www.advancinganalytics.co.uk

A DIVE INTO DELTA LAKE

A MODERN FILE FORMAT FOR NEXT-GEN LAKES

ADVANCING ANALYTICS

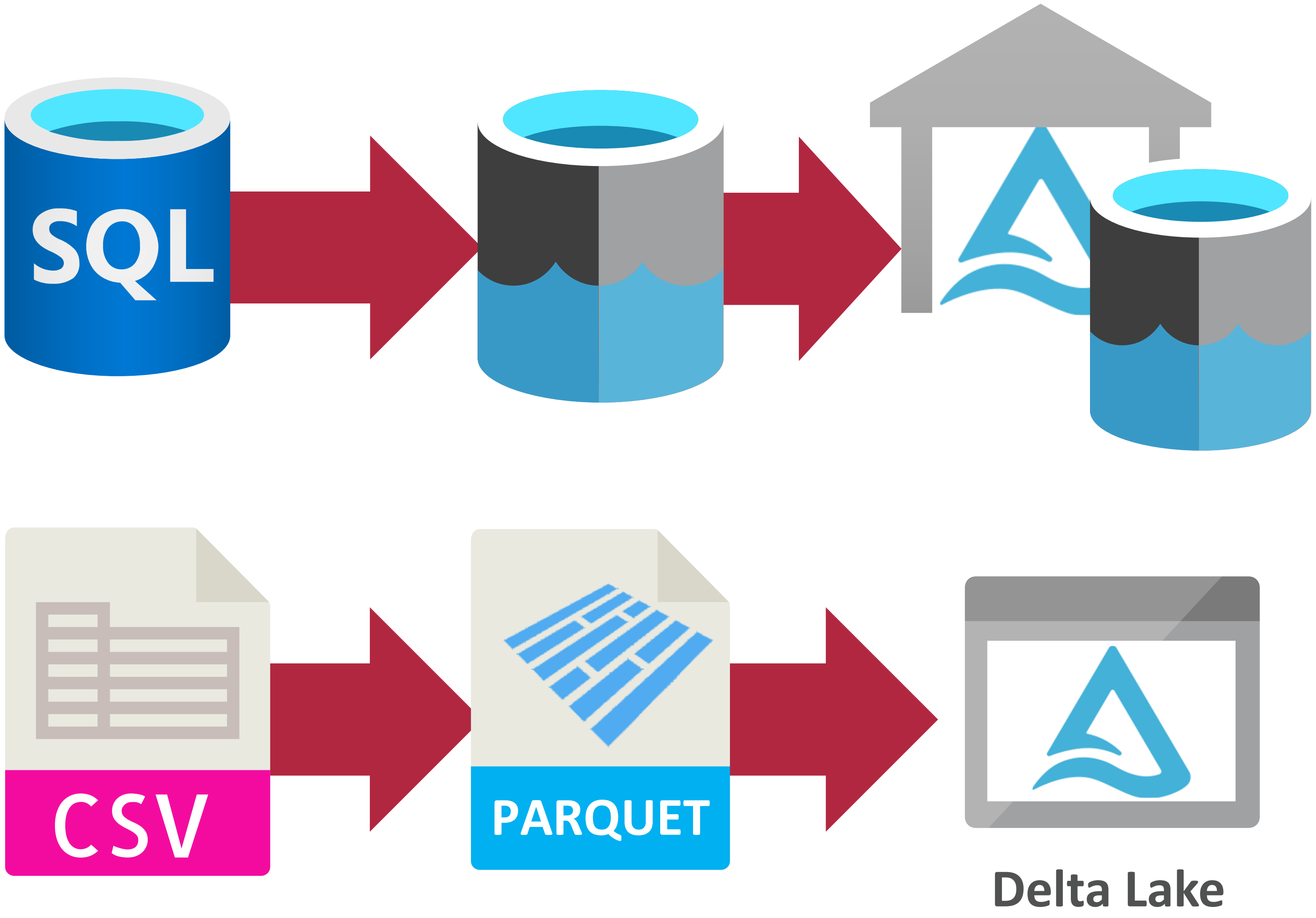
MVP Microsoft® Most Valuable Professional

