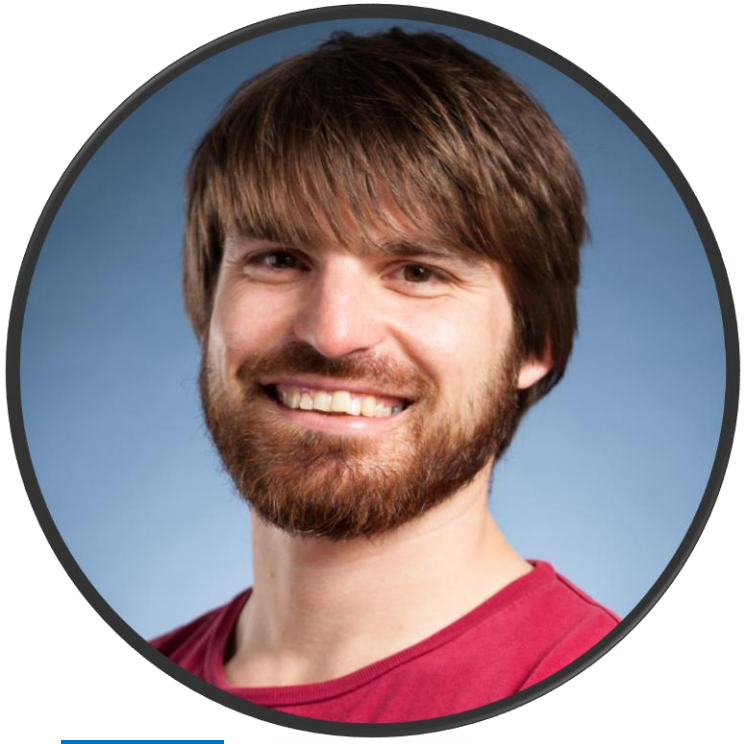
Introduction to Delta Lake

By Gerhard Brueckl



MVP
Microsoft
Most Valuable Professional

SSAS MAESTRO
by Microsoft

@gbrueckl

blog.gbrueckl.at

gerhard@gbrueckl.at

https://github.com/gbrueckl

DatabricksPS

Databricks VSCode

PowerBI Connector

DELTA LAKE

paiqo

www.paiqo.com

PASS SQL saturday VIENNA

AUSTRIA PASS

Agenda

- What is Delta Lake?
- How does it work?
- What do I need to watch out for?

Delta Lake – delta.io

Delta Lake is an open-source storage layer that brings ACID transactions to Apache Spark™ and big data workloads.

- ACID compliant transactions
  - Optimistic Concurrency Control
- Support for UPDATE / MERGE
- Time-Travel
- Schema enforcement and evolution
  - Across multiple files/folders
- Batch & Streaming
- 100% compatible with Apache Spark

Delta Lake – delta.io

- Self-contained table
  - Meta-data / schema
  - Partitions
  - Data
- Can reside on any storage
- Based on Apache Parquet and JSON
- Transaction isolation on partition level
- Native support in Databricks / Spark (Azure Synapse, …)
- Restore of previous version, cloning of tables
- APIs for SQL, Python, Scala, …

Delta Lake – delta.io

_delta_log

- FactInternetSales_part.delta
  - _delta_log
    - _last_checkpoint
    - 00000000000000000000.crc
    - 00000000000000000000.json
    - 00000000000000000001.crc
    - 00000000000000000001.json
    - •••
    - 00000000000000000009.crc
    - 00000000000000000009.json
    - 00000000000000000010.checkpoint.parquet
    - 00000000000000000010.crc
    - 00000000000000000010.json
    - 00000000000000000011.crc
    - 00000000000000000011.json
  - SalesTerritoryKey=1
  - SalesTerritoryKey=2

data

- FactInternetSales_part.delta
  - _delta_log
  - SalesTerritoryKey=1
    - part-00000-1b1c:ae75.c000.snappy.parquet
    - part-00000-dcd82f-:.c000.snappy.parquet
    - part-00000-e31d3b88.c000.snappy.parquet
    - part-00000-f454d11-‹.c000.snappy.parquet
    - part-00001-2a5b77e-.c000.snappy.parquet
    - part-00001-2ba13e9?.c000.snappy.parquet
    - part-00001-5f6d9d47.c000.snappy.parquet
    - part-00001-9e8573-‹.c000.snappy.parquet

DDL Operations – CREATE TABLE AS SELECT

```sql
CREATE TABLE IF NOT EXISTS DimProduct
USING DELTA
PARTITIONED BY (ProductSubcategoryKey)
LOCATION '/mnt/adls/tables/DimProduct'
TBLPROPERTIES ('myKey' = 'myValue')
AS SELECT *
FROM stage.Products
```

