Horkan

a blog by Wayne Horkan

From SCD2 Bronze to a Non-SCD Silver Layer in Databricks

No Comments

This article explains a best-practice Databricks lakehouse pattern for transforming fully historical SCD2 Bronze data into clean, non-SCD Silver tables. Bronze preserves complete temporal truth for audit, compliance, and investigation, while Silver exposes simplified, current-state views optimised for analytics and data products. Using Delta Lake features such as MERGE, Change Data Feed, OPTIMIZE, and ZORDER, organisations, particularly in

regulated Financial Services, can efficiently maintain audit-proof history while delivering fast, intuitive, consumption-ready datasets.

Contents

Table of Contents

Introduction: How to Transform Fully Historical Data into Clean, Simple, Current-State Views

Modern data platforms increasingly use a **layered lakehouse architecture**, especially in regulated sectors like Financial Services. One common pattern has emerged as a clear best-practice approach:

- **Bronze Layer:** Full **SCD2 history** for every meaningful change
- **Silver Layer:** Clean, simple, **non-SCD** (often latest-record-only) data for analytics and Data Products

This pattern offers the best of both worlds:
audit-proof history and **easy-to-use data**.

In Databricks, Delta Lake makes this pattern highly effective thanks to ACID transactions, Delta MERGE, and powerful optimisation features.
But making the transition from SCD2 Bronze to a non-SCD Silver layer requires deliberate design.

This article explains the **why**, the **how**, and the **best practices** behind this transformation.

This is part of a related series of articles on using SCD2 at the bronze layer of a medallion-based data platform for highly regulated Financial Services (such as the UK). This Databricks-focused article shows how to transform full-history SCD2 Bronze into clean, current-state Silver for regulated FS. Databricks practitioners, data modellers, and architects learn Delta features that make Silver performant and compliant, delivering fast consumption without losing auditability, and enabling practical analytics on "manage it early" temporal foundations.

1. Why Move from SCD2 Bronze to a Non-SCD Silver Layer?

A layered lakehouse architecture is not just about storage efficiency—it is about serving fundamentally different user needs from the same trusted data foundation. While SCD2 history is essential for traceability and compliance, it is often cumbersome for everyday analytical use. This section explains why separating a fully historical Bronze layer from a simplified Silver layer is both a practical and strategic design decision, particularly in regulated environments.

An SCD2 Bronze layer preserves *everything*.
It is the canonical truth of what changed, when it changed, and why it changed.

But full history is not always the right format for users.

1.1 Bronze = Temporal Truth

The Bronze layer captures:

- Complete audit history
- Point-in-time views
- All versioned changes
- Records needed for FCA/PRA compliance
- Evidence for AML/KYC investigations

- Accurate historical reconstruction

It is the system of **record**.

1.2 Silver = Analytical Simplicity

Most downstream analytics, dashboards, and ML models do not want historical versions.
They need:

- Clean tables
- One row per business entity
- No effective dates
- No version flags
- Current values only
- Lightweight join logic

The Silver layer is the system of **insight**.

1.3 Why separate them?

Because the needs of auditors and investigators aren't the same as those of analysts, data scientists, or dashboards.

The **Bronze layer stores history**.
The **Silver layer exposes the current state**.

Databricks supports this separation elegantly.

2. How Databricks Enables This Pattern

Not all platforms make it easy to balance deep historical lineage with high-performance analytics. Databricks, through Delta Lake, provides native capabilities that remove much of the complexity traditionally associated with SCD2 processing and downstream consumption. This section outlines the specific Delta features that make the SCD2 Bronze to non-SCD Silver pattern scalable, reliable, and operationally efficient.

Delta Lake provides three essential capabilities that make SCD2 → non-SCD transformations easy and efficient:

2.1 Delta MERGE

Efficient for applying SCD2 updates into the Bronze layer.

2.2 Delta Change Data Feed (CDF)

Provides incremental change streams so Silver tables can process only what changed, not full tables.

2.3 OPTIMIZE and ZORDER

Keep Bronze performant even as it grows into billions of rows.