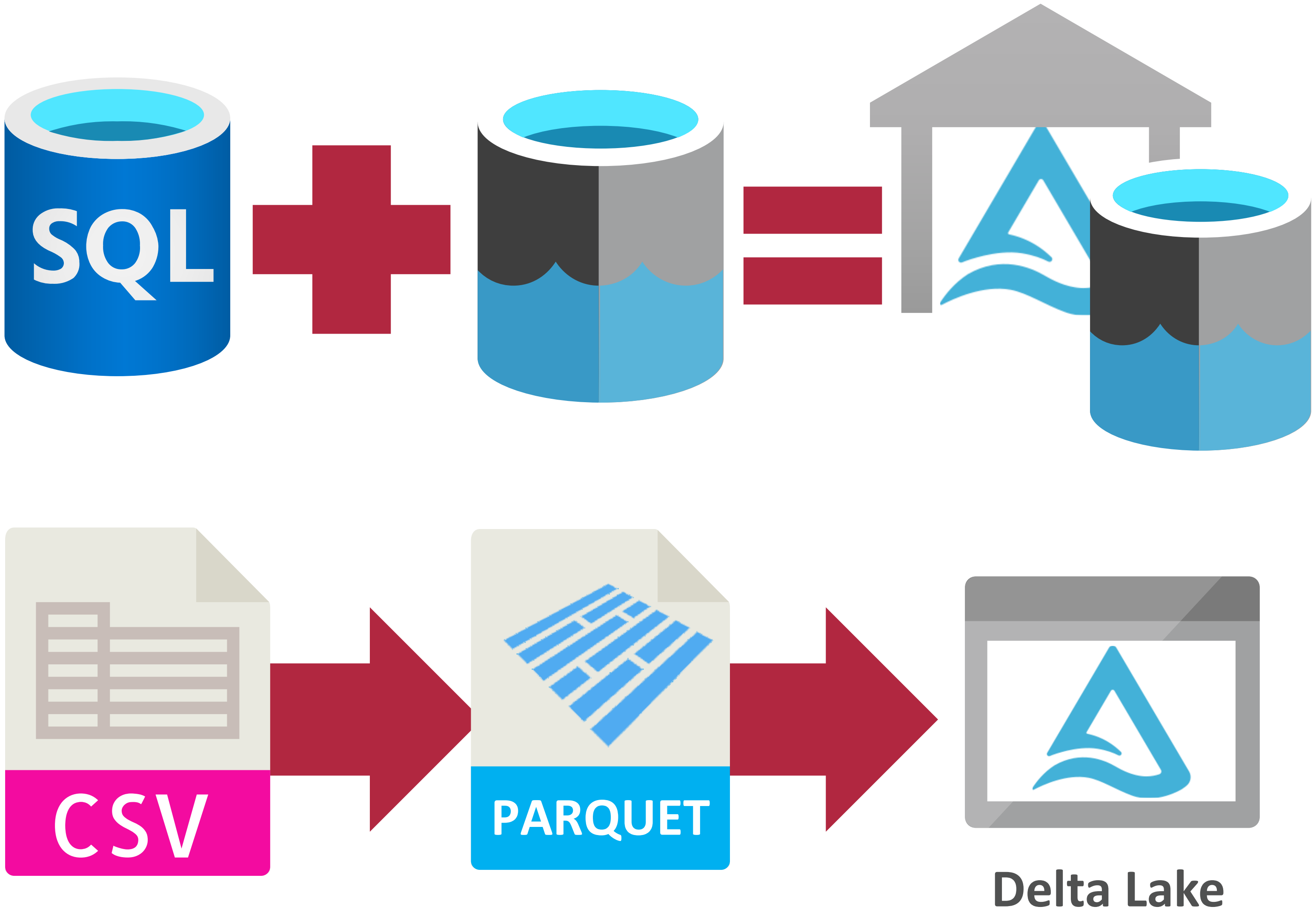
Databricks BEACONS

@ADVANCINGANALYTICS @ADVANALYTICSUK /ADVANCING ANALYTICS

https://delta.io/

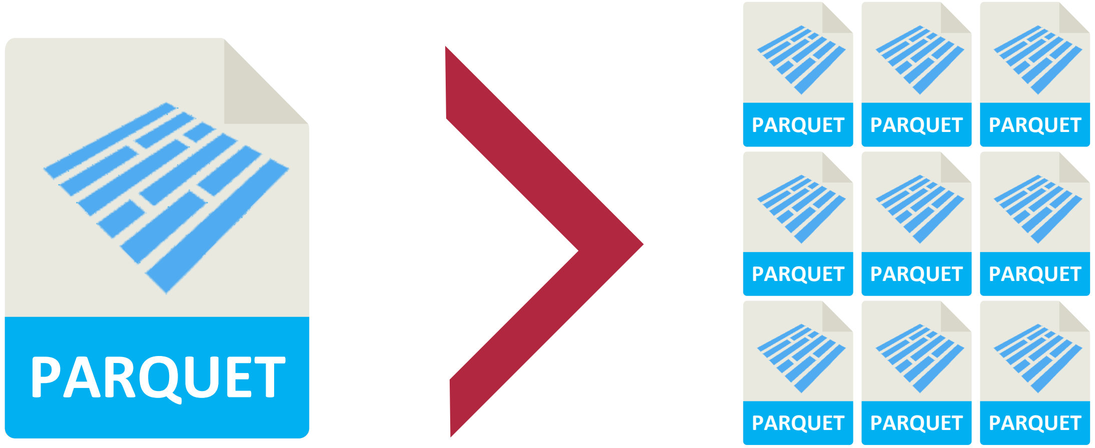
EVOLUTION OF LAKES TO LAKEHOUSES


SQL
CSV
PARQUET
Delta Lake


SQL
CSV
PARQUET
Delta Lake

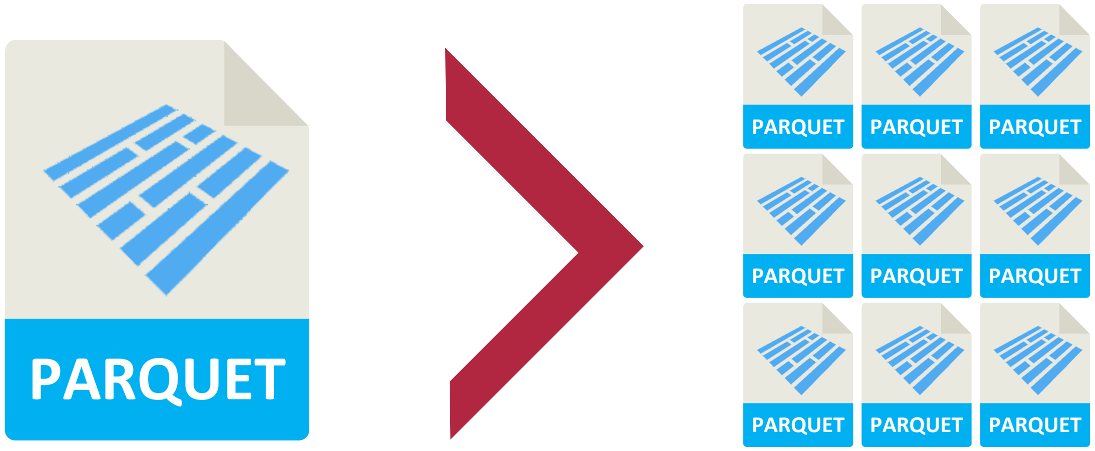
SO WHAT'S WRONG WITH PARQUET?

THE PROBLEMS OF PARQUET



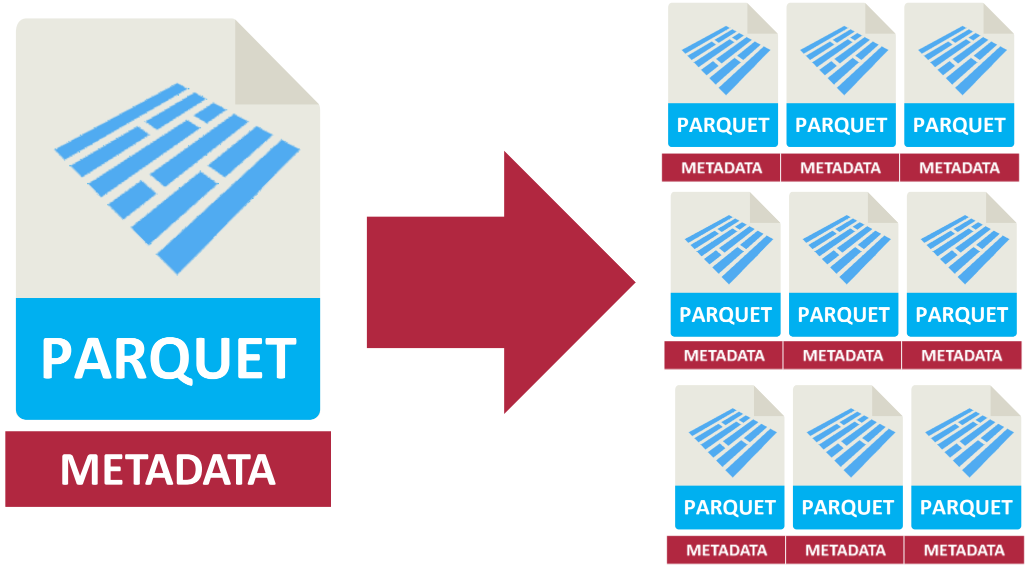
Small files have a heavy performance impact

