SQL
KONFERENZ
2025



Beyond the Basics:

Advanced SCD2 Implementation for Compound Dimension Tables

Who am I?



Marisol Steinau

- Autodidact Data Enthusiast
- > 8 years experience using Microsoft Data Platform
- Freelance Data Solution Architect
- Passionate about Architecture & DevOps
- LinkedIn: linkedin.com/in/marisol-steinau-bb1253253
- Github: https://github.com/Bieine/Public_Speaking
- Blog: https://www.refugeinaudacity.eu/

Agenda

- Dimensional Modelling
- Slowly Changing Dimension Type 2
- Desired result
- How to get there: problems & solution
- Summary



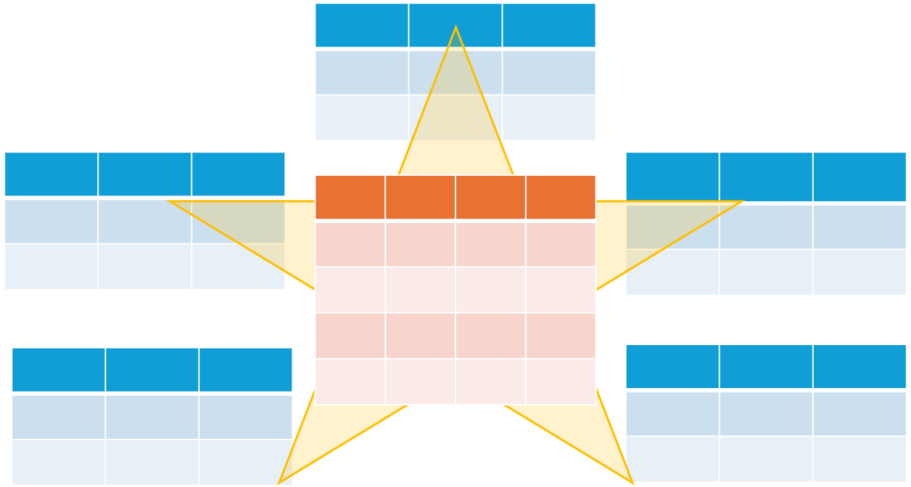
Dimensional Modelling

Dimension Tables

- Provide descriptive context for facts, with attributes for filtering and grouping
- Customers, Products, Regions, Dates

