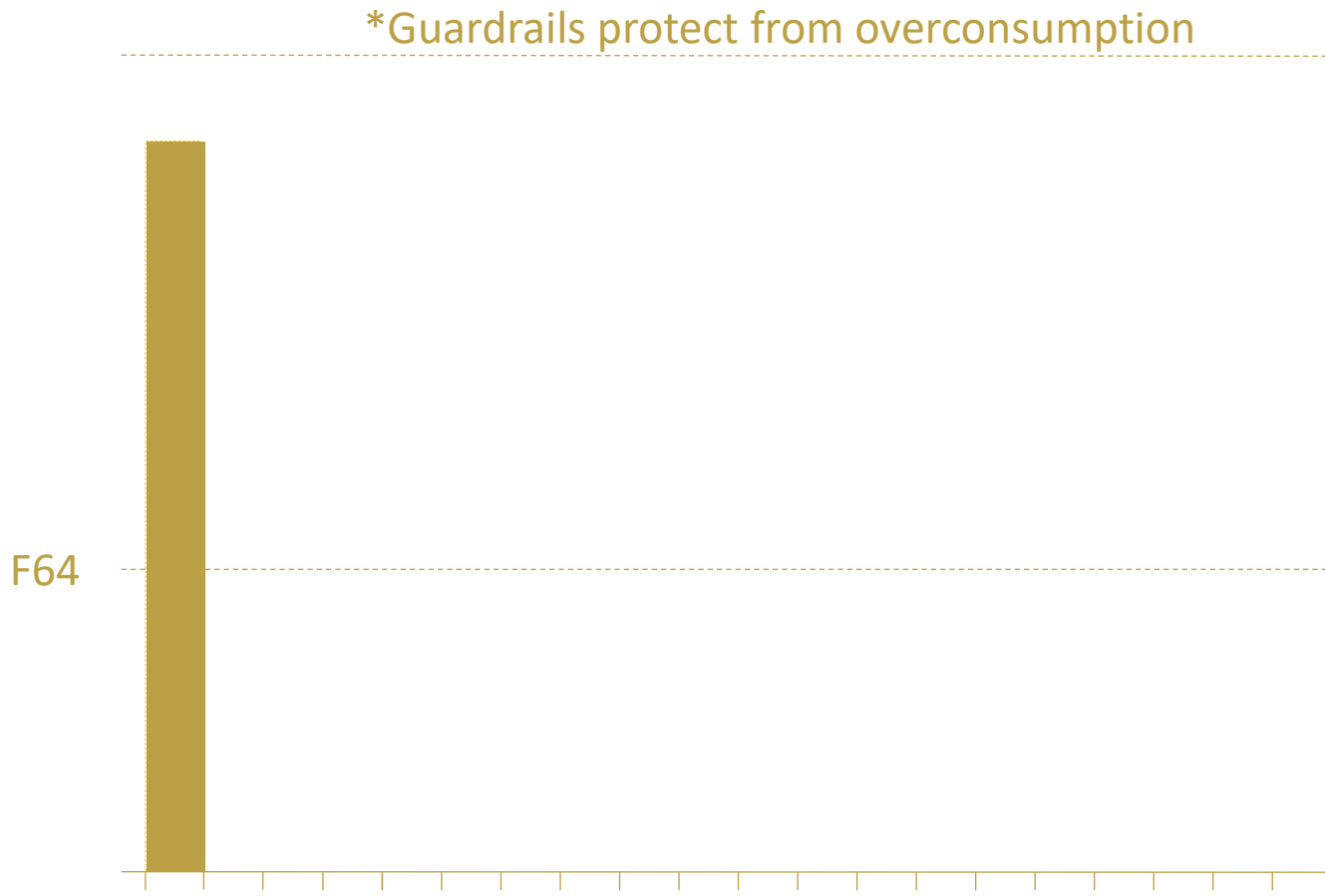
Agenda

- Key ingredients for optimal performance
- Query activity and insights
- Speed up your queries with data clustering
- Consumption considerations
- Ingestion best practices