THE PROBLEMS OF PARQUET



Small files have a heavy performance impact

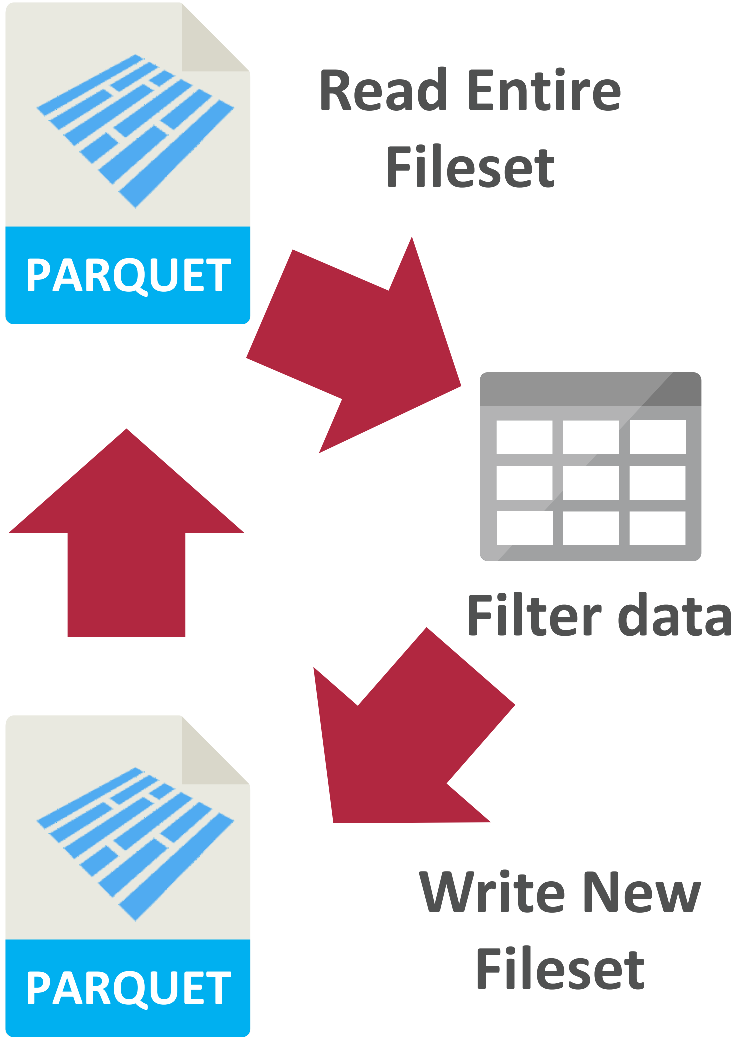
METADATA IN DATA FILES



Metadata scan = reading all files

OPERATIONAL COMPLEXITY

DELETE FROM MyTable WHERE…

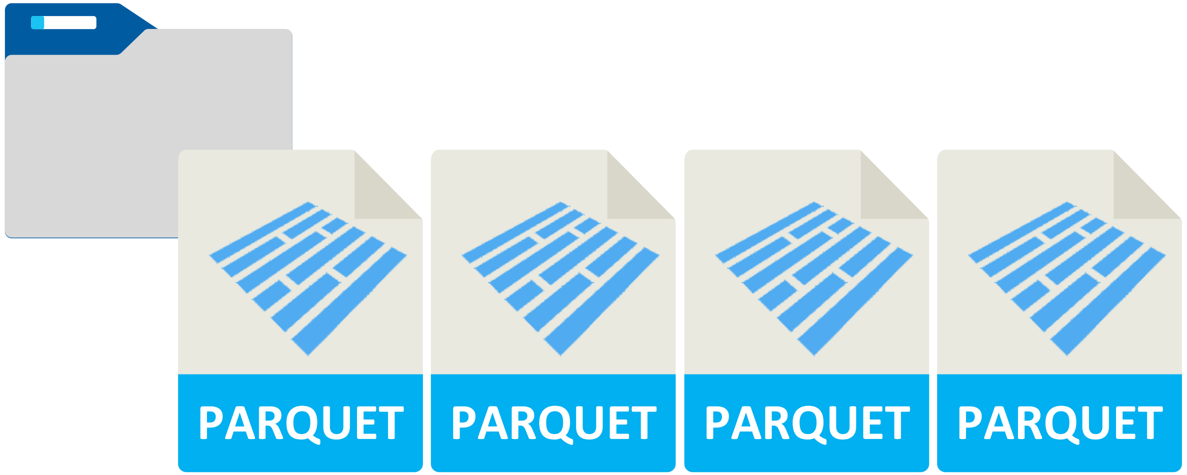


Read Entire Fileset

Filter data

Write New Fileset

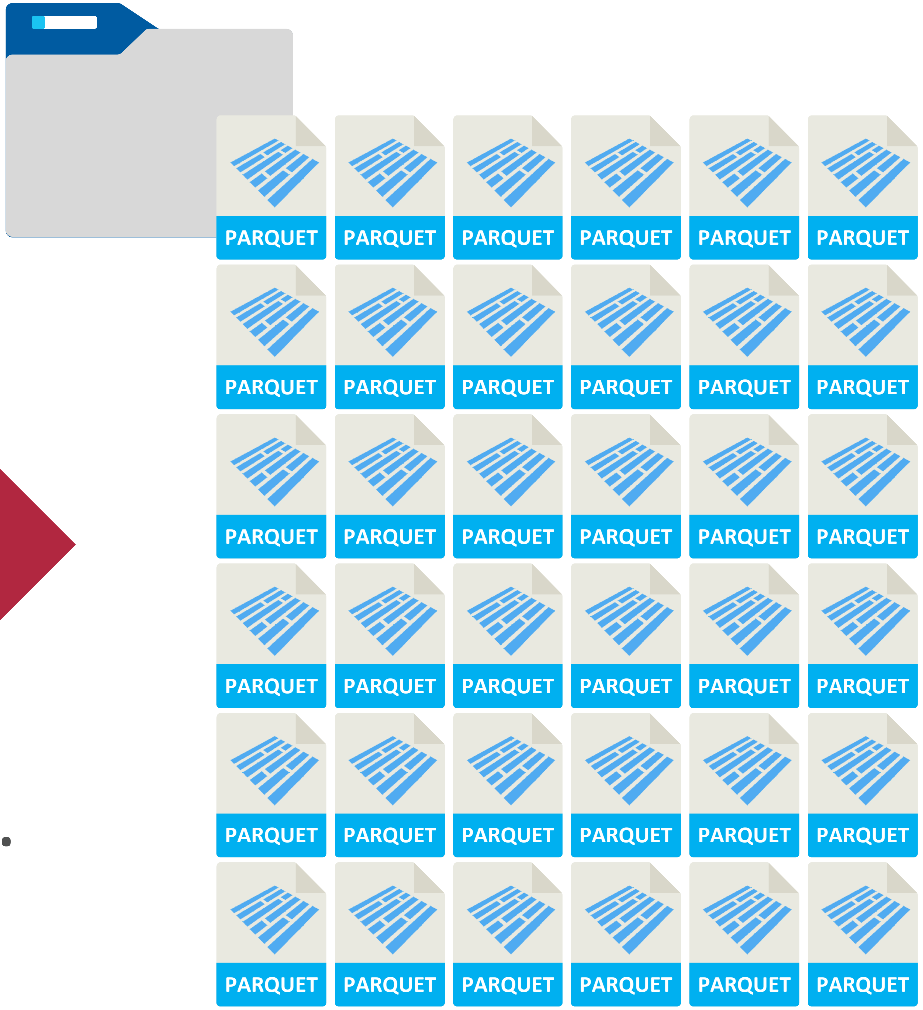
AUDITING & RECOVERY



```
DELETE FROM MyTable WHERE ID = 132
```

NO INDEXES

SELECT * FROM MyTable WHERE…



SO WHAT IS DELTA?

WHAT IS DELTA?

“Databricks Delta is a unified data management system that brings reliability and performance(10-100x faster than Spark on Parquet) to cloud data lakes.

Delta's core abstraction is a Spark table with built-in reliability and performance optimizations.”

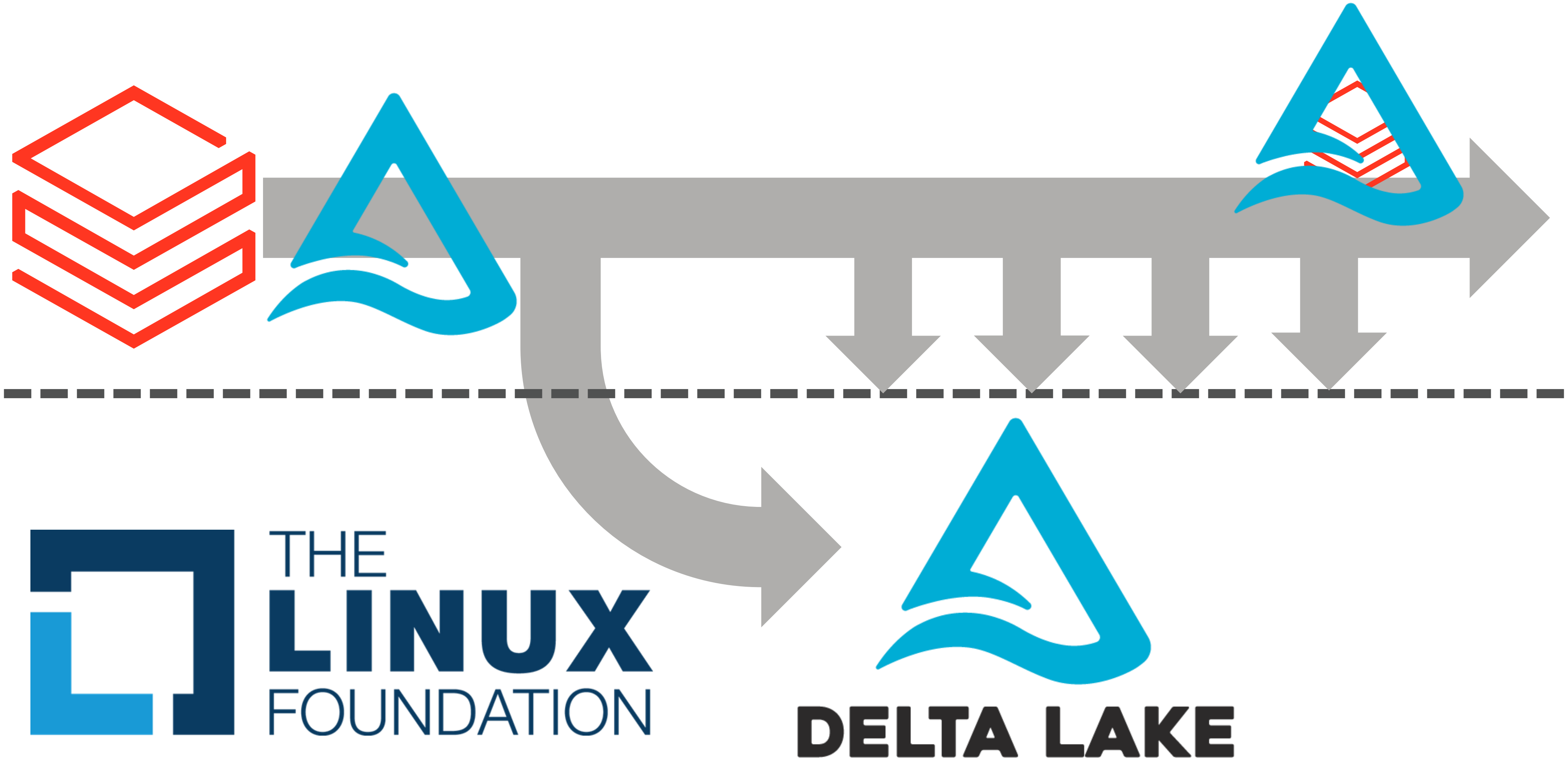
Databricks (2019)

WHAT IS DELTA?

Delta Lake is an ***optimised, managed format*** *for organising & working with* ***Parquet*** *files*

"It's Parquet, but better"

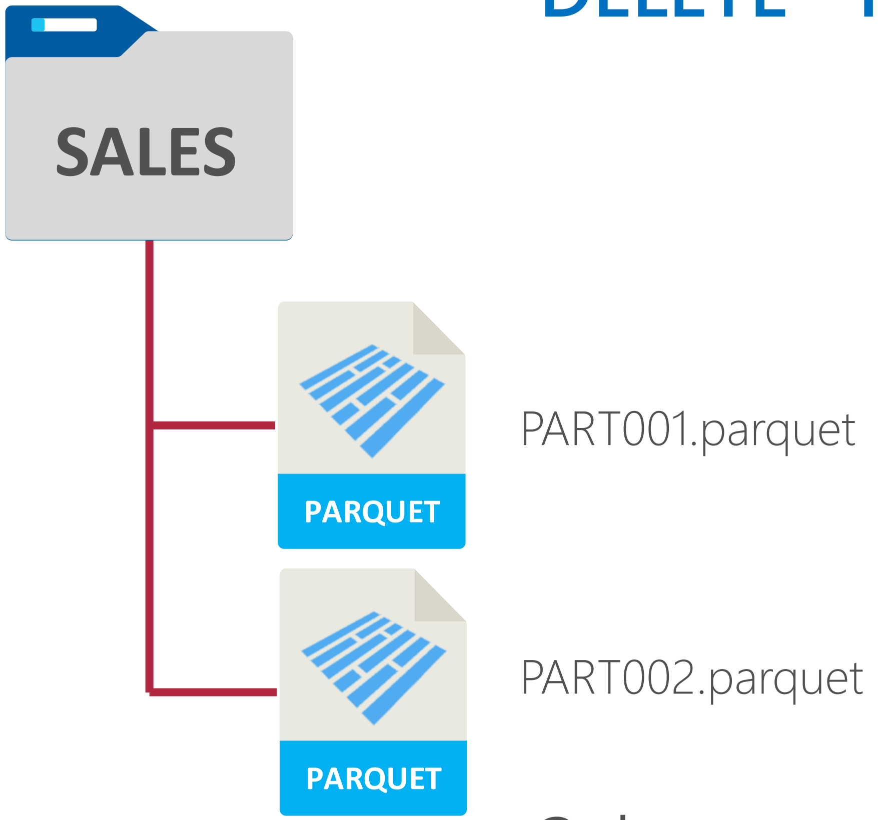
HOW OPEN SOURCE IS DELTA?