Fact Tables

- Observations or event
- Sales Orders, Inventory, Financial transactions



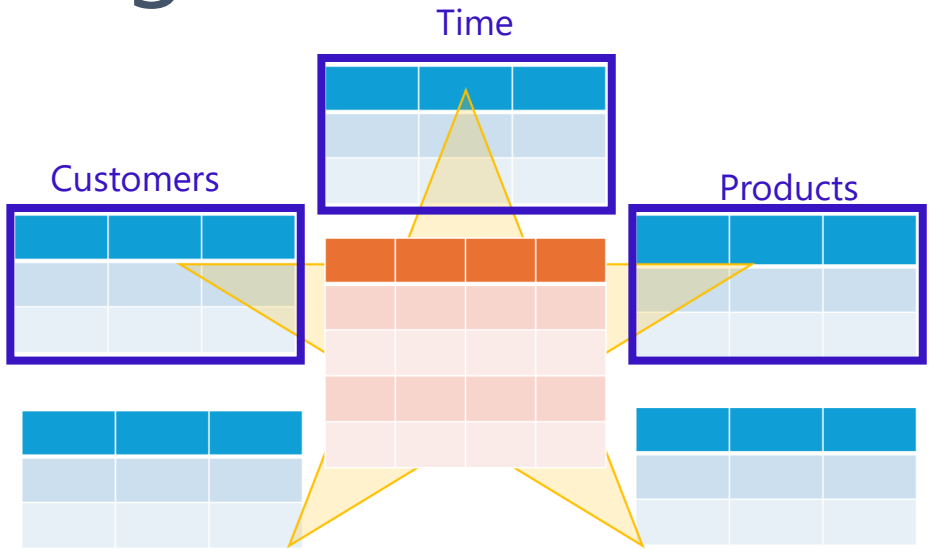
Dimensional Modelling

Dimension Tables

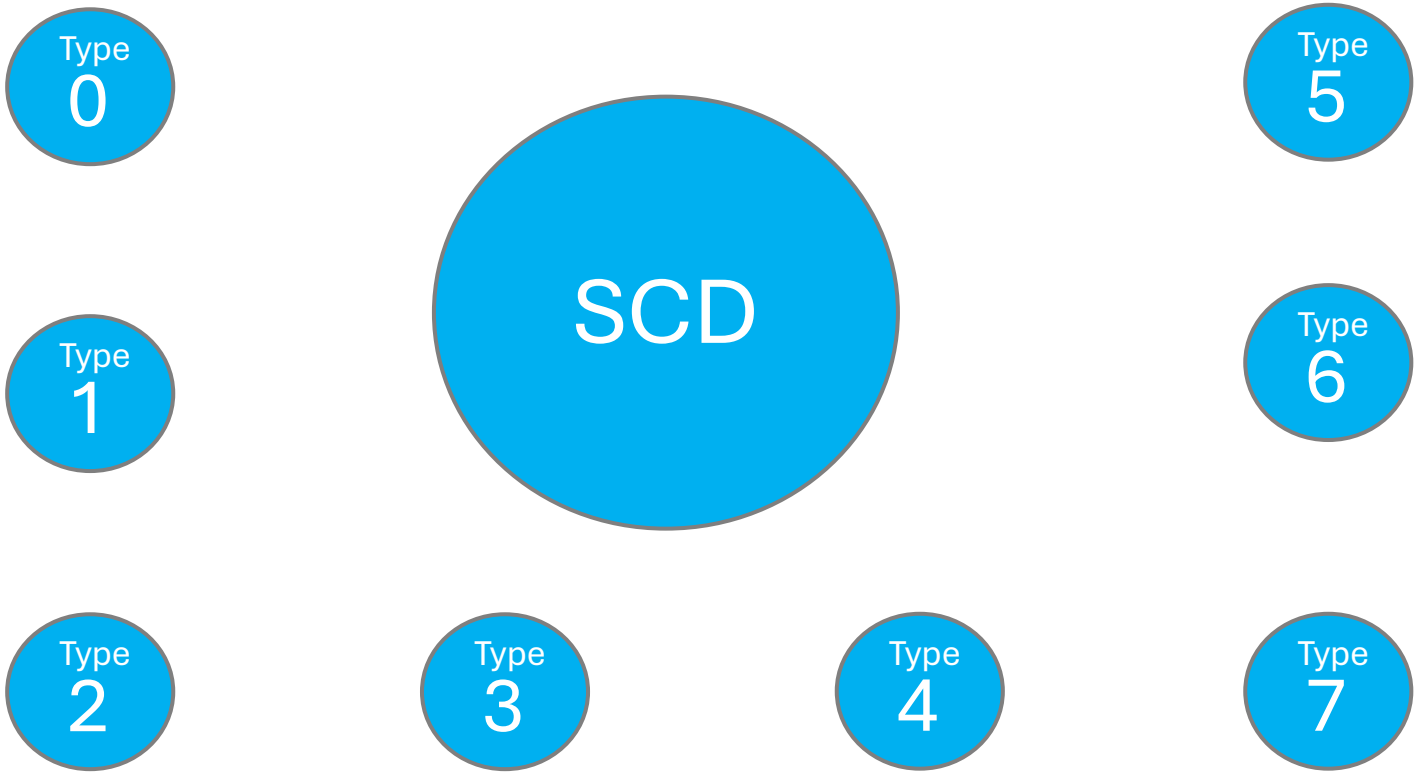
- Provide descriptive context for facts, with attributes for filtering and grouping
- Customers, Products, Regions, Dates