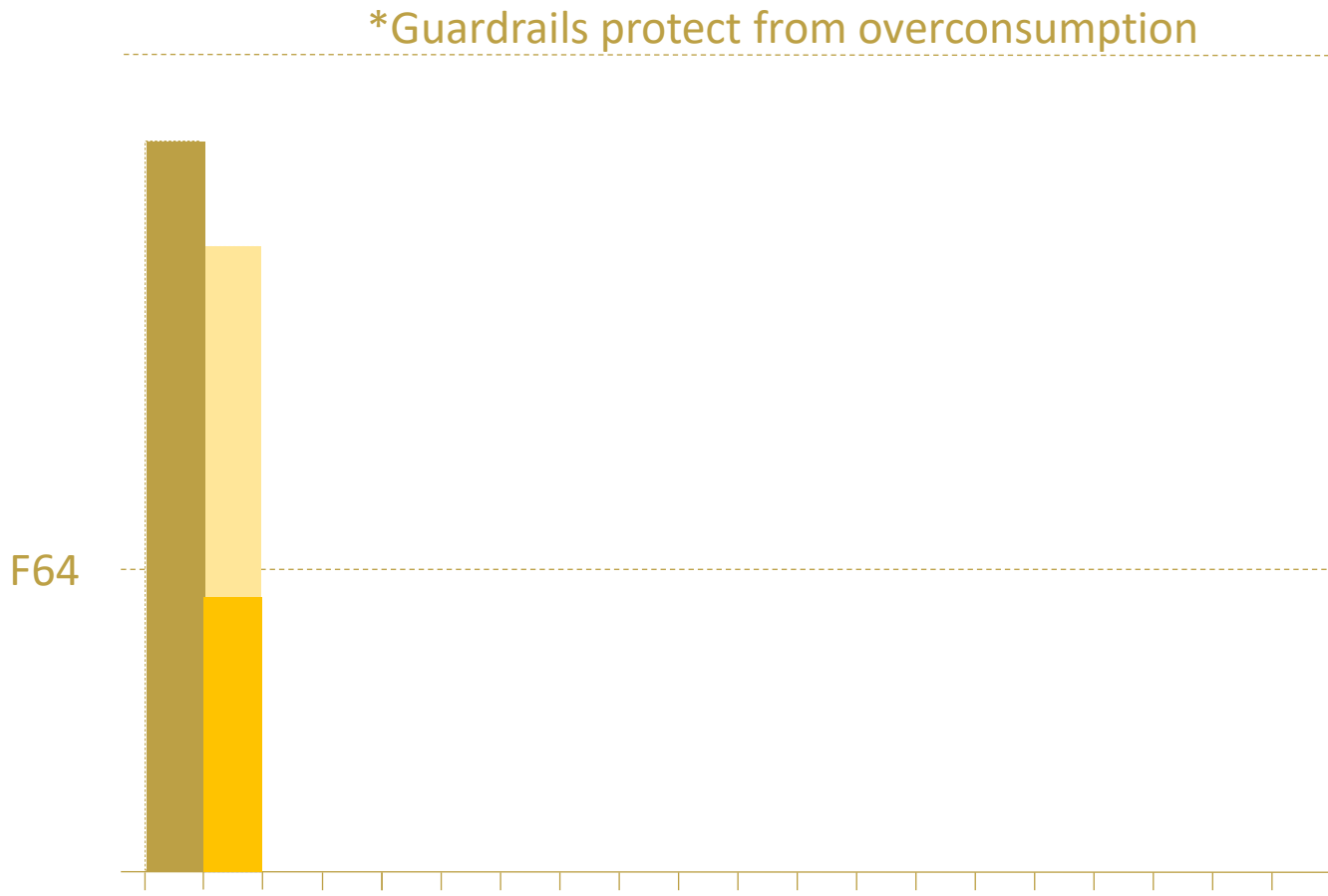
Key ingredients for optimal performance

Always optimal query performance

- Fabric bursting automatically allocates resources as needed to achieve optimal performance
- Query performance is always the same regardless of SKU*
- F2 capacity can run 1TB workloads thanks to bursting!

