Apache Iceberg Internals

What Changes During Each Script Action

Tick Data Lakehouse — Trino + MinIO + Iceberg
Reference: iceberg_maintenance.py • iceberg_time_travel.py

1. Iceberg Folder Structure in MinIO

An Iceberg table is not a database table — it is a folder in object storage with a strict layout. Trino reads metadata files to discover which data files are active for the current snapshot. Queries never scan the folder directly.

```
gold/btcusd/
metadata/
version-hint.text <- entry point, contains '3'
v1.metadata.json <- empty table after CREATE TABLE
v2.metadata.json <- after first INSERT
v3.metadata.json <- after rollback
snap-aaa111.avro <- manifest list for snapshot 1
snap-bbb222.avro <- manifest list for snapshot 2
manifest-ccc333.avro <- lists data files for snapshot 2
data/
year=2023/
0001-abc.parquet <- actual tick data
year=2024/
0001-def.parquet <- actual tick data
```

The metadata/ folder is Iceberg. The data/ folder is just Parquet. Any Parquet-aware tool (DuckDB, PyArrow, pandas) can read data/ directly without knowing anything about Iceberg.

2. What Each File Type Does

File	Location	Purpose	Grows when
`version-hint.text`	metadata/	Single line: current metadata version number. Entry point for all readers.	Every write operation — even rollback
`vN.metadata.json`	metadata/	Full table state: schema, partition spec, current snapshot ID, snapshot log.	Every write operation
`snap-N.avro (manifest list)`	metadata/	List of all active manifest files for a snapshot. One per snapshot that touches data.	INSERT, DELETE, compaction only
`manifest-N.avro (manifest file)`	metadata/	List of actual data files with stats (row counts, min/max per column).	INSERT, DELETE, compaction only
`data/year=N/*.parquet`	data/	Actual tick data. Partitioned by year(datetime). Never mutated in place.	INSERT, compaction (rewrites)

3. File Changes Per Script Action

The table below shows exactly which files are created, rewritten, or deleted for each action in the maintenance and time travel scripts. +1 NEW means one new file is written. — means that file type is not touched.

Action	version-hint .text	vN.metadata .json	snap-N.avro (manifest list)	manifest-N .avro	data/year=N *.parquet
`CREATE TABLE`	**+1 NEW pointer=v1**	**+1 NEW empty schema**	**+1 NEW empty list**	—	—
`INSERT data`	**+1 NEW pointer=v2**	**+1 NEW new snapshot**	**+1 NEW new list**	**+1 NEW new entries**	**+1 NEW new files**
`Rollback`	**+1 NEW pointer=vN**	**+1 NEW points to old snap**	—	—	—
`expire_snapshots()`	**+1 NEW pointer=vN**	**+1 NEW records expiry**	—	—	—
`optimize() compaction`	**+1 NEW pointer=vN**	**+1 NEW new snapshot**	**+1 NEW new list**	**+1 NEW new entries**	**+1 NEW rewritten**
`remove_orphan_files()`	**+1 NEW pointer=vN**	**+1 NEW records cleanup**	—	**DELETES unreferenced .avro deleted**	**DELETES unreferenced .parquet deleted**
`Query (any)`	—	—	—	—	—
`FOR VERSION AS OF`	—	—	—	—	—

Key insight: rollback, expire_snapshots, and queries only ever touch metadata. Parquet data files are immutable — they are only written by INSERT and rewritten by compaction. They are only deleted by remove_orphan_files.

4. Rollback Deep Dive

A rollback is the most misunderstood operation. It does not delete data. It writes two new files and nothing else:

```
1. A new vN.metadata.json is written.
This file says: 'current snapshot = '
2. version-hint.text is updated to point at the new vN.
This is the only file that gets overwritten rather than appended.
```

File state before and after rollback:

File	Before Rollback	After Rollback	Physically deleted?
`version-hint.text`	contains '3'	contains '4'	No — overwritten
`v3.metadata.json`	current	still exists, now old	No
`v4.metadata.json`	does not exist	NEW — created	No
`snap-xyz.avro`	active	still exists	No
`manifest-abc.avro`	active	still exists	No
`data/year=2023/*.parquet`	exists	still exists	No
`data/year=2024/*.parquet`	exists	still exists	No

Why MinIO still shows all your data after rollback

MinIO is a file store — it has no knowledge of Iceberg snapshots. When you browse the bucket you will always see every parquet file that was ever written, regardless of rollback. Only Trino follows the snapshot chain and filters what it returns.

To physically remove files after rollback:

```
Step 1: expire_snapshots(retention_threshold => '7d')
Marks the unreachable snapshots as expired in metadata.
Does NOT delete any files yet.

Step 2: remove_orphan_files(retention_threshold => '24h')
Walks MinIO, compares every file against active manifests,
and deletes any file not referenced by a live snapshot.
This is the only operation that physically removes parquet and avro files.
```

Production recommendation: never run remove_orphan_files automatically after a rollback. Keep a 7-30 day window so you can roll forward again if needed. For financial tick data, consider archiving to cold storage instead of deleting.

5. Why JSON Files Keep Growing

Every single write operation — including rollback — appends a new vN.metadata.json. Iceberg never overwrites old metadata files (except version-hint.text). This is intentional: it allows concurrent readers to safely read an old snapshot while a writer commits a new one.

Operation	JSON files added	Cumulative after 10x
CREATE TABLE	`1 (v1)`	1
INSERT x10	`1 per insert`	11
Rollback x3	`1 per rollback`	14
expire_snapshots()	`1 (records expiry)`	15
remove_orphan_files()	`1 (records cleanup)`	16 — but old ones deleted

5b. When Avro Files Grow

Avro manifest files only grow when the set of active parquet data files changes. A rollback moves the metadata pointer but the set of parquet files does not change, so no new avro is written. Compaction rewrites parquet files, so it always triggers new avro manifests.

Operation	Avro written?	Reason
CREATE TABLE	`Yes — empty manifest list`	Snapshot created, even if empty
INSERT	`Yes — new manifest list + file`	New parquet files added to manifest
Rollback	`No`	Parquet file set unchanged
expire_snapshots()	`No`	Parquet file set unchanged
optimize() compaction	`Yes — new manifest list + file`	Old parquet replaced with new merged files
remove_orphan_files()	`No new — deletes old`	Removes unreferenced avro from disk

6. Production Maintenance Checklist

Recommended schedule for a tick data lakehouse where data is loaded in batches and queries are read-heavy.

Frequency	Action	What it cleans	Safe to automate?
After every load	`verify_partitions()`	Confirms row counts per year are correct	Yes
Weekly	`optimize()`	Merges small parquet files into 128MB files	Yes
Weekly	`expire_snapshots(30d)`	Marks snapshots older than 30 days as expired	Yes
Manual only	`remove_orphan_files()`	Physically deletes unreferenced parquet + avro from MinIO	No — confirm rollback window is closed first
After rollback	`reconnect + verify_partitions()`	Confirms Trino sees the rolled-back state	Yes

Core rule: JSON files grow with every operation. Avro files only grow when parquet files change. Parquet files are immutable — they are only ever appended or rewritten, never edited in place.