Fact Tables

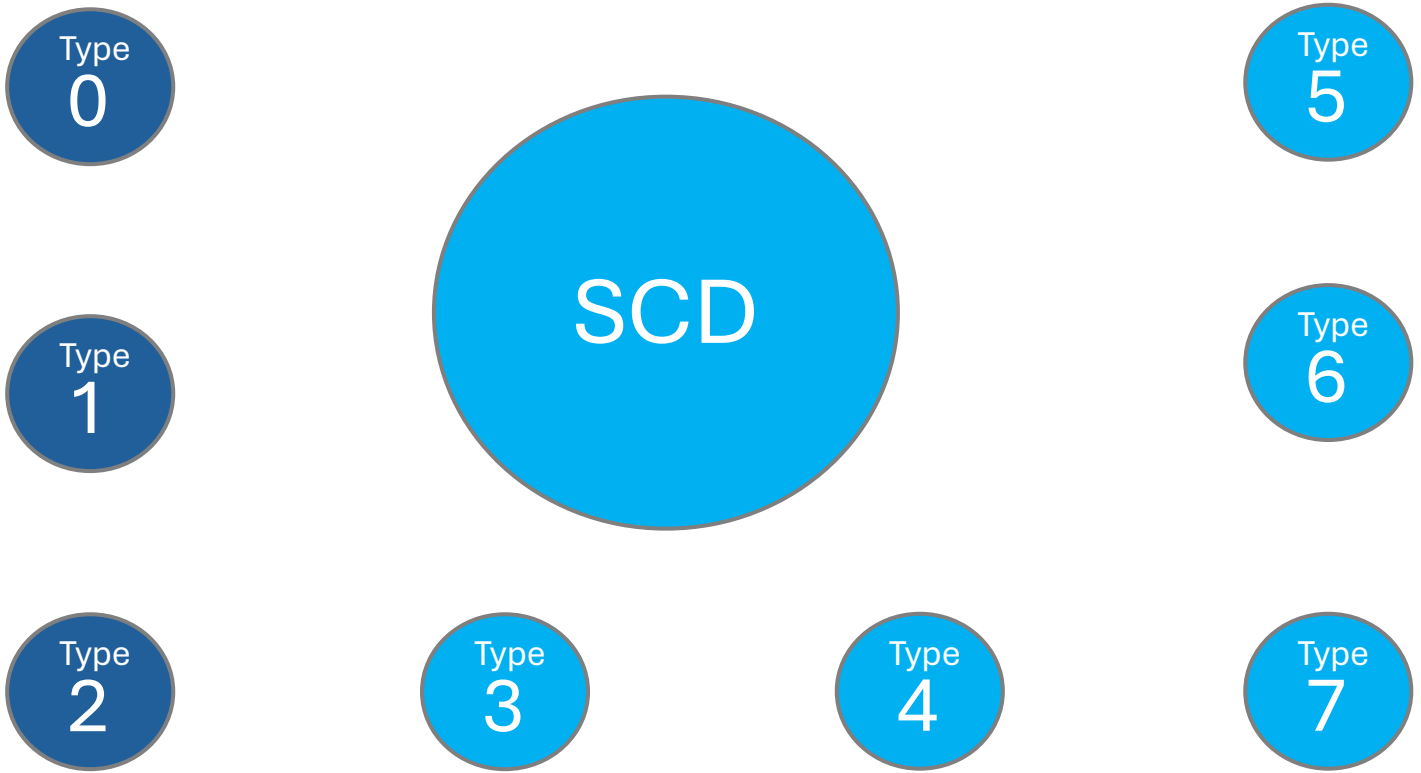
- Observations or event
- Sales Orders, Inventory, Financial transactions



Slowly Changing Dimension Types



Slowly Changing Dimension Types



Types of Keys



A surrogate key is a unique, artificial identifier assigned to each record in a dimension table, typically a sequential number. It has no business meaning and is used solely to uniquely identify each record, helping manage changes in Slowly Changing Dimensions (SCDs) without relying on natural keys.

Types of Keys



An *alternate key* is a unique identifier in a table, often the primary key in the source system, often representing a business attribute like customer ID.