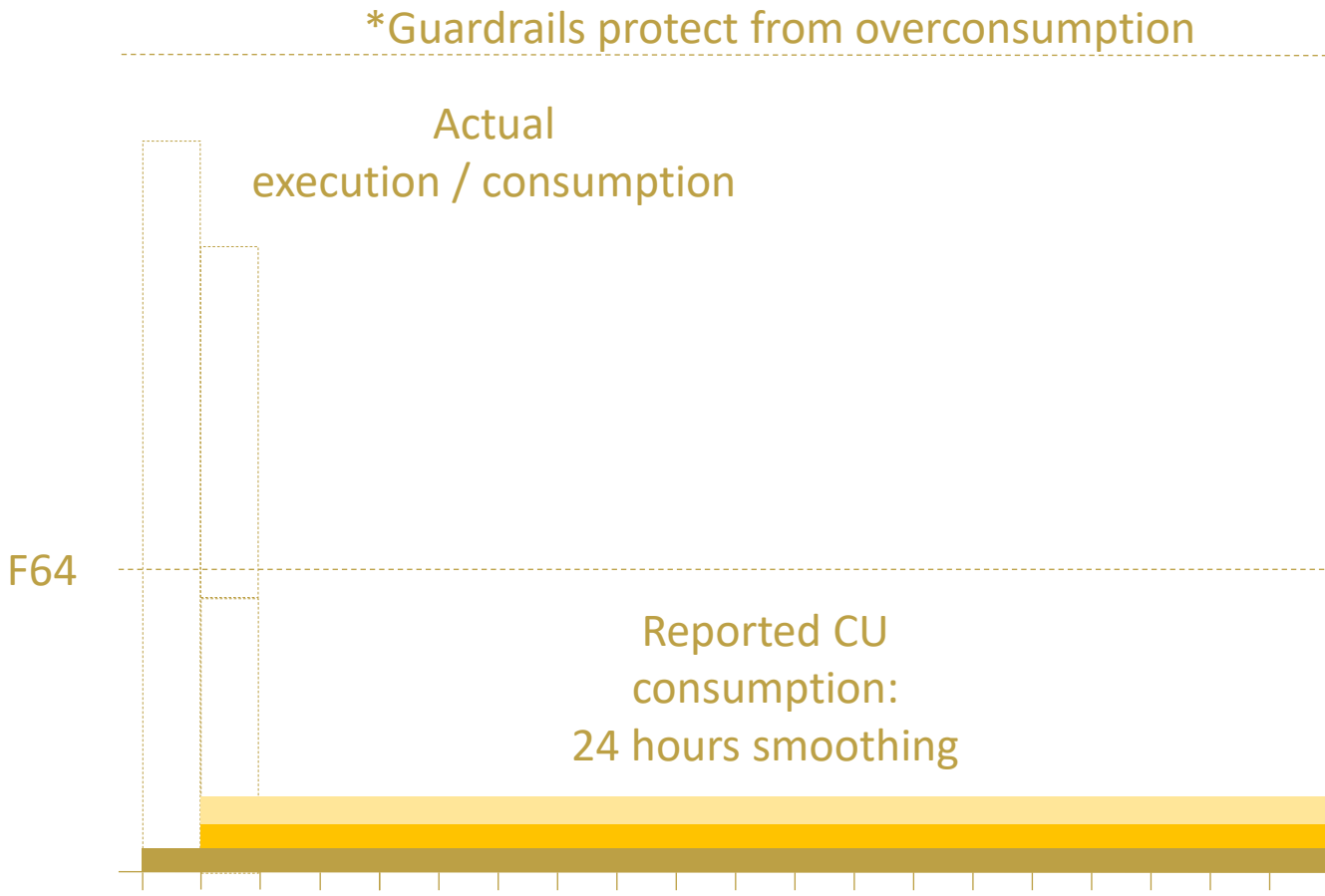


No more sizing for peaks

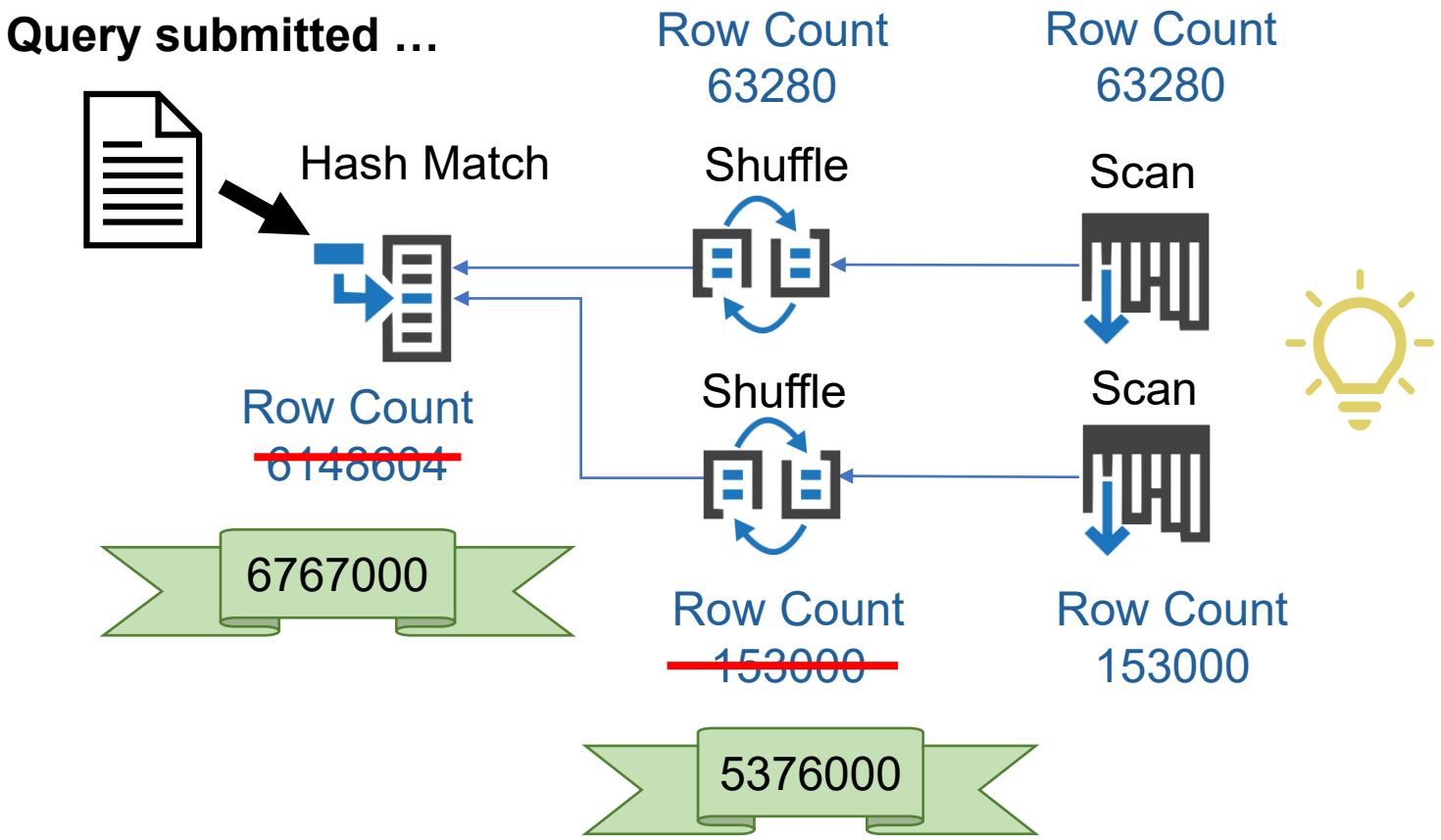


No more sizing for peaks

- Smoothing to future time windows to avoid throttling
- Temporary peaks are handled without DBA intervention or increasing capacity size
- Use Capacity metrics app to track consumption
- F64 (trial) capacity can run TPCH 10TB power run thanks to bursting and smoothing!



Learned resource estimation



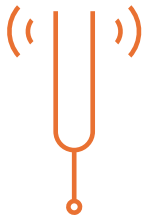
1. Generate plan with current statistics.

2. Check if actual cardinality for sub-trees exists from previous query execution.

3. Use actual cardinality when applicable.

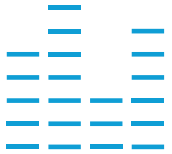
4. Execute query with accurate estimations!

Learned resource estimation



Optimize resource estimates for each task in your query based on previous executions

Improve query performance



Improve compute utilization by avoiding resource miss-estimates

Reduce data spilling significantly

Query activity and insights

Workload monitoring

- DMVs
  - Active connections, sessions and requests
- Query Insights
  - Historical – track performance over time
  - Aggregated – more actionable insights
    - Queries are aggregated by query text excluding predicates
  - Cross-database queries are shown in context DB query insights
- Query activity
  - Monitoring UI built on top of DMVs and Query Insights

