SharePoint Lists specific

Introduction

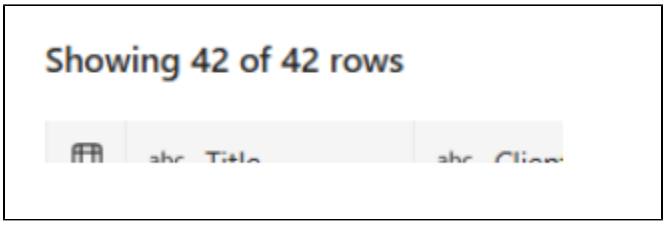
Microsoft Fabric provides native support for Delta Lake tables within Lakehouses, including time travel and versioning capabilities. However, when working with **mirrored tables (CDC via Mirroring)**, these capabilities are not directly exposed in the Fabric UI as they are in standard Lakehouse tables.

This document summarizes how to access:

- The underlying Delta table data
- The physical storage (Parquet + metadata)
- Historical versions of the table using Spark

0. Context of the test

- I see 42 rows because I actually have 42 rows in the list (first image)



- however, the replication shows 44 rows because it appears to have stored the changes (second image):
    - “The cumulative count of replicated rows, including all inserts, updates, and deletes applied to the target table.”

Is there a way to see those 44 lines (including updates and deletions) instead of the 42?

1. Reading the Delta Table (Current Version)

Even for mirrored tables, the data is stored as a standard Delta table in OneLake and can be accessed via its ABFSS path.

Delta table - data

```
path = "abfss://69300957-941f-4c8a-970f-49b3fce16e0d@onelake.dfs.fabric.microsoft.com/f73eb213-d657-4e43-8856-30a30f7cb1bf/Tables/dbo/RefSujetsIAAURA"
df = spark.read.format("delta").load(path)

display(df)
```

This returns the **latest state** of the table, equivalent to what is visible in Fabric after synchronization.

2. Exploring Underlying Storage (Metadata & Files)

The Delta table is physically composed of:

- Parquet data files
- Transaction log (_delta_log)
- Additional metadata (indexes, deletion vectors)

You can list these files using:

```
path = "abfss://69300957-941f-4c8a-970f-49b3fce16e0d@onelake.dfs.fabric.microsoft.com/f73eb213-d657-4e43-8856-30a30f7cb1bf/Tables/dbo/RefSujetsIAAURA"
files = spark._jvm.org.apache.hadoop.fs.FileSystem \
    .get(spark._jsc.hadoopConfiguration()) \
    .listStatus(spark._jvm.org.apache.hadoop.fs.Path(path))

for f in files:
    print(f.getPath().toString())
```

Example

- abfss://69300957-941f-4c8a-970f-49b3fce16e0d@onelake.dfs.fabric.microsoft.com/f73eb213-d657-4e43-8856-30a30f7cb1bf/Tables/dbo/RefSujetsIAAURA/_delta_log
- abfss://69300957-941f-4c8a-970f-49b3fce16e0d@onelake.dfs.fabric.microsoft.com/f73eb213-d657-4e43-8856-30a30f7cb1bf/Tables/dbo/RefSujetsIAAURA/_index_bin
- abfss://69300957-941f-4c8a-970f-49b3fce16e0d@onelake.dfs.fabric.microsoft.com/f73eb213-d657-4e43-8856-30a30f7cb1bf/Tables/dbo/RefSujetsIAAURA/deletion_vector_e7de9379-f721-4637-8585-2d37ec53ad1b.bin
- abfss://69300957-941f-4c8a-970f-49b3fce16e0d@onelake.dfs.fabric.microsoft.com/f73eb213-d657-4e43-8856-30a30f7cb1bf/Tables/dbo/RefSujetsIAAURA/part-00000-3e4b092f-2246-4b05-a541-10d61f776d3b.c000.zstd.parquet
- abfss://69300957-941f-4c8a-970f-49b3fce16e0d@onelake.dfs.fabric.microsoft.com/f73eb213-d657-4e43-8856-30a30f7cb1bf/Tables/dbo/RefSujetsIAAURA/part-00000-a9a515f8-02ab-48c9-8fb9-9688eaf4567a.c000.zstd.parquet
- abfss://69300957-941f-4c8a-970f-49b3fce16e0d@onelake.dfs.fabric.microsoft.com/f73eb213-d657-4e43-8856-30a30f7cb1bf/Tables/dbo/RefSujetsIAAURA/metadata