Type 0: Retain Original

- Dimension value attribute never changes
- Type 0 is appropriate for any attribute labeled “original”, e.g., customer original credit score, date of birth, social security numbers
- Applies to most attributes in a date dimension table

SurrogateKey	CustomerID	Name	LastName	DateOfBirth
1	1001	Anna	Schmidt	1990-1-1
2	1002	Lukas	Mueller	1987-2-25

Type 1: Overwrite

- Old attribute value is replaced with current value
- The attribute always reflects the most recent assignment
- No modification of keys in dimension or fact table
- Easy to implement but it does not maintain any history of prior attribute values

Original row in Customer dimension:

SurrogateKey	CustomerID	Name	LastName	SalesTeam
1	1001	Anna	Schmidt	Diamond

Updated row in Customer dimension:

SurrogateKey	CustomerID	Name	LastName	SalesTeam
1	1001	Anna	Schmidt	Ruby

Type 2: Add New Row

- Creates a new record for each change in attribute value
- Maintains full history of attribute values over time
- Uses effective dates and/or current record indicators to track validity
- Fact table remains untouched regarding keys
- Requires additional storage but provides historical insights

Original row in Customer dimension:

SurrogateKey	CustomerID	Name	LastName	SalesTeam	ValidFrom	ValidTo	IsCurrent
1	1001	Anna	Schmidt	Diamond	2023-1-1	9999-12-31	1

Rows in Customer dimension following Sales Team reassignment:

SurrogateKey	CustomerID	Name	LastName	SalesTeam	ValidFrom	ValidTo	IsCurrent
1	1001	Anna	Schmidt	Diamond	2023-1-1	2024-6-15	0
2	1001	Anna	Schmidt	Ruby	2024-6-15	9999-12-31	1

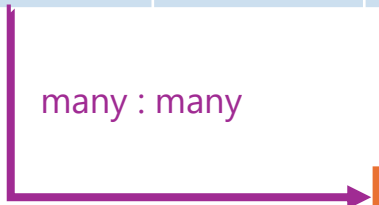
What has happened historically?

- Sales team *Diamond* used to look after customer 103
- Now sales team *Ruby* looks after customer 103
- Customer 103 was first part of the customer group *bronze*
- Now customer 103 is part of the customer group *gold*

SurrogateKey	CustomerID	FirstName	LastName	SalesTeam	CustomerGroup	ValidFrom	ValidTo	IsCurrrent
3	103	Maria	Pereira	Diamond	Bronze	2023-06-10	2023-07-10	0
4	103	Maria	Pereira	Diamond	Silver	2023-07-10	2024-09-01	0
5	103	Maria	Pereira	Ruby	Gold	2024-09-01	9999-12-31	1

How can we create a relationship between dim and fact tables?

SurrogateKey	CustomerID	FirstName	LastName	SalesTeam	CustomerGroup	ValidFrom	ValidTo	IsCurrrent
3	103	Maria	Pereira	Diamond	Bronze	2023-06-10	2023-07-10	0
4	103	Maria	Pereira	Diamond	Silver	2023-07-10	2024-09-01	0
5	103	Maria	Pereira	Ruby	Gold	2024-09-01	9999-12-31	1



CustomerID	Product	SalesDate	Quantity	Price
103	Hand Lotion	2024-11-11	40	12.49
103	Hair Oil	2024-11-11	40	11.99
103	Lipstick Red	2023-07-10	5	12.49
103	Mascara	2023-06-11	1	9.99

Use Surrogate Key in the dimension and add a reference in the fact

SurrogateKey	CustomerID	FirstName	LastName	SalesTeam	CustomerGroup	ValidFrom	ValidTo	IsCurrrent
3	103	Maria	Pereira	Diamond	Bronze	2023-06-10	2023-07-10	0
4	103	Maria	Pereira	Diamond	Silver	2023-07-10	2024-09-01	0
5	103	Maria	Pereira	Ruby	Gold	2024-09-01	9999-12-31	1



CustomerID	FK	Product	SalesDate	Quantity	Price
103	5	Hand Lotion	2024-11-11	40	12.49
103	5	Hair Oil	2024-11-11	40	11.99
103	4	Lipstick Red	2023-07-10	5	12.49
103	3	Mascara	2023-06-11	1	9.99