Identify optimization opportunities

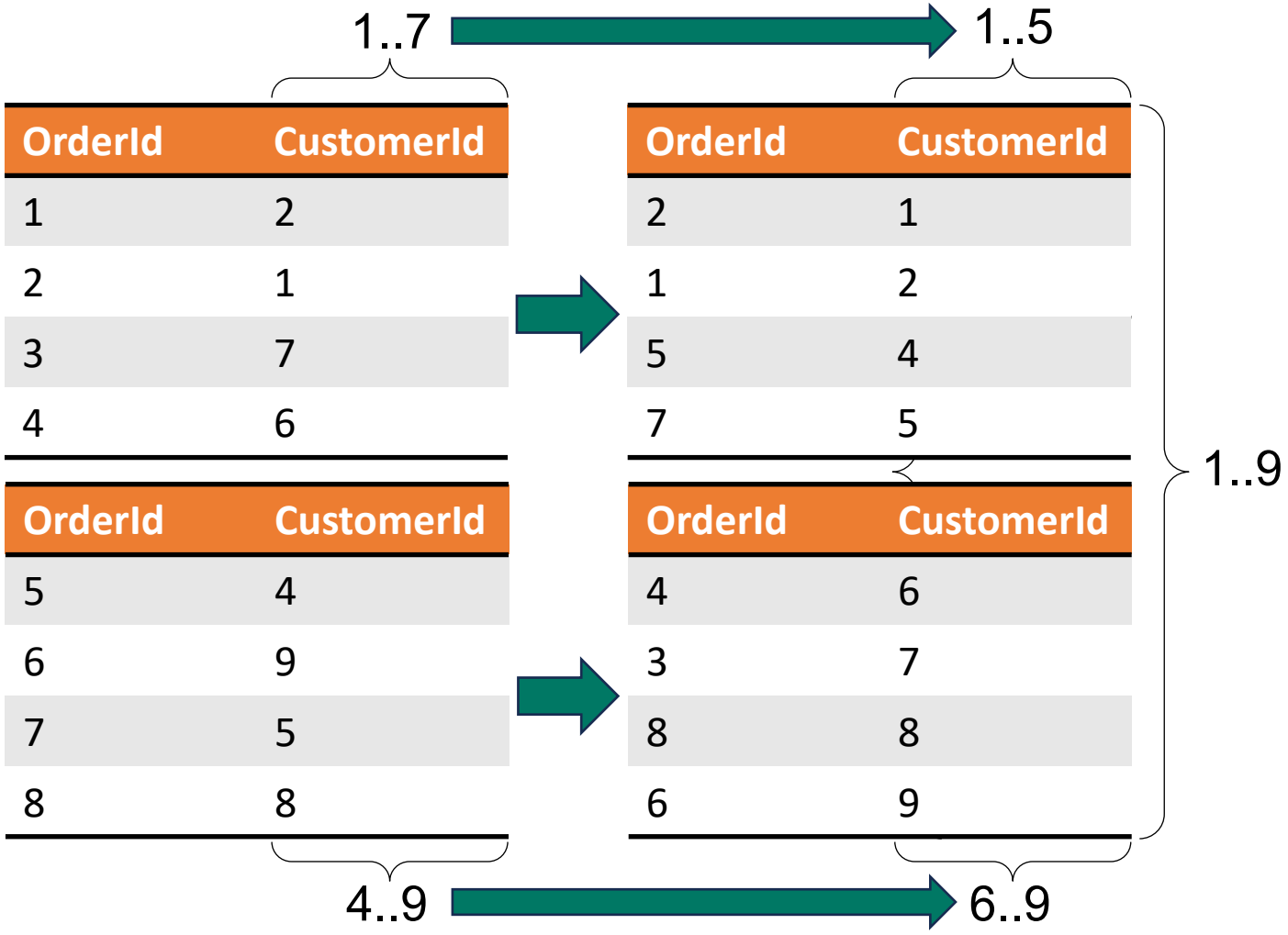
- queryinsights.exec_requests_history
  - data_scanned_remote_storage_mb
  - data_scanned_disk_mb
  - data_scanned_memory_mb
  - allocated_cpu_time_ms

	label	data_scanned_remote_storage_mb	data_scanned_memory_mb	data_scanned_disk_mb	total_cpu_time_ms
1	baseline - cold	63399	0	0	17985154
2	baseline - warm	0	63399	0	3562064
3	clustering - cold	8788	0	0	1312427
4	clustering - warm	0	8788	0	270994