- Column definitions are taken from SELECT

DDL Operations – CREATE TABLE (existing)

```
CREATE TABLE IF NOT EXISTS DimProduct
USING DELTA
LOCATION '/mnt/adls/tables/DimProduct'
```

- Avoid defining any additional information – this is all handled by the delta log!
  - Everything within the delta-folder is self-contained!
  - If you still define it, it has to match the existing information in the delta log

DML Operations – Delta Lake - MERGE

```
MERGE INTO targetTable
USING updatesTable
        ON targetTable.year = 2021
        AND targetTable.id = updatesTable.id
WHEN MATCHED THEN
        UPDATE SET targetTable.value = updatesTable.value
WHEN NOT MATCHED THEN
        INSERT (id, year, value)
        VALUES (updatesTable.id, updatesTable.year, updatesTable.value)
```

- UPDATE, INSERT and DELETE
  - in one statement
  - In one transaction

DML Operations – Delta Lake - UPDATE

User

Product	Price
Notebook	900 €
PC	1,500 €
Tablet	500 €

```
UPDATE TABLE DimProduct
SET Price = 1300
WHERE Product = 'PC'
```

Product	Price
Notebook	900 €
PC	1,300 €
Tablet	500 €

_delta_log

000000000.json

```
"add": {
        "path": "part-01.parquet",
        ...
}
```

000000001.json

```
"remove": { "path": "part-01.parquet", ... },
"add": { "path": "part-02.parquet", ... }
```

Storage

Parquet
part-01
(3 rows)

Parquet
part-01
(3 rows)

Parquet
part-02
(3 rows)

DML Operations - Delta Lake – DELETE

User

Product	Price
Notebook	900 €
PC	1,300 €
Tablet	500 €

```
DELETE FROM DimProduct
WHERE Product = 'PC'
```

Product	Price
Notebook	900 €
Tablet	500 €

_delta_log

000000000.json

```
"add": {
    "path": "part-01.parquet",
... }
```

000000001.json

```
"remove": {
    "path": "part-01.parquet", ... },
"add": {
    "path": "part-02.parquet", ... }
```

000000002.json

```
"remove": { "path": "part-02.parquet", ... },
"add": { "path": "part-03.parquet", ... }
```

Storage

part-01
(3 rows)

part-02
(3 rows)

part-01
(3 rows)

part-02
(3 rows)

part-03
(2 rows)

DML Operations - Delta Lake – INSERT

User

Product	Price
Notebook	900 €
PC	1,300 €
Tablet	500 €

```
INSERT INTO DimProduct
VALUES ('Monitor', 200)
```

Product	Price
Notebook	900 €
Tablet	500 €
Monitor	200 €

_delta_log

000000000.json

000000001.json

"remove": {
"path": "part-01.parquet", ... },
"add": {
"path": "part-02.parquet", ... }

000000002.json

"remove": { "path": "part-02.parquet", ... },
"add": { "path": "part-03.parquet", ... }

000000003.json

"add": { "path": "part-04.parquet", ... }

Storage


Parquet
part-01
(3 rows)


Parquet
part-02
(3 rows)


Parquet
part-03
(2 rows)


Parquet
part-01
(3 rows)


Parquet
part-02
(3 rows)


Parquet
part-03
(2 rows)


Parquet
part-04
(1 row)

DML Operations - Delta Lake – VACUUM

User

Product	Price
Notebook	900 €
PC	1,300 €
Tablet	500 €

```
VACUUM DimProduct
```

Product	Price
Notebook	900 €
Tablet	500 €
Monitor	200 €

_delta_log

000000000.json
000000001.json

000000002.json

"remove": {
"path": "part-02.parquet", ... },
"add": {
"path": "part-03.parquet", ... }

000000003.json

"add": {
"path": "part-04.parquet",
... }

Storage

Parquet
part-01
(3 rows)

Parquet
part-02
(3 rows)

Parquet
part-03
(2 rows)

Parquet
part-03
(2 rows)

Parquet
part-04
(1 row)

Delta Lake – UPDATE/DELETE/MERGE

- Operations are logged in _delta_log
  - Old files are removed/invalidated (logically)
  - New files are added/referenced

- Conflicts have to be handled by the User!

- Always results in new, duplicated files! Even a `DELETE` can!

Can create A LOT of files!

Link

Delta Lake – Storage Consumption

Each DML Operation can potentially double the storage consumption (if all files are touched)

- UPDATE/DELETE/MERGE
- OPTIMIZE (!)

VACUUM is mandatory !

It makes sense to monitor the table size and storage consumption using a script and storing the results in a table