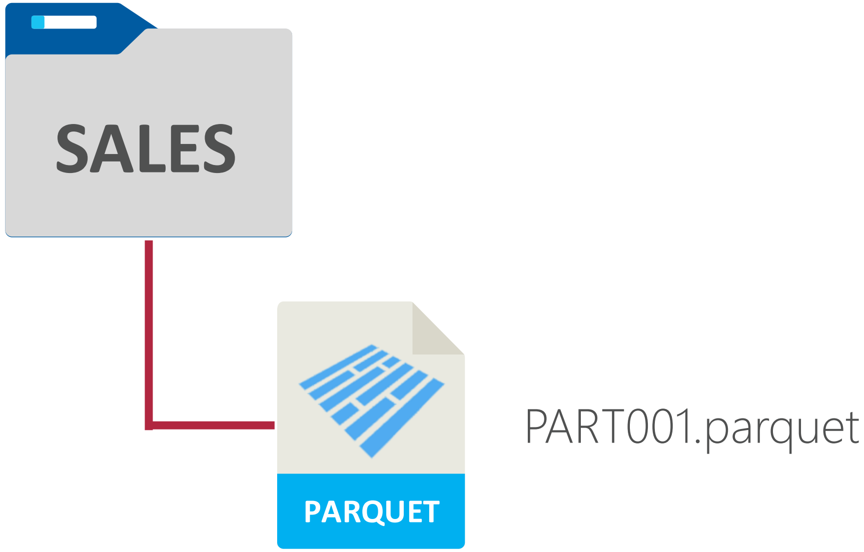


SOUNDS FANCY… HOW DOES IT WORK?

BEFORE DELTA

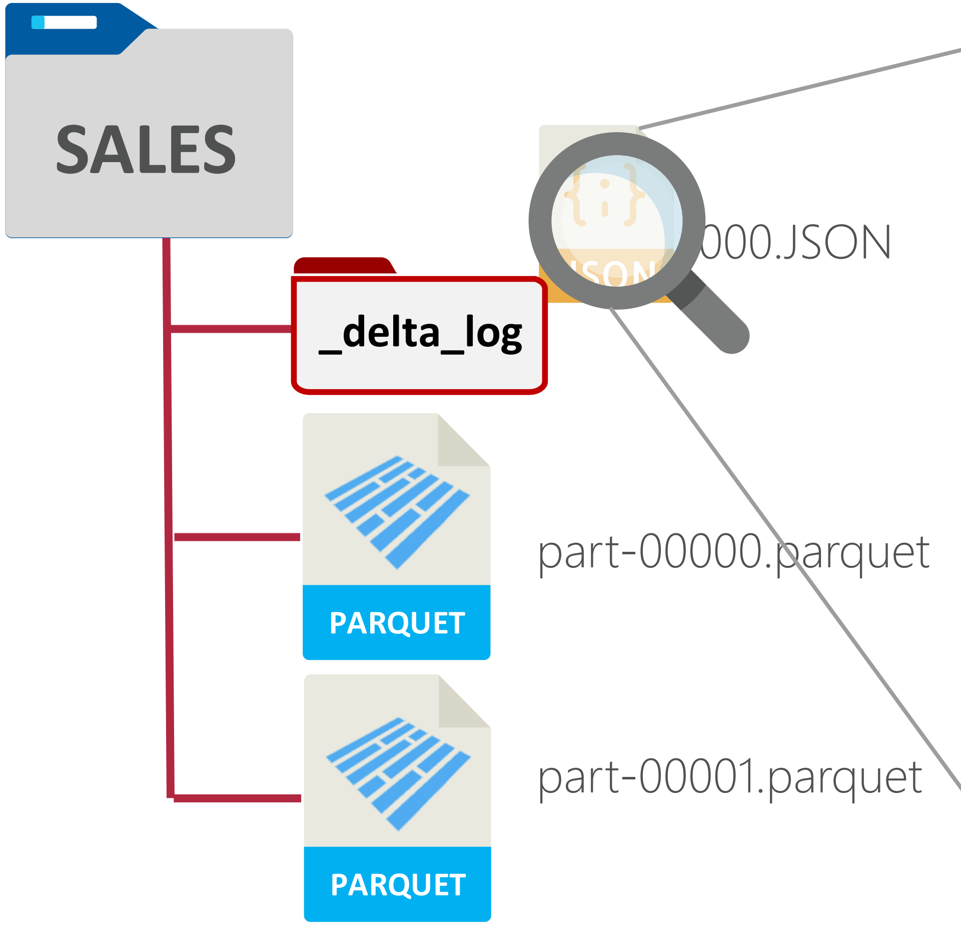
```
DELETE * FROM SALES WHERE Segment = 3
```





Only way to delete is to replace the existing files with a new file containing the non-deleted data

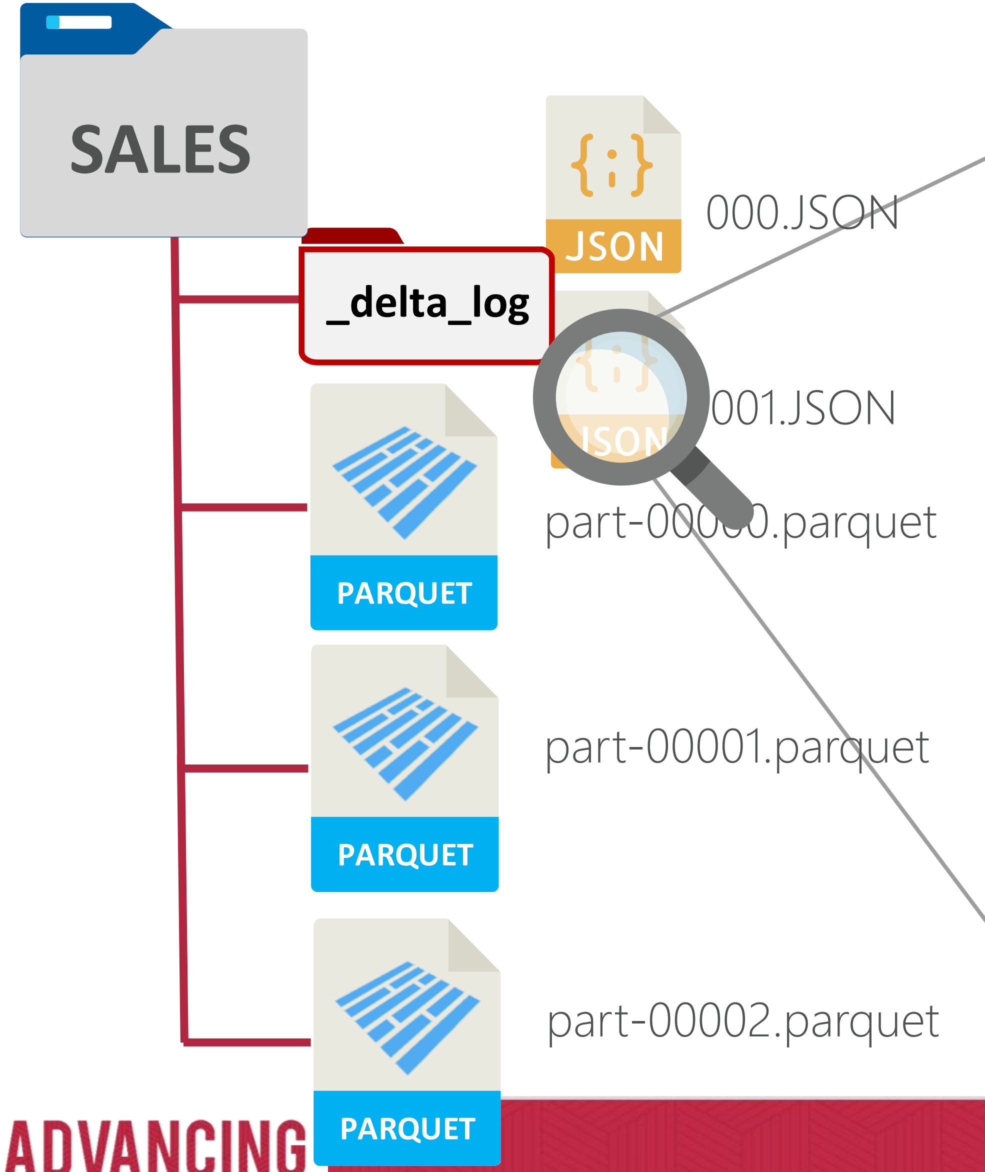
BUT WHAT ACTUALLY IS IT? - WITH DELTA:



```
…
{
   "add": {
      "path": "part-00000.parquet",
      "partitionValues": {},
      "size": 255520,
      "modificationTime": 1572823237000,
      "dataChange": true,
      "stats":[…]},
 "add": {
      "path": "part-00001.parquet",
      "partitionValues": {},
      "size": 242520,
      "modificationTime": 1572823237000,
      "dataChange": true,
      "stats": […]}
}
```

BUT WHAT ACTUALLY IS IT? - WITH DELTA:

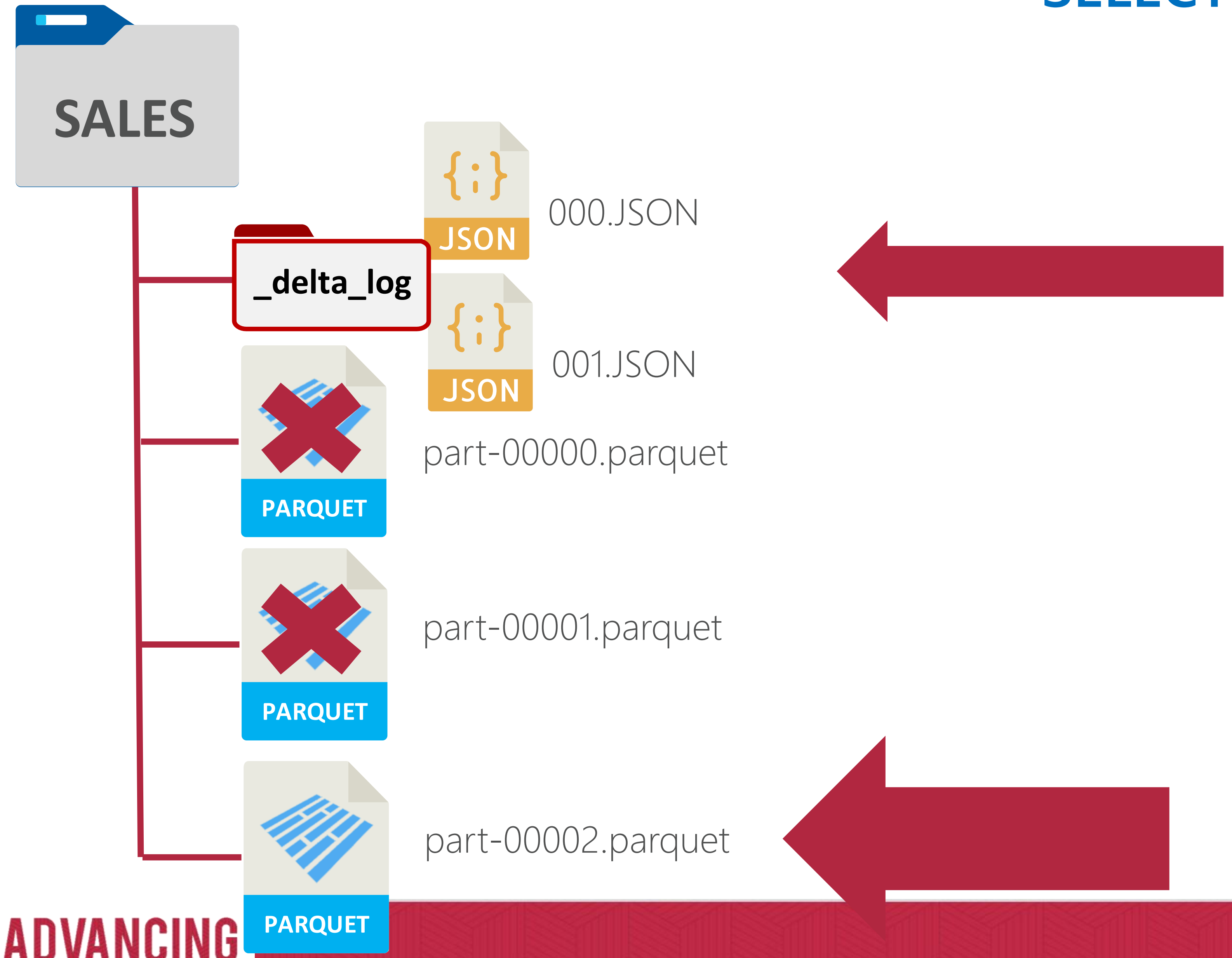
DELETE * **FROM** SALES **WHERE** Segment = 3

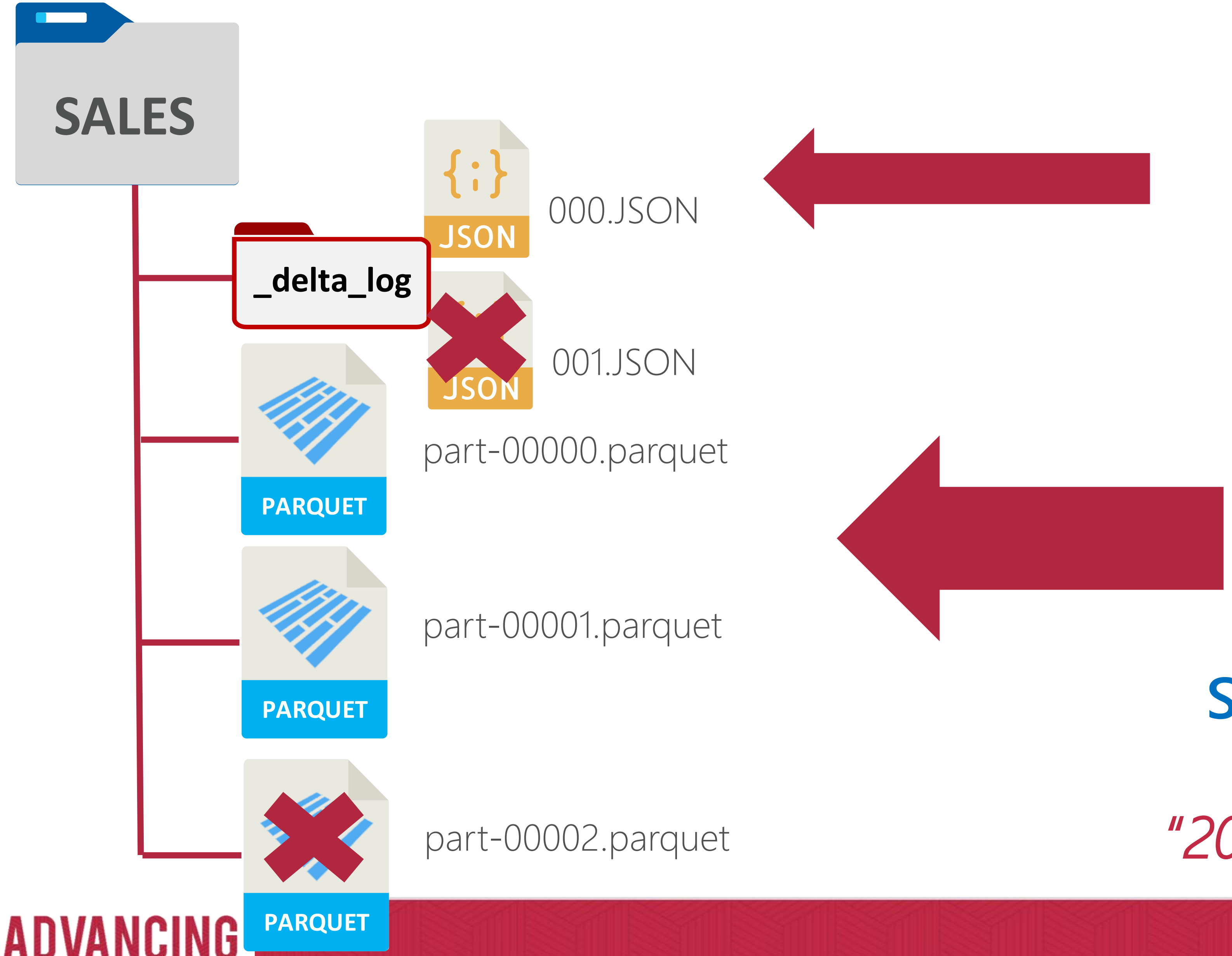


```
…
{
   "add": {
      "path": "part-00002.parquet",
      "partitionValues": {},
      "size": 255520,
      "modificationTime": 1572823237000,
      "dataChange": true,
      "stats":[…]},
 “remove": {
      "path": "part-00000.parquet",
       "modificationTime": 1572823237000,
      "dataChange": true},
 “remove": {
      "path": "part-00001.parquet",
       "modificationTime": 1572823237000,
      "dataChange": true}
}
```

ADVANCING ANALYTICS

SELECT * FROM SALES





SELECT * **FROM** SALES
TIMESTAMP AS OF
"2022-06-16T10:15:12.013Z"

TUTORIAL SETUP

Local Development

We're assuming Spark will be the main tool uses to interact with Delta. So you need a local spark environment first!

https://delta.io/learn/getting-started

Once your spark environment is configured, delta can be installed during initialisation:

```
pyspark --packages io.delta:delta-core_2.12:1.0.0 \
  --conf
"spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension" \
  --conf
"spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog"
```

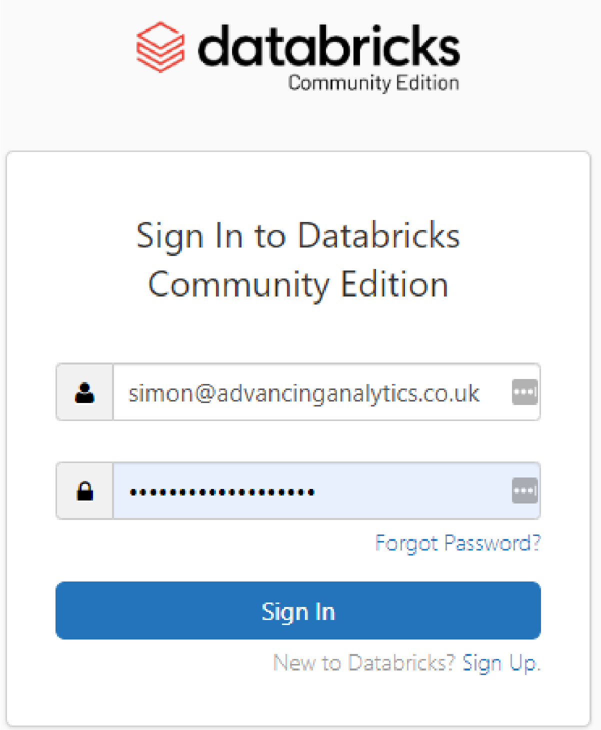
You will need to import delta within your pyspark scripts, then you can use it as you wish!

Hosted Sandbox

However… spark setup can be awkward and has several installation pre-requisites.