Example output:

_delta_log/

_index_bin/

deletion_vector_*.bin

part-*.parquet

metadata/

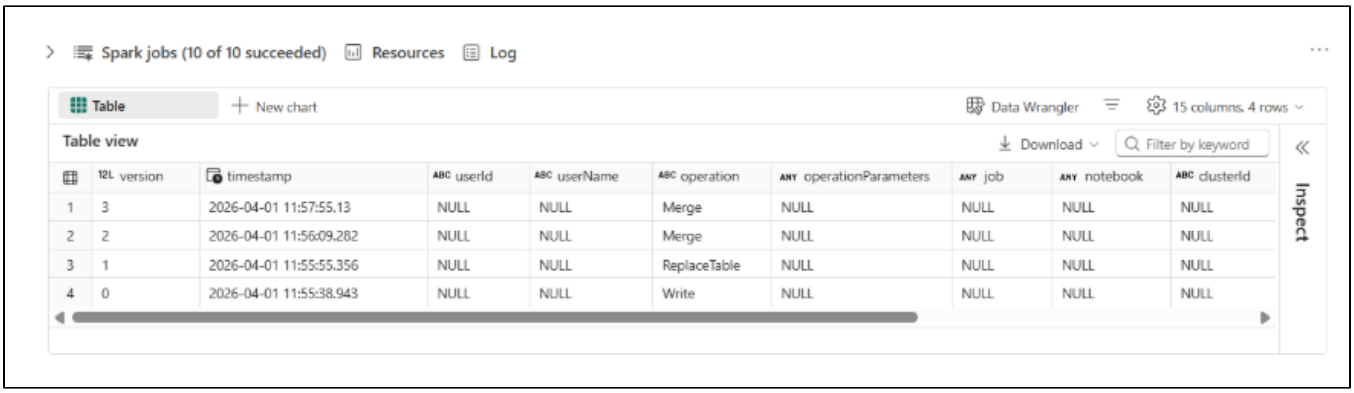
Key components:

- _delta_log/ transaction history (versions)
- part-*.parquet actual data
- deletion_vector logical deletes (CDC optimization)

3. Accessing Historical Versions (Time Travel)

Although not exposed in the Fabric UI for mirrored tables, Delta Lake versioning is still fully available via Spark.

```
from delta.tables import DeltaTable
from pyspark.sql import functions as F

path = "abfss://69300957-941f-4c8a-970f-49b3fce16e0d@onelake.dfs.fabric.microsoft.com/f73eb213-d657-4e43-8856-30a30f7cb1bf/Tables/dbo/RefSujetsIAAURA"


# 1) Historique Delta
delta_table = DeltaTable.forPath(spark, path)
history_df = delta_table.history()  # reverse chronological order

display(history_df)

# 2) Liste des versions disponibles, triées dans l'ordre croissant
versions = [
    row["version"]
    for row in history_df.select("version").distinct().orderBy("version").collect()
]print("Versions disponibles :", versions)

# 3) Création d'un DataFrame par version
dfs_by_version = {}

for v in versions:
    df_v = (
        spark.read
        .format("delta")
        .option("versionAsOf", v)
        .load(path)
        .withColumn("_delta_version", F.lit(v))
    )
    dfs_by_version[v] = df_v


print(f"{len(dfs_by_version)} DataFrames créés")
print("Exemple version la plus récente :", max(dfs_by_version.keys()))
```



	12L version	timestamp	ABC userId	ABC userName	ABC operation	ANY operationParameters	ANY job	ANY notebook	ABC clusterId
1	3	2026-04-01 11:57:55.13	NULL	NULL	Merge	NULL	NULL	NULL	NULL
2	2	2026-04-01 11:56:09.282	NULL	NULL	Merge	NULL	NULL	NULL	NULL
3	1	2026-04-01 11:55:55.356	NULL	NULL	ReplaceTable	NULL	NULL	NULL	NULL
4	0	2026-04-01 11:55:38.943	NULL	NULL	Write	NULL	NULL	NULL	NULL