Together, these features make Databricks ideal for the “historical Bronze → simplified Silver” workflow.

2.4 Why These Features Matter at Scale (Operational Perspective)

While Delta MERGE, Change Data Feed, and OPTIMIZE/ZORDER are often described individually, their real value emerges when used together as part of an end-to-end SCD2-to-current-state pattern.

Delta MERGE allows SCD2 logic to be expressed declaratively and transactionally, removing the need for fragile, multi-step update logic. This is particularly important in regulated environments, where partial updates or failed batch jobs can lead to audit gaps or inconsistent history.

Delta Change Data Feed (CDF) fundamentally changes how downstream layers are built. Instead of reprocessing entire Bronze tables, Silver pipelines can operate incrementally by default—processing only impacted business keys. This dramatically reduces compute costs, improves pipeline reliability, and enables near-real-time propagation of changes without sacrificing correctness.

OPTIMIZE and ZORDER ensure that fully historical SCD2 tables remain performant even as they grow into billions of rows. Without these capabilities, SCD2 designs often degrade over time and become operational liabilities rather than trusted sources of truth.

Together, these features make Databricks uniquely well-suited for architectures that must balance deep historical lineage with fast, intuitive analytical access.

3. The Core Pattern: Building a Current-State Silver Table

At the heart of this architecture is a simple but powerful idea: derive a clean, current-state representation from a fully temporal source of truth. However, achieving this consistently requires more than a basic filter on current records. This section introduces the core conceptual pattern for extracting “what is true now” from SCD2 data while preserving correctness and trust.

To derive a non-SCD Silver table, you must select the **current active version** of each entity from the Bronze SCD2 dataset.

In a typical SCD2 model, Bronze tables contain:

- surrogate key
- natural business key
- attributes
- EffectiveFrom
- EffectiveTo
- IsCurrent flag

3.1 The simplest—and most common—approach

```
SELECT *
FROM bronze_scd2
WHERE IsCurrent = true
```

But this is rarely enough for a production Data Product.

Instead, Silver typically requires a richer transformation.

4. Steps for Building a Non-SCD Silver Layer in Databricks

Turning architectural principles into production-grade pipelines requires a clear, repeatable workflow. From incremental processing to schema simplification and optimisation, each step plays a role in ensuring the Silver layer is performant, accurate, and easy to consume. This section walks through the practical steps commonly used in mature Databricks implementations within UK Financial Services.

Below is the standard workflow used by top Databricks implementations in UK Financial Services.

4.1 Step 1: Incremental Processing with Delta CDF

Instead of reprocessing the entire Bronze table, use CDF to process only new or updated changes:

```
SELECT *
FROM table_changes('bronze_scd2', 'latest')
```

CDF output includes:

- new SCD2 rows
- existing rows that have been made non-current
- the current version for changed business keys

This ensures your Silver transformations are **minimal and efficient**.

4.2 Step 2: Identify the Current Versions

Most Silver tables want just the current state.
There are two main options:

Option A: Use IsCurrent Flag

Fastest, simplest:

```
SELECT *
FROM bronze_scd2
WHERE IsCurrent = true
```

Option B: Use EffectiveTo filtering

This is safer when you don't fully trust upstream IsCurrent logic:

```
SELECT *
FROM bronze_scd2
WHERE EffectiveTo = '9999-12-31'
```

4.3 Step 3: Clean and Simplify the Schema

Silver tables should typically expose:

- business keys
- meaningful attributes
- derived fields
- validated domain structures
- standard naming conventions

Avoid including:

- EffectiveFrom
- EffectiveTo
- SCD2 surrogate keys
- SCD2 flags
- Version numbers

Silver is not for lineage—it is for consumption.

4.4 Step 4: Deduplicate and Normalise

In some frameworks, SCD2 ingestion pipelines may introduce potential duplicates (e.g., same EffectiveFrom timestamp arriving twice).
Silver models should enforce clean uniqueness:

```
ROW_NUMBER() OVER (PARTITION BY CustomerID ORDER BY EffectiveFrom DESC)
```

Deduplication ensures each key appears **once**, which is what consumers expect.