For today's examples, we can use a free sandbox version of the Databricks hosted spark to test out Delta features

https://databricks.com/try-databricks



databricks
Community Edition

Sign In to Databricks
Community Edition

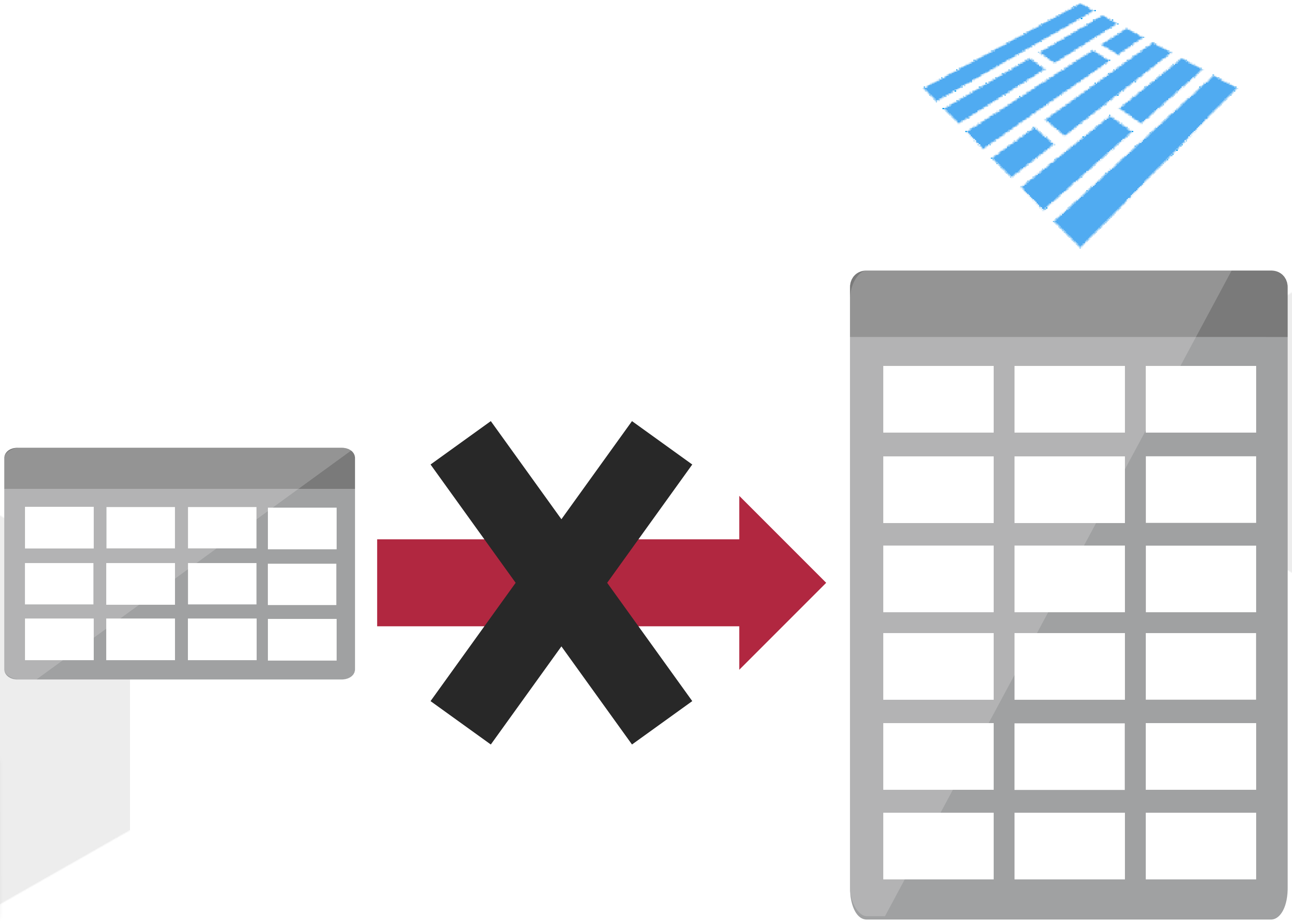
simon@advancinganalytics.co.uk

••••••••••••••••••••

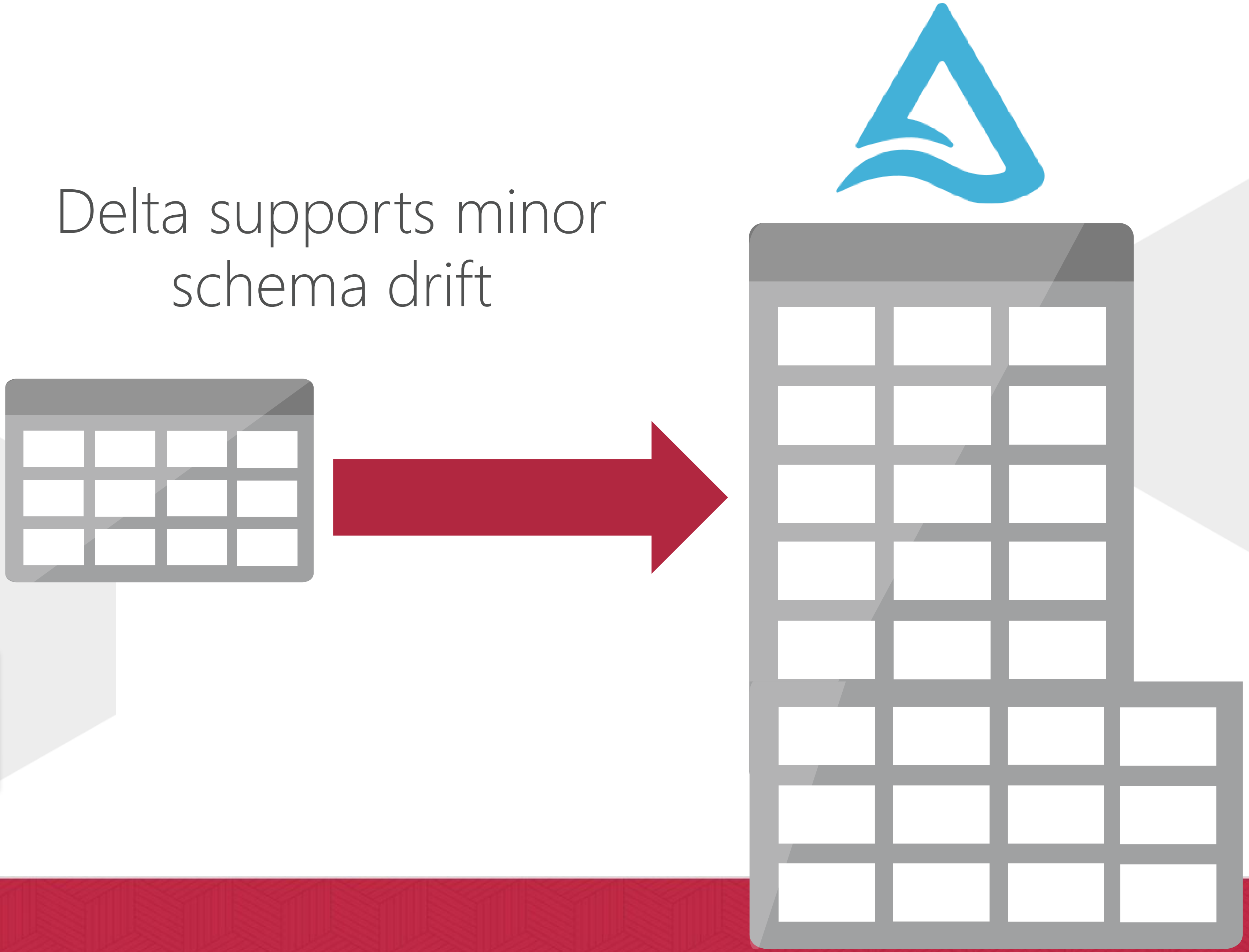
DEMO: WORKING WITH DELTA

- Creating a Delta Table
- The Delta Transaction Log
- Updating Delta Tables

KILLER DELTA FEATURES



Delta supports minor schema drift



DELTA MERGE

```
MERGE INTO <target_table>
USING <source_table>
ON <merge_condition>
[ WHEN MATCHED [ AND <condition> ] THEN <matched_action> ]
[ WHEN MATCHED [ AND <condition> ] THEN <matched_action> ]
[ WHEN NOT MATCHED [ AND <condition> ]  THEN <not_matched_action> ]

where

<matched_action>  =
  DELETE  |
  UPDATE SET *  |
  UPDATE SET column1 = value1 [, column2 = value2 ...]

<not_matched_action>  =
  INSERT *  |
  INSERT (column1 [, column2 ...]) VALUES (value1 [, value2 ...])
```