Example

Version 2 :

```
display(dfs_by_version[2])
```

42	TCI	Syensqo	Mirroring S...	2026	Projet	NULL	SyensqoRLS

Version 3 - Updated made on one field (most recent version so equivalent from delta table synchronization) most recent version

```
display(dfs_by_version[3])
```

42	TCI	Syensqo	Mirroring S...	2026	Projet	Fini	SyensqoRLS

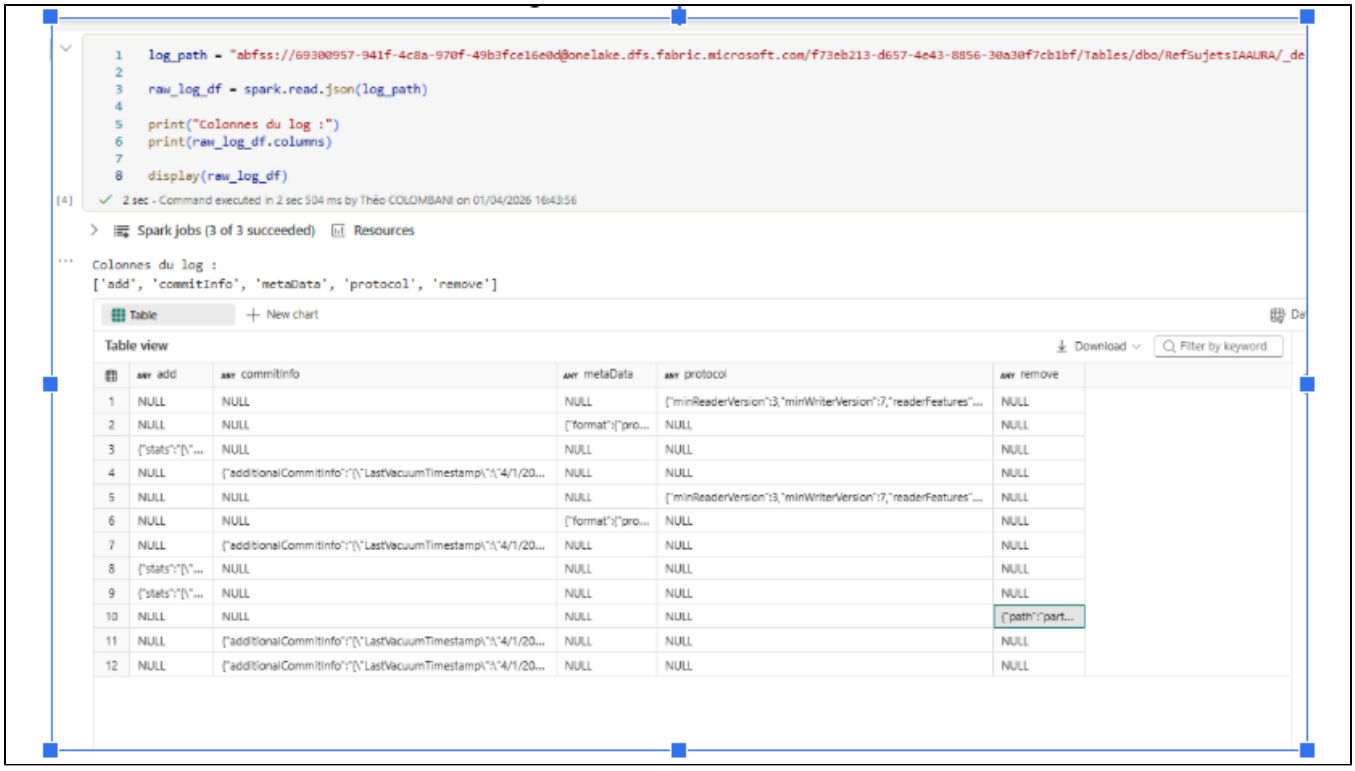
4. Key Observations

- Mirrored tables **are standard Delta tables under the hood**

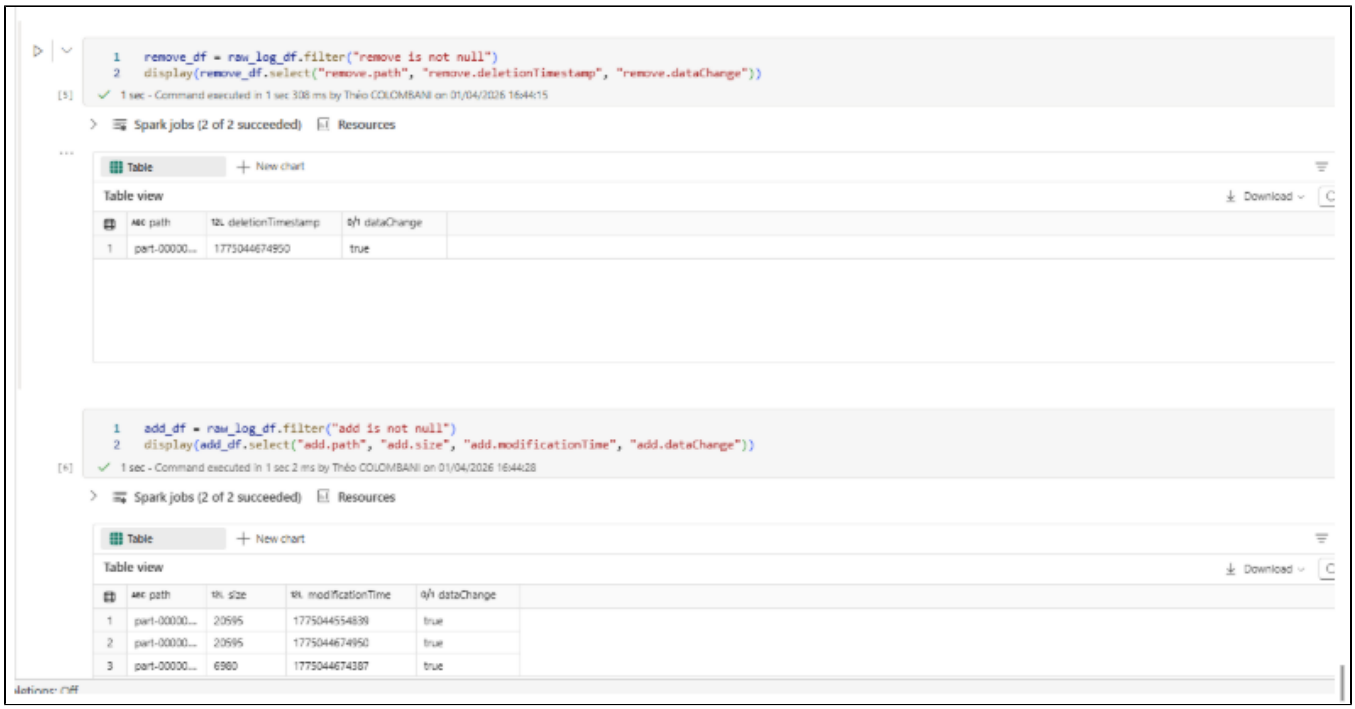
But UI exposed only the last version (SCD1 type (owerwritte))

- All Delta capabilities (time travel, versioning) remain accessible via Spark
- The Fabric UI does **not expose version history** for mirrored tables
- Each version represents a **full snapshot**, not just incremental changes, no change lo apparenlty
- CDC changes are internally managed through:
    - Delta commits
    - Deletion vectors
    - Transaction logs

It seems we can access to commit logs



But without testing more tok now which row was deleted or added


remove_df = raw_log_df.filter("remove is not null")
display(remove_df.select("remove.path", "remove.deletionTimestamp", "remove.dataChange"))
[5] 1 sec - Command executed in 1 sec 308 ms by Théo COLOMBANI on 01/04/2026 16:44:15
Spark jobs (2 of 2 succeeded) Resources
Table
New chart
Table view
Download
path
deletionTimestamp
dataChange
1 part-00000... 1775044674950 true
add_df = raw_log_df.filter("add is not null")
display(add_df.select("add.path", "add.size", "add.modificationTime", "add.dataChange"))
[6] 1 sec - Command executed in 1 sec 2 ms by Théo COLOMBANI on 01/04/2026 16:44:28
Spark jobs (2 of 2 succeeded) Resources
Table
New chart
Table view
Download
path
size
modificationTime
dataChange
1 part-00000... 20595 1775044554839 true
2 part-00000... 20595 1775044674950 true
3 part-00000... 6960 1775044674387 true