4.5 Step 5: Materialise as a Delta Table

Use `CREATE OR REPLACE TABLE` or Delta Live Tables to make the Silver table concrete.

Benefits:

- Faster downstream queries
- Governed schema
- Consistent data contracts
- Easier lineage tracking in Unity Catalog

4.6 Step 6: Optimise, ZORDER, and Cache (Optional)

Silver layers tend to be queried often and aggressively.

Best practices:

- OPTIMIZE weekly or daily depending on volume
- ZORDER by business key
- Cache frequently used tables

This keeps Silver performant even at scale.

4.7 End-to-End Example: Incremental CDF → Current-State Silver Table

The steps above describe the conceptual workflow, but in practice, these are typically implemented as a single incremental pipeline. The following end-to-end Databricks SQL example shows how Delta Change Data Feed is used to identify impacted business keys, select the latest current-state records from an SCD2 Bronze table, de-duplicate edge cases, and upsert a clean, non-SCD Silver table using a single MERGE operation.

```
-- 0) One-time: ensure CDF is enabled on the Bronze SCD2 table
ALTER TABLE bronze.customer_scd2
SET TBLPROPERTIES (delta.enableChangeDataFeed = true);

-- 1) One-time: create the Silver current-state table (non-SCD schema)
CREATE TABLE IF NOT EXISTS silver.customer_current (
  CustomerID        STRING,
  FullName          STRING,
  Email             STRING,
  CountryCode       STRING,
  RiskRating        STRING,
  UpdatedAt         TIMESTAMP,
  IsDeleted         BOOLEAN
)
USING DELTA;

-- 2) Incremental load: read only changes since a stored checkpoint version
--    (Replace with your orchestration mechanism, e.g. DLT or job parameter)
CREATE OR REPLACE TEMP VIEW cdf_raw AS
SELECT *
FROM table_changes('bronze.customer_scd2', ${last_processed_version});

-- 3) Identify business keys impacted by the CDF window
CREATE OR REPLACE TEMP VIEW changed_keys AS
SELECT DISTINCT CustomerID
FROM cdf_raw
WHERE _change_type IN ('insert','update_postimage','delete');

-- 4) Pull the latest *current-state candidate* per impacted key from Bronze
--    (Safer: prefer IsCurrent, but fall back to EffectiveTo sentinel if needed)
CREATE OR REPLACE TEMP VIEW bronze_current_candidates AS
SELECT
  b.CustomerID,
  b.FullName,
  b.Email,
  b.CountryCode,
  b.RiskRating,
  b.EffectiveFrom,
  b.IsDeleted,
  current_timestamp() AS UpdatedAt
```

```
FROM bronze.customer_scd2 b
JOIN changed_keys k USING (CustomerID)
WHERE (b.IsCurrent = true OR b.EffectiveTo = DATE '9999-12-31');

-- 5) Dedupe: enforce one row per key (handles duplicate EffectiveFrom edge cases)
CREATE OR REPLACE TEMP VIEW bronze_current_deduped AS
SELECT *
FROM (
  SELECT
    *,
    ROW_NUMBER() OVER (
      PARTITION BY CustomerID
      ORDER BY EffectiveFrom DESC, UpdatedAt DESC
    ) AS rn
  FROM bronze_current_candidates
)
WHERE rn = 1;

-- 6) Upsert into Silver: current-state merge (non-SCD)
MERGE INTO silver.customer_current AS s
USING bronze_current_deduped AS u
ON s.CustomerID = u.CustomerID
WHEN MATCHED THEN UPDATE SET
  s.FullName     = u.FullName,
  s.Email        = u.Email,
  s.CountryCode  = u.CountryCode,
  s.RiskRating   = u.RiskRating,
  s.UpdatedAt    = u.UpdatedAt,
  s.IsDeleted    = COALESCE(u.IsDeleted, false)
WHEN NOT MATCHED THEN INSERT (
  CustomerID, FullName, Email, CountryCode, RiskRating, UpdatedAt, IsDeleted
) VALUES (
  u.CustomerID, u.FullName, u.Email, u.CountryCode, u.RiskRating, u.UpdatedAt, COALESCE(u.IsDeleted,
false)
);

-- 7) (Optional) If you hard-delete in Bronze, you can delete from Silver too:
-- DELETE FROM silver.customer_current
-- WHERE CustomerID IN (SELECT CustomerID FROM cdf_raw WHERE _change_type='delete');
```

```
-- 8) One-time / periodic: performance hygiene
-- OPTIMIZE silver.customer_current ZORDER BY (CustomerID);
```