mlflow™

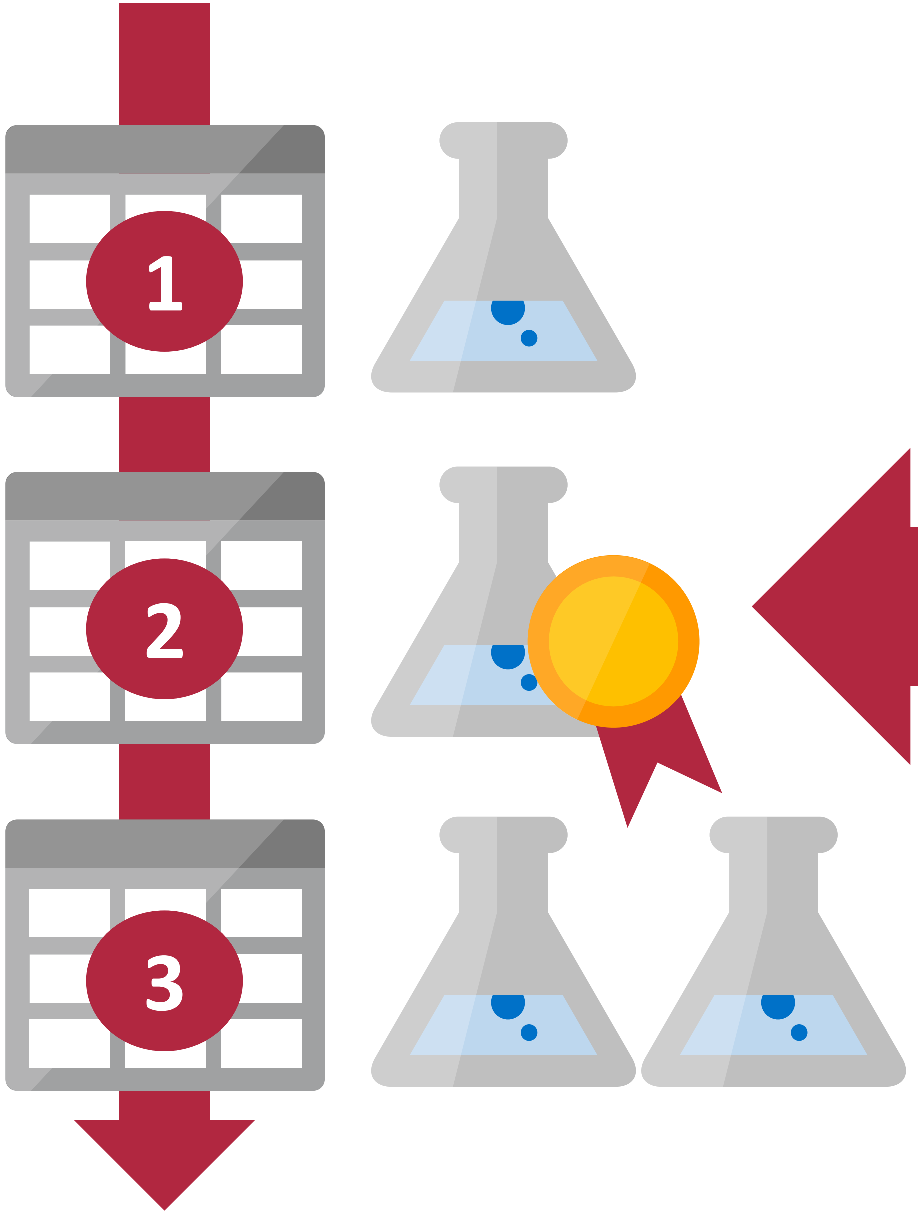


TABLE AS OF VERSION 2

Reduced need for multiple copies of data attached to evolving machine learning experiments

DEMO: DELTA UPDATES

- Schema Drift
- Merge into a Delta Table
- TIME TRAVEL

DELTA TABLES PERFORMANCE

AUDITING DELTA TABLES

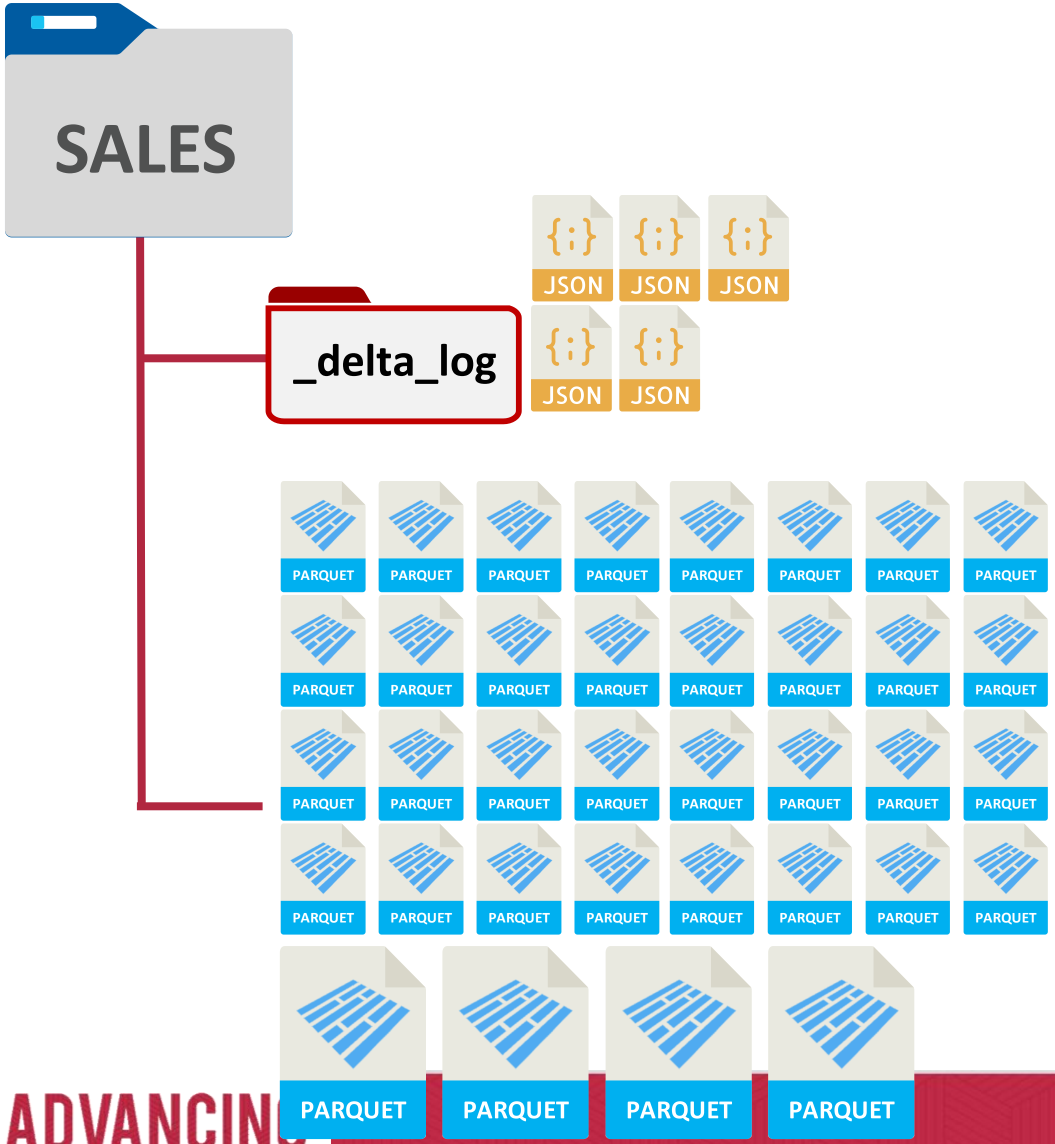
Delta Tables have additional commands for interrogating the transaction log:

Describe <Table> - Returns the table schema as usual

Describe Detail <Table> – Returns the underlying delta metadata, number of files, storage location etc

Describe History <Table> - Displays the most recent changes made to the table (inserts, updates, optimize, vacuum etc)

SMALL/INCREMENTAL UPDATES



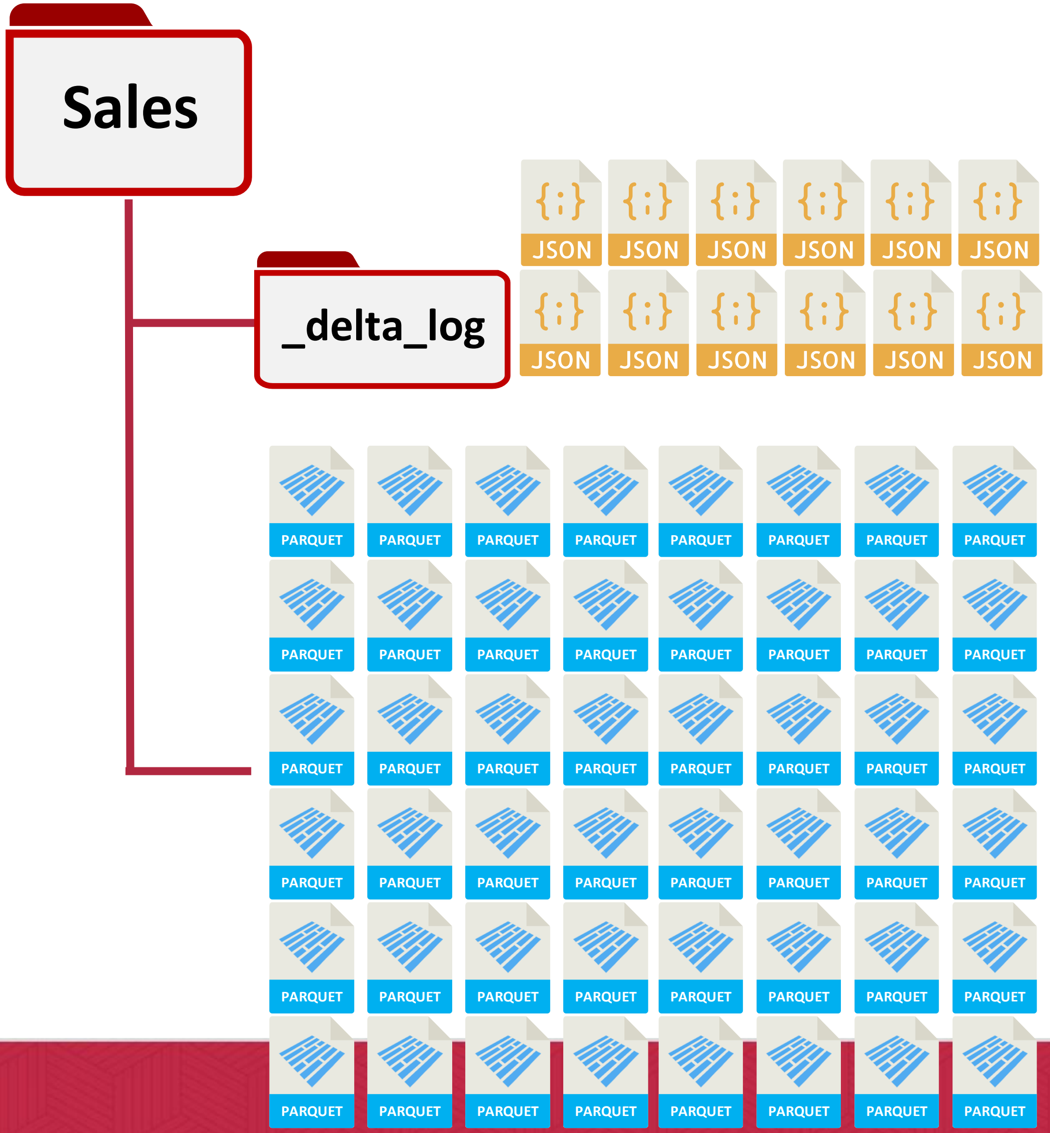
The Optimize command compacts small files into larger, better compressed files