Speed up your queries with data clustering

Coming soon

Reduce IO with data clustering



Coming soon

Collocate data points that are close to each other

Improve performance by skipping rowgroups and files

Data clustering – guidelines

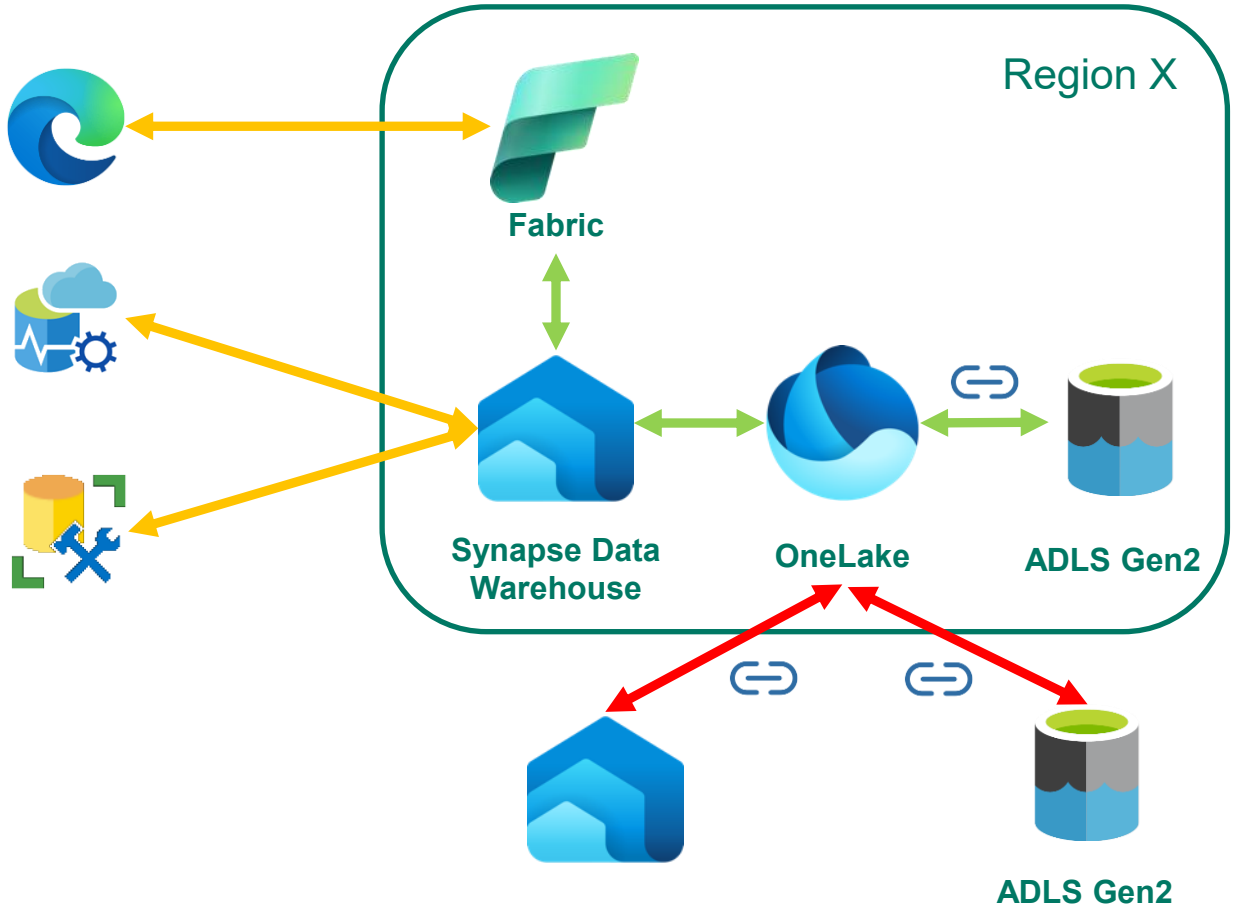
- Pick columns used in query predicates
- Pick column with medium to high ratio of unique values (>=60%)

```
SELECT COUNT(DISTINCT <columnName>) * 1. / COUNT(*) FROM <table>
```

Consumption considerations

Where are my compute and data?

- Collocate resources
- Compute is where capacity is
- Mind network latency between:
  - The client and the endpoint
  - The engine and the data in case of shortcuts



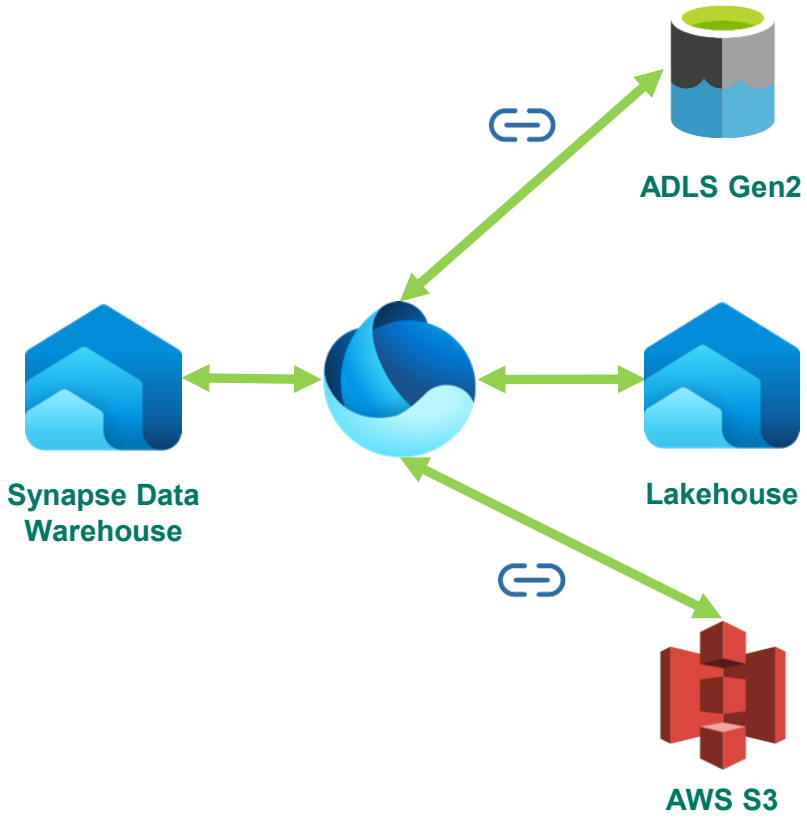
Choosing Optimal Data Types for Real-World Problems*

- Use smallest data type that accommodates values
- Use proper types
- Use varchar instead of char
- Use NOT NULL instead of NULL
- Check table schema created by tools (e.g. Pipelines)

```
year bigint           ->  year int
amount decimal        ->  amount decimal(p,s)

code_id varchar(10)   ->  code_id int


name char(50)         ->  name varchar(50)


id bigint             ->  id bigint NOT NULL


name varchar(8000)    ->  name varchar(50)
```

Optimize Lakehouse tables

Fabric Warehouse can only optimize Warehouse tables



Yet another table to SQL engine, same best practices apply

Check data types, particularly string lengths, decimal precision and scale and use NOT NULL if possible

OPTIMIZE table to get optimal number and sizes of files

Partition table by columns commonly used for filtering

Z-ORDER table by non-partitioning columns commonly used for filtering.