5. Example Architecture Pipeline

Seeing how the layers connect end-to-end helps clarify responsibilities, data contracts, and processing boundaries. A well-designed pipeline makes it obvious where history is preserved, where simplification occurs, and where business value is ultimately delivered. This section presents a reference architecture that shows how Bronze, Silver, and Gold layers work together in practice.

Here's how a well-designed SCD2 → non-SCD flow works in Databricks:

Bronze (SCD2)

- Ingest raw data
- Apply SCD2 logic via MERGE
- Store full history
- Partition + ZORDER
- Use Delta CDF to expose incremental changes

Silver (Non-SCD)

- Use CDF to get only new changes
- Filter for `IsCurrent = true`
- Clean and model data
- Remove SCD2 history columns
- Standardise schema
- Deduplicate
- Materialise as a Delta table

Gold (Business Models / Data Products)

- Join across domains
- Compute KPIs and metrics
- Provide domain views (Customers, Products, Accounts, Transactions)
- Serve BI, ML, dashboards

6. Operational and Resilience Considerations

Designing a Silver current-state layer from SCD2 Bronze data is not just a modelling exercise—it is an operational one. In Financial Services environments, pipelines must be resilient, auditable, and predictable under fail-

ure scenarios.

6.1 Initial Bootstrap vs Incremental Operation

Most Silver tables require an initial full bootstrap, followed by incremental maintenance.

A common approach is:

- First run: build Silver from the full Bronze dataset using `IsCurrent = true` or `EffectiveTo` filtering.
- Subsequent runs: switch to Delta Change Data Feed–driven incremental processing.

This avoids unnecessary complexity during initial load while ensuring long-term efficiency.

6.2 CDF Retention and Backfill Strategy

Delta CDF is subject to retention policies. Pipelines should:

- Track the last processed Delta version explicitly
- Detect gaps in CDF availability
- Fall back to a bounded reprocessing window or full refresh when required

This ensures resilience to operational delays without compromising data correctness.

6.3 Auditability and Failure Recovery

Delta Lake table history provides a built-in audit trail for Silver materialisations:

- Every MERGE is versioned
- Changes are reproducible
- Historical Silver states can be reconstructed if required

This is especially valuable for regulatory review, incident investigation, and controlled backfills.

6.4 Idempotent Silver Pipelines

Silver pipelines should be designed to be idempotent:

- Reprocessing the same CDF window should yield the same result
- MERGE logic should fully overwrite current-state records for affected keys

This simplifies orchestration, retries, and operational recovery.

6.5 Monitoring, Observability, and Data Health

Operational Silver pipelines should expose simple, platform-native monitoring signals rather than bespoke instrumentation.

Databricks provides several built-in capabilities that are particularly effective for SCD2-driven architectures:

- **Delta table history** can be queried to validate when Silver tables were last updated, which operations ran, and whether MERGE activity occurred as expected.
- **Change Data Feed metrics** can be monitored to detect unexpected spikes, drops, or gaps in change volumes, which often indicate upstream ingestion issues.
- **Row-count and key-count checks** on Silver tables help quickly identify failed or partial updates.

These lightweight checks are usually sufficient for operational confidence and align well with Financial Services control frameworks without introducing unnecessary complexity.

6.6 Governance and Unity Catalog Integration

When used with Unity Catalog, the SCD2 Bronze → non-SCD Silver pattern gains additional governance and control benefits.

Unity Catalog enables:

- Centralised access control across Bronze, Silver, and Gold layers
- Clear separation of duties between ingestion, modelling, and consumption
- Column-level security for sensitive attributes
- End-to-end lineage from SCD2 history through to analytical data products

This reinforces the architectural separation between temporal truth and analytical simplicity while ensuring that regulatory, privacy, and data governance requirements are consistently enforced.