This is treated like all other updates, files are NOT deleted

```
# Optimize a Delta Table in Spark SQL
OPTIMIZE [TableName]
```

ADVANCING ANALYTICS

OBSOLETE FILES



To remove obsolete history files, Delta has the VACUUM command

This command physically deletes data files older than a specified date

You CANNOT time travel past dates where history has been vacuumed

USING THE VACUUM COMMAND

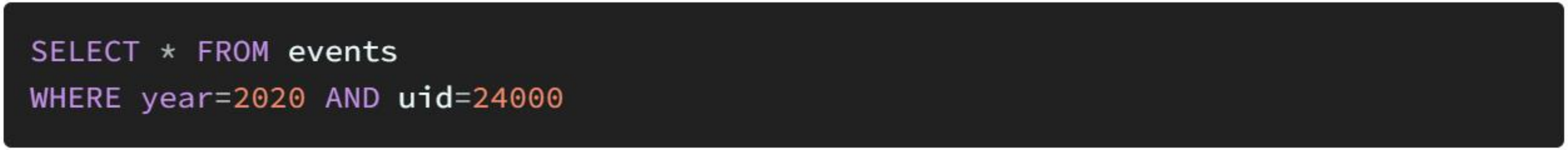
Vacuuming in SQL:

```
--Vacuum Table using defaults
VACUUM [database].[table]

--Vacuum using path not Hive table
VACUUM '/mnt/lake/BASE/myTable/'

--VACUUM for a non-default time period
VACUUM [database].[table] RETAIN 168 HOURS

--TEST THE VACUUM BEFORE YOU RUN IT
VACUUM [database].[table] RETAIN 168 HOURS DRY RUN
```

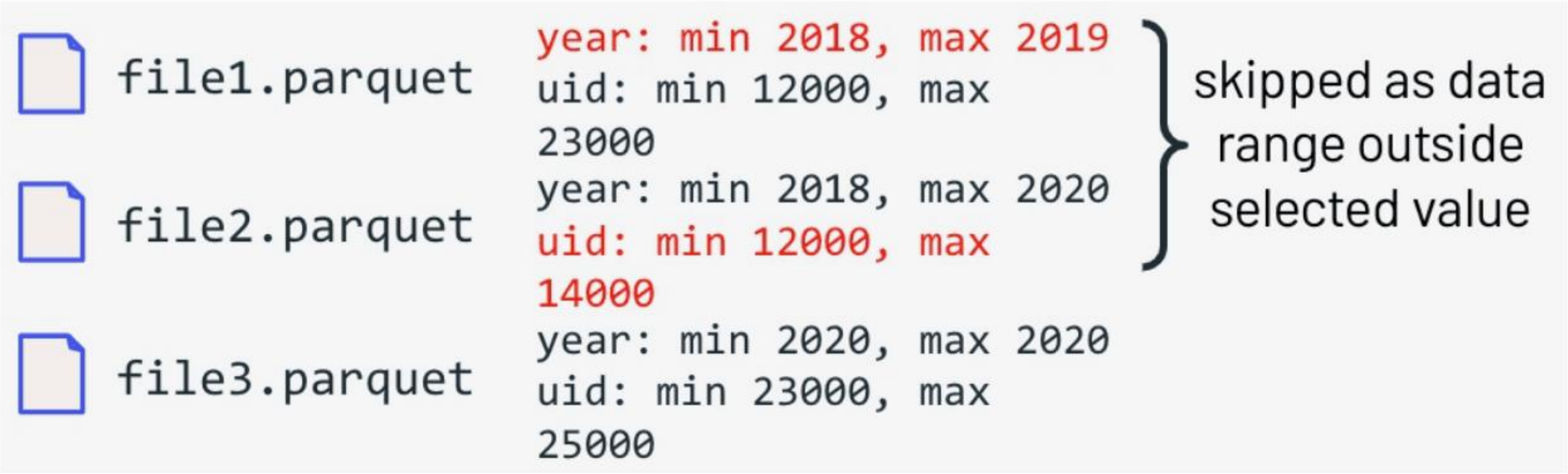
Using the python deltaTable object:

```
# Vacuum Table using defaults
deltaTable.vacuum()

# Vacuum Table for files older than 7 days (168 hours)
deltaTable.vacuum(168)
```

FILE SKIPPING



```sql
SELECT * FROM events
WHERE year=2020 AND uid=24000
```



file1.parquet

year: min 2018, max 2019
uid: min 12000, max 23000

file2.parquet

year: min 2018, max 2020
uid: min 12000, max 14000

file3.parquet

year: min 2020, max 2020
uid: min 23000, max 25000

skipped as data range outside selected value

DELTA MANAGEMENT

- View Delta Metadata
- Optimize a Delta Table
- Vacuum a Delta Table

THE DATA LAKEHOUSE

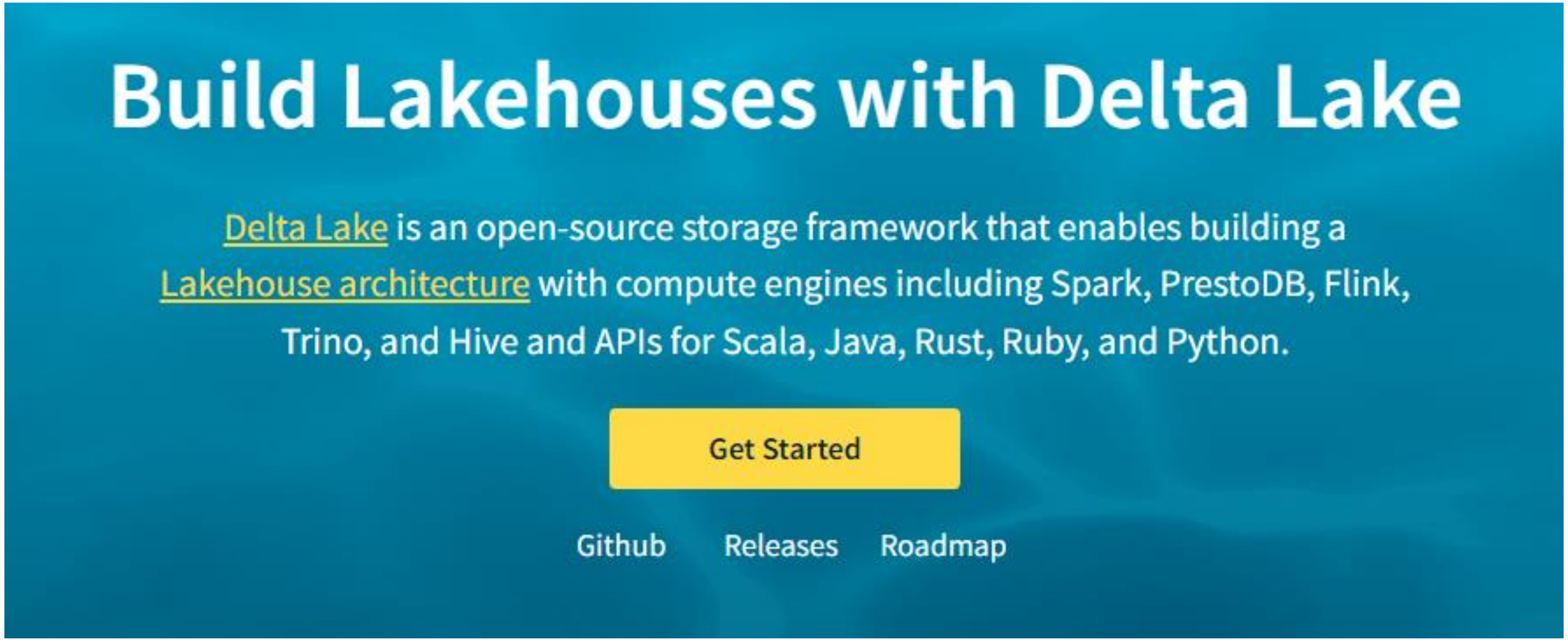
THE DATA LAKEHOUSE

- Structured
- Governed
- Familiar
- Fast
- Flexible
- Cheap
- Scalable

- ~~Small Files~~
- ~~Operational Complexity~~
- ~~Auditability~~
- ~~Indexing~~

DELTA LAKE

https://delta.io/



Build Lakehouses with Delta Lake

Delta Lake is an open-source storage framework that enables building a Lakehouse architecture with compute engines including Spark, PrestoDB, Flink, Trino, and Hive and APIs for Scala, Java, Rust, Ruby, and Python.

THANKS FOR LISTENING

Twitter: @MrSiWhiteley

youtube.com/c/AdvancingAnalytics

AdvancingAnalytics.co.uk