Add the key into the fact

SurrogateKey	CustomerID	FirstName	LastName	SalesTeam	CustomerGroup	ValidFrom	ValidTo	IsCurrrent
3	103	Maria	Pereira	Diamond	Bronze	2023-06-10	2023-07-10	0
4	103	Maria	Pereira	Diamond	Silver	2023-07-10	2024-09-01	0
5	103	Maria	Pereira	Ruby	Gold	2024-09-01	9999-12-31	1
						>=	<	

Join based on two criteria:

- CustomerID match
- SalesDate is greater or equal than ValidFrom and less than ValidTo

CustomerID	FK	Product	SalesDate	Quantity	Price
103	5	Hand Lotion	2024-11-11	40	12.49
103	5	Hair Oil	2024-11-11	40	11.99
103	4	Lipstick Red	2023-07-10	5	12.49
103	3	Mascara	2023-06-11	1	9.99

What if I want to see the sales for a customer by current sales team?

SurrogateKey	CustomerID	FirstName	LastName	SalesTeam	CustomerGroup	ValidFrom	ValidTo	IsCurrrent	CurrentSales Team
3	103	Maria	Pereira	Diamond	Bronze	2023-06-10	2023-07-10	0	Ruby
4	103	Maria	Pereira	Diamond	Silver	2023-07-10	2024-09-01	0	Ruby
5	103	Maria	Pereira	Ruby	Gold	2024-09-01	9999-12-31	1	Ruby

- Add column to dim with current values
- No need to change relationship between dim and fact

CustomerID	FK	Product	SalesDate	Quantity	Price
103	5	Hand Lotion	2024-11-11	40	12.49
103	5	Hair Oil	2024-11-11	40	11.99
103	4	Lipstick Red	2023-07-10	5	12.49
103	3	Mascara	2023-06-11	1	9.99

SCD in Action



Okay… this all seems easy, right?
But what if your dimension table isn’t coming from just one source? What if it’s actually built from multiple tables?
Now things get interesting!

customer — Showing 4 rows

	IDKunde	Created	Vorname	Nachname	Firma	Anrede	Kundengruppe	Vertriebsteam	Email
1	104	2/20/2025 12:10:40 PM	Selina	Bruno		Frau	bronze	Emerald	selbruno2001@brunoproducts.de
2	100	2/20/2025 12:10:32 PM	Maria	Müller	M&M Cosmetics	Frau	bronze	Azurite	m.mueller03041990@gmail.com
3	101	2/20/2025 12:11:05 PM	Anna	Elchert	Alles für die Frau	Frau	bronze	Emerald	anna.elchert@beautysanctuary.onmicrosoft.com
4	102	2/20/2025 12:10:19 PM	Marius	Schwarz		Herr	bronze	Sapphire	schwarz.marius.panda@outlook.com

customeraddress — Showing 3 rows

	IDKunde	Created	Land	Ort	PLZ	Nummer	Strasse
1	102	2/20/2025 12:09:29 PM	Deutschland	Stuttgart	70173	11	Elsterweg
2	100	2/20/2025 12:09:54 PM	Deutschland	Munich	80331	100	Amselweg..
3	101	2/20/2025 12:04:09 PM	Deutschland	Berlin	10115	8	Unter den Linden

Historizing compound dimension tables– It’s not that simple!

Initial Data (Both Tables Start with One Version)

Customer

CustomerID	Created	LastName	Name	Company	CustomerGroup	SalesTeam	ValidFrom	ValidTo
100	2024-12-01	Schmitt	Alice	Schmitt beauty products	bronze	Diamond	2024-12-01	9999-12-31

Customeraddress

CustomerID	Created	Country	City	Postal Code	Street	Number	ValidFrom	ValidTo
100	2024-12-01	Germany	Berlin	10117	Marienstrasse	14	2024-12-01	9999-12-31

Historizing compound dimension tables– It’s not that simple!