Data ingestion best practices

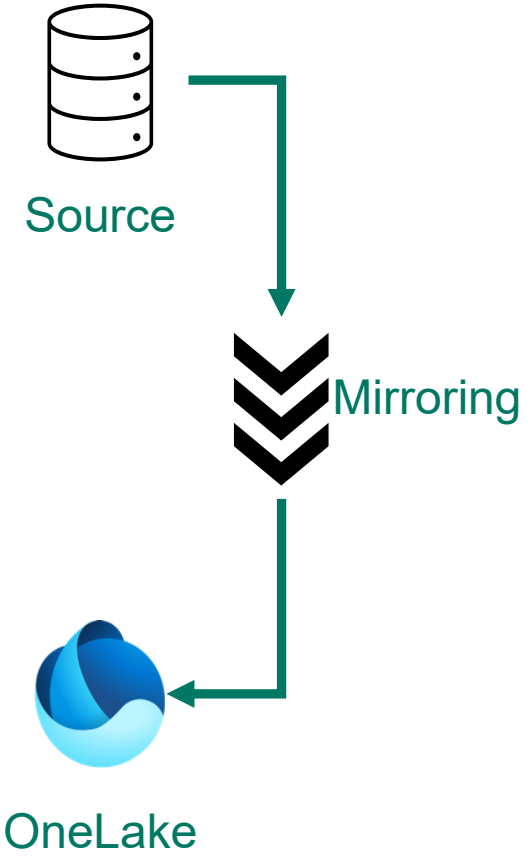
Don’t’ ingest, shortcut or mirror instead

Shortcuts into the lakehouse

- Makes no additional copy of the data

Mirror data

- Near real time replication with zero ETL
- Optimal format for analytics by default
- Currently supported sources:
  - Snowflake
  - Azure Cosmos DB
  - Azure SQL DB
  - Azure SQL MI
  - Azure Database for PostgreSQL Flexible Server
  - Extensible with Open Mirroring!



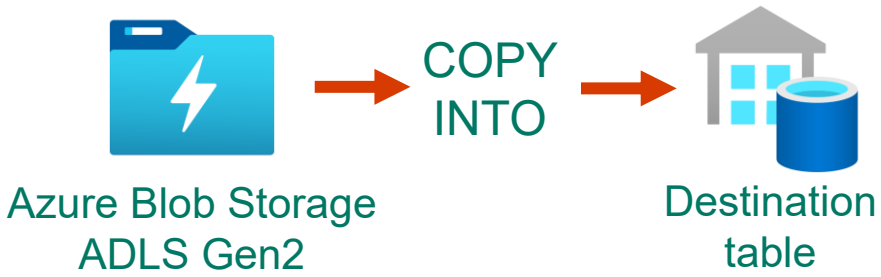
Few tips for a start

From	Format	T-SQL statement	Capabilities
Lakehouse table	Delta	CTAS/INSERT with SELECT	Ingestion with transformation.
ADLS	Delta	Shortcut + CTAS/INSERT with SELECT	Ingestion with transformation.
ADLS	CSV, Parquet	COPY INTO	Ingestion w/o transformation.
ADLS	CSV, Parquet	CTAS/INSERT+SELECT with OPENROWSET	Ingestion with transformation of subset of rows.
ADLS	CSV	BULK INSERT	Ingestion with transformation. Migrating existing code and third-party code.

CLONE TABLE for instantly available replica

COPY INTO

- Enables flexible, high-throughput data ingestion from external Azure storage accounts

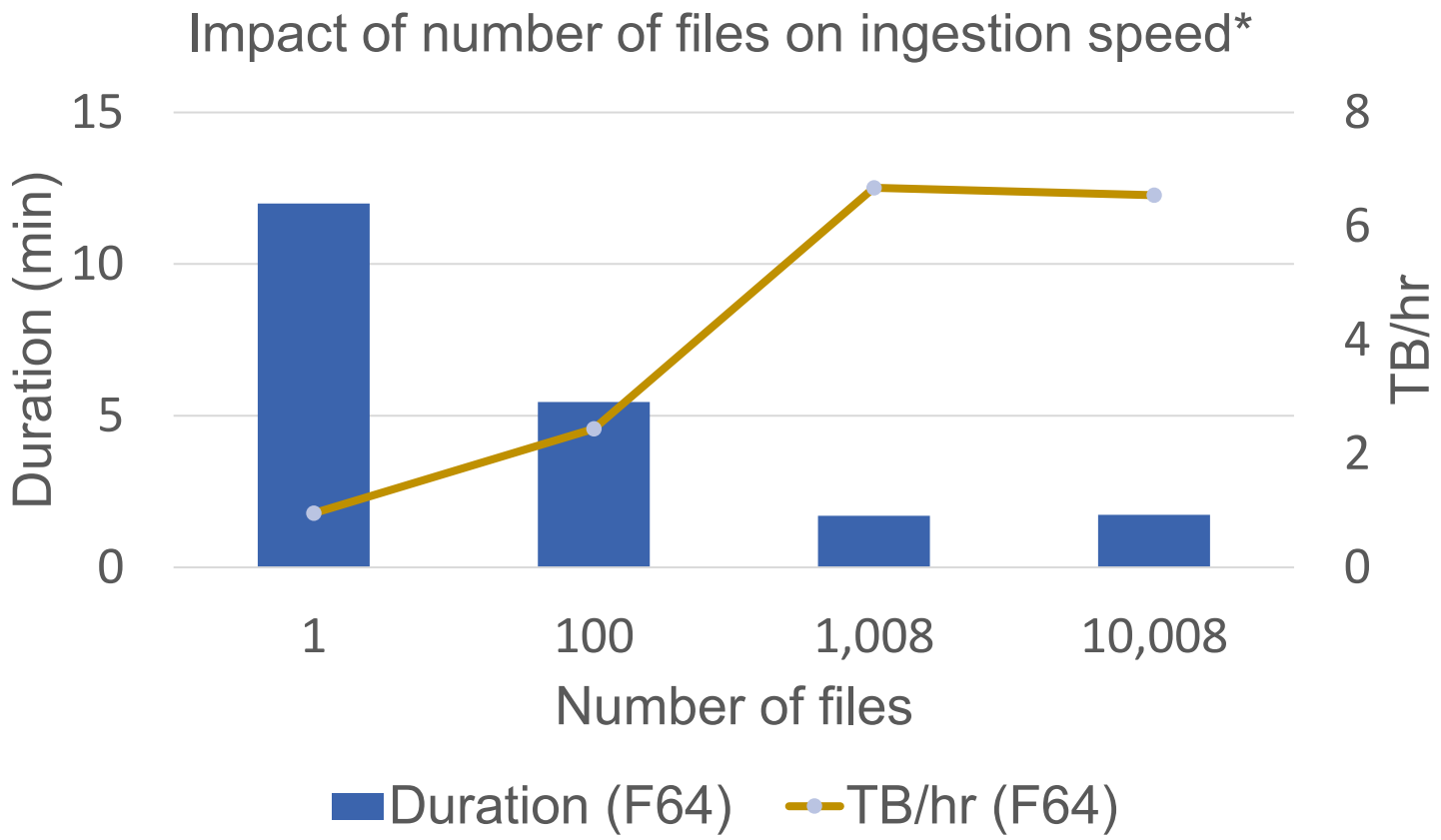



```
COPY INTO table_name.[(Column_list)]
FROM '<external_location>' [,...n]
WITH
 (
 [FILE_TYPE = {'CSV' | 'PARQUET' '} ]
 [,CREDENTIAL = (AZURE CREDENTIAL) ]
 [,ERRORFILE =
'[http(s)://account/container]/directory[/]]']
 [,ERRORFILE_CREDENTIAL = (AZURE CREDENTIAL) ]
 [,MAXERRORS = max_errors ]
 [,COMPRESSION = { 'Gzip' | 'Snappy'}]
 [,FIELDQUOTE = 'string_delimiter']
 [,FIELDTERMINATOR = 'field_terminator']
 [,ROWTERMINATOR = 'row_terminator']
 [,FIRSTROW = first_row]
 [,ENCODING = {'UTF8'|'UTF16'}]
 [,PARSER_VERSION = {'1.0'|'2.0'}]
 )
```