7. Design Alternatives and Trade-Offs

While materialised, non-SCD Silver tables are the most common approach in Financial Services, there are valid alternatives depending on workload and governance requirements.

7.1 Views vs Materialised Silver Tables

- **Views** over Bronze SCD2 tables reduce storage but push complexity and compute to query time.
- **Materialised Silver tables** offer predictable performance, stable schemas, and clearer data contracts.

In regulated environments, materialised Silver tables are typically preferred for operational certainty and audit clarity.

7.2 Delta Live Tables (DLT) vs Manual Pipelines

Databricks Delta Live Tables provide a declarative option for building Silver layers, including:

- Built-in dependency management
- Data quality expectations
- Simplified orchestration

However, many Financial Services organisations prefer explicit CDF- and MERGE-based pipelines because they:

- Offer finer-grained operational control
- Align more easily with external schedulers and controls
- Make data movement and logic fully explicit for audit purposes

Both approaches are valid—the choice is primarily organisational rather than technical.

8. Why This Pattern Works So Well in Financial Services

Financial Services organisations face unique pressures: regulatory scrutiny, auditability, and the need for fast insight from complex data estates. The separation of temporal truth and analytical simplicity directly addresses these competing demands. This section explains why the SCD2 Bronze to non-SCD Silver pattern has become a de facto standard across banks, insurers, and regulated institutions.

Financial Services organisations benefit from separating temporal Bronze and simple Silver because:

8.1 Regulatory Needs

Bronze provides:

- forensic trails
- reconstruction for complaints
- historical snapshots
- audit assurance
- complete customer change history

8.2 Operational Efficiency

Silver provides:

- fast analytics
- clean tables
- reduced cognitive load
- stable schemas
- efficient joins

8.3 Data Mesh Alignment

Bronze = source-aligned, lineage-rich
Silver = domain-aligned, consumption-ready

This aligns neatly with the principles of Data Mesh and governed Data Products.

9. Summary: From Temporal Truth to Analytical Simplicity

A strong data architecture should make the right thing easy for every user persona, without compromising governance or trust. By clearly separating historical record-keeping from analytical consumption, organisations can satisfy regulators and analysts alike. This final section summarises the value of the pattern and reinforces why Databricks is particularly well-suited to supporting it at scale.

Moving from an SCD2 Bronze layer to a simple, non-SCD Silver layer is one of the most important architectural moves in a modern Databricks Lakehouse.

It ensures:

- Bronze preserves **what happened**
- Silver exposes **what is true right now**

Databricks is uniquely suited for this transition thanks to Delta Lake features such as MERGE, CDF, and ZORDER, making it possible to maintain rich historical lineage while serving clean, high-performance analytical data.

The pattern is now widely regarded as best practice in Financial Services and beyond.
This entry was posted in article and tagged Bronze Layer, Change Data Feed, Data Engineering, Data Governance, Databricks, Delta Lake, Financial Services Data, Lakehouse Architecture, Medallion Architecture, SCD2, Silver Layer, UK FS SCD2 Bronze on December 12, 2025 [https://horkan.com/2025/12/12/from-scd2-bronze-to-a-non-scd-silver-layer-in-databricks] .

ALSO ON **HORKAN - A BLOG BY WAYNE HORKAN**

I Miss Kartoo: A Nostalgic Look at an ...

a year ago · 2 comments

Kartoo, an innovative search engine launched in 2001, revolutionized the search ...

Scam Alert: My Strange Encounter ...

6 months ago · 3 comments

A detailed account of a suspicious job offer from a fake company called ...

Microsoft's very public "Blue Screen of ...

2 years ago · 13 comments

First reported by RiverCoolCool on his blog: ...

Curios Rabie

a year a

Prompt Teneri explore

0 Comments

1 Login ▼

G

Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

♡ Share

Best Newest Oldest

Be the first to comment.

Subscribe Privacy Do Not Sell My Data