An Update in One Table Only (CustomerGroup Changes)
Alice moves from bronze → silver

Customer

CustomerID	Created	LastName	Name	Company	CustomerGroup	SalesTeam	ValidFrom	ValidTo
100	2024-12-01	Schmitt	Alice	Schmitt beauty products	bronze	Diamond	2024-12-01	2025-01-01
100	2025-01-01	Schmitt	Alice	Schmitt beauty products	silver	Diamond	2025-01-01	9999-12-31

Customeraddress

CustomerID	Created	Country	City	Postal Code	Street	Number	ValidFrom	ValidTo
100	2024-12-01	Germany	Berlin	10117	Marienstrasse	14	2024-12-01	9999-12-31

Historizing compound dimension tables– It’s not that simple!

The approach – wrong Naïve Join on CustomerID

Dim_Customer

CustomerID	LastName	Name	Company	CustomerGroup	SalesTeam	Country	City	PostalCode	Street	Number
100	Schmitt	Alice	Schmitt beauty products	bronze	Diamond	Germany	Berlin	10117	Marienstrasse	14
100	Schmitt	Alice	Schmitt beauty products	silver	Diamond	Germany	Berlin	10117	Marienstrasse	14

✕ **Problem:** This looks fine **only because the address didn’t change**.
But what happens if the address changes later?

Historizing compound dimension tables– It’s not that simple!

The Cartesian Product Issue (Address Update)
→A new address is added on 2025-02-20

Customeraddress

CustomerID	Created	Country	City	Postal Code	Street	Number	ValidFrom	ValidTo
100	2024-12-01	Germany	Berlin	10117	Marienstrasse	14	2024-12-01	2025-02-20
100	2025-02-20	Germany	Munich	80335	Neue Strasse	47	2025-02-20	9999-12-31

The Cartesian Product Issue (Address Update)

If we naïvely join again on CustomerID:

Customer

CustomerID	Created	LastName	Name	Company	CustomerGroup	SalesTeam	ValidFrom	ValidTo
100	2024-12-01	Schmitt	Alice	Schmitt beauty products	bronze	Diamond	2024-12-01	2025-01-01
100	2025-01-01	Schmitt	Alice	Schmitt beauty products	silver	Diamond	2025-01-01	9999-12-31

Customeraddress

CustomerID	Created	Country	City	Postal Code	Street	Number	ValidFrom	ValidTo
100	2024-12-01	Germany	Berlin	10117	Marienstrasse	14	2024-12-01	2025-02-20
100	2025-02-20	Germany	Munich	80335	Neue Strasse	47	2025-02-20	9999-12-31

What's wrong?



Dim_Customer

CustomerID	LastName	Name	Company	CustomerGroup	SalesTeam	Country	City	PostalCode	Street	Number
100	Schmitt	Alice	Schmitt beauty products	bronze	Diamond	Germany	Berlin	10117	Marienstrasse	14
100	Schmitt	Alice	Schmitt beauty products	silver	Diamond	Germany	Berlin	10117	Marienstrasse	14
100	Schmitt	Alice	Schmitt beauty products	bronze	Diamond	Germany	Munich	80335	Neue Strasse	47
100	Schmitt	Alice	Schmitt beauty products	silver	Diamond	Germany	Munich	80335	Neue Strasse	47

The Cartesian Product Issue (Address Update)

If we naïvely join again on CustomerID:

Customer

CustomerID	Created	LastName	Name	Company	CustomerGroup	SalesTeam	ValidFrom	ValidTo
100	2024-12-01	Schmitt	Alice	Schmitt beauty products	bronze	Diamond	2024-12-01	2025-01-01
100	2025-01-01	Schmitt	Alice	Schmitt beauty products	silver	Diamond	2025-01-01	9999-12-31

Customeraddress

CustomerID	Created	Country	City	Postal Code	Street	Number	ValidFrom	ValidTo
100	2024-12-01	Germany	Berlin	10117	Marienstrasse	14	2024-12-01	2025-02-20
100	2025-02-20	Germany	Munich	80335	Neue Strasse	47	2025-02-20	9999-12-31

What's wrong?

Dim_Customer

CustomerID	LastName	Name	Company	CustomerGroup	SalesTeam	Country	City	PostalCode	Street	Number
100	Schmitt	Alice	Schmitt beauty products	bronze	Diamond	Germany	Berlin	10117	Marienstrasse	14
100	Schmitt	Alice	Schmitt beauty products	silver	Diamond	Germany	Berlin	10117	Marienstrasse	14
100	Schmitt	Alice	Schmitt beauty products	bronze	Diamond	Germany	Munich	80335	Neue Strasse	47
100	Schmitt	Alice	Schmitt beauty products	silver	Diamond	Germany	Munich	80335	Neue Strasse	47

This combination did never exist. The customer group bronze is incorrectly linked to the address in Munich

The Cartesian Product Issue (Address Update)

Instead of a naïve join, we need to **also align records by their validity periods**

Customer

CustomerID	Created	LastName	Name	Company	CustomerGroup	SalesTeam	ValidFrom	ValidTo
100	2024-12-01	Schmitt	Alice	Schmitt beauty products	bronze	Diamond	2024-12-01	2025-01-01
100	2025-01-01	Schmitt	Alice	Schmitt beauty products	silver	Diamond	2025-01-01	9999-12-31

Customeraddress

CustomerID	Created	Country	City	Postal Code	Street	Number	ValidFrom	ValidTo
100	2024-12-01	Germany	Berlin	10117	Marienstrasse	14	2024-12-01	2025-02-20
100	2025-02-20	Germany	Munich	80335	Neue Strasse	47	2025-02-20	9999-12-31

Dim_Customer

CustomerID	LastName	Name	Company	CustomerGroup	SalesTeam	Country	City	PostalCode	Street	Number	ValidFrom	ValidTo
100	Schmitt	Alice	Schmitt beauty products	bronze	Diamond	Germany	Berlin	10117	Marienstrasse	14	2024-12-01	2025-01-01
100	Schmitt	Alice	Schmitt beauty products	silver	Diamond	Germany	Berlin	10117	Marienstrasse	14	2025-01-01	2025-02-20
100	Schmitt	Alice	Schmitt beauty products	silver	Diamond	Germany	Munich	80335	Neue Strasse	47	2025-02-20	999-12-31

The Cartesian Product Issue (Address Update)

Instead of a naïve join, we need to **also align records by their validity periods**

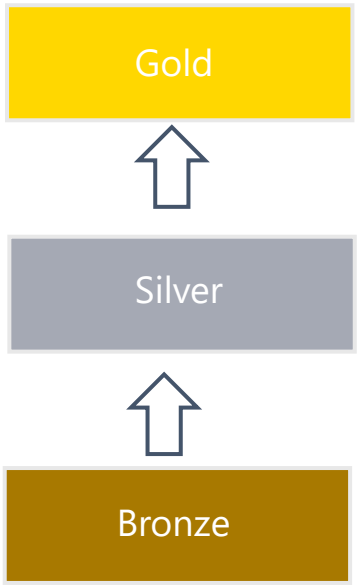
```
SELEC
        c.CustomerID,
        c.LastName,
        c.Name,
        c.Company,
        c.CustomerGroup,
        c.SalesTeam,
        ca.Country,
        ca.City,
        ca.PostcalCode,
        ca.Street,
        ca.Number,
GREATEST(c.ValidFrom, ca.ValidFrom) AS ValidFrom,
LEAST(c.ValidTo, ca.ValidTo) AS ValidTo
FROM customer c
JOIN customeraddress ca
        ON c.CustomerID = ca.CustomerID
        AND c.ValidFrom <= ca.ValidTo
        AND c.ValidTo > ca.ValidFrom
```

Historizing Multiple Source Tables – It’s not That Simple!

- Each source table needs to be historized separately!
- Assign independent ValidFrom–ValidTo timestamps per source table
- Merge records carefully using temporal joins to avoid incorrect history

Architecture





Dimension Tables

- Combines multiple Silver tables into dimensions
- Ensures the correct temporal joins to align different historized tables
- Used for Reporting