Ingestion best practices - size and number of files

COPY INTO:

- More small files > few large files
- At least 4MB file size
- Keep the number of files to be at least 1,000
- Best throughput using 1,000 files is 7x faster than single large file

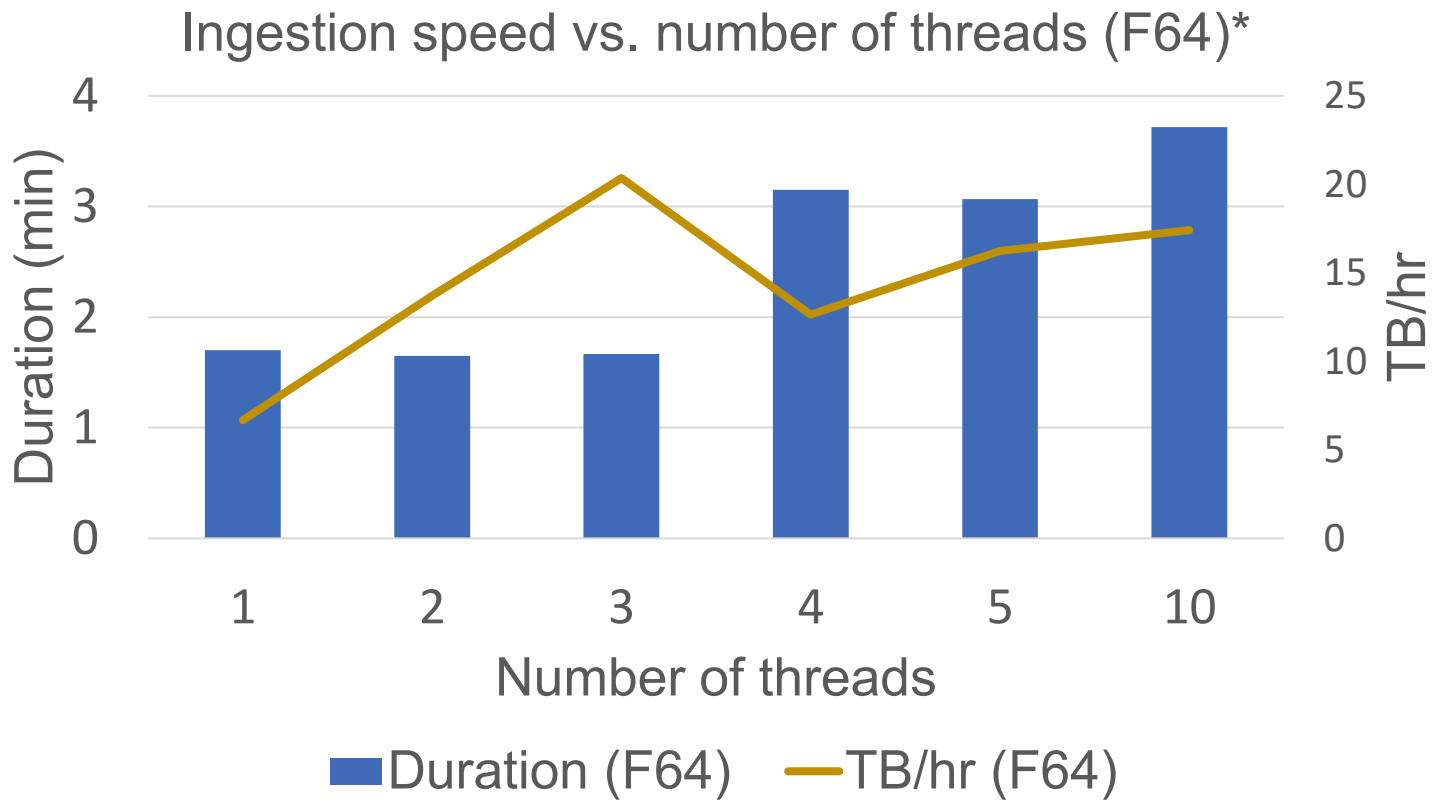


*TPCH 1TB, Orders table

Ingestion best practices - parallelize

COPY INTO:

- Parallel loads achieve higher throughput

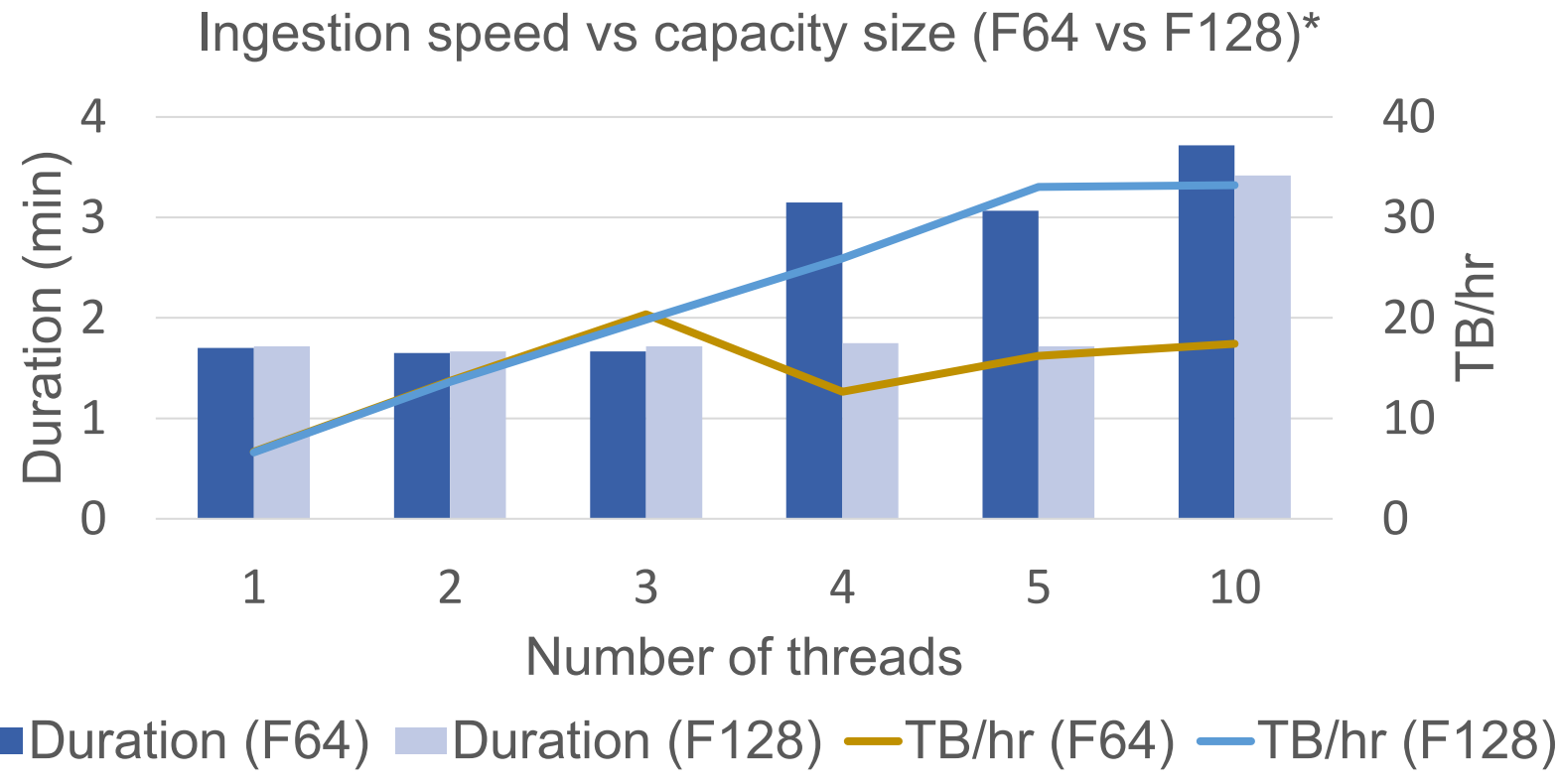


*TPCH 1TB, Orders table, 170GB each thread

Increase throughput with larger Fabric capacities

All ingestion (pipelines, dataflows, SQL):

- Scales with larger Fabric Capacities

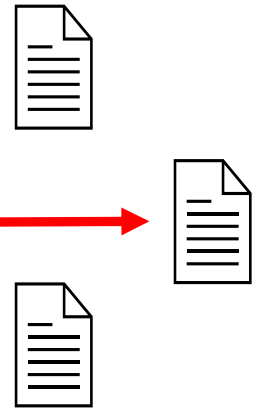


*TPCH 1TB, Orders table, 170GB each thread

INSERT/UPDATE/DELETE data in batches

Avoid singleton INSERT INTO statements – ineffective for both writes and reads

```
INSERT INTO person VALUES (1, 'Mike')
INSERT INTO person VALUES (2, 'Peter')
INSERT INTO person VALUES (3, 'John')
```



```
INSERT INTO person VALUES (1, 'Mike'),
    (2, 'Peter'),
    (3, 'John')
```

Q&A, feedback and bits for refreshing memory

- Q&A
- Please provide feedback in the Whova App
- Share with your colleagues and your future self ☺
  - Ingesting data into Fabric Warehouse
  - Performance best practices

17th edition
SQLDay Conference

12-14 May 2025, WROCŁAW + ONLINE

Platinum sponsors

DATUMO

Woodler

Gold sponsors

lingaro

VOLVO

TECHNOLOGY INNOVATION DATA KNOWLEDGE tidk

software one

Dataedo

Silver sponsors

Capgemini

DATA RISE LAB

elitmind

C&F

kursy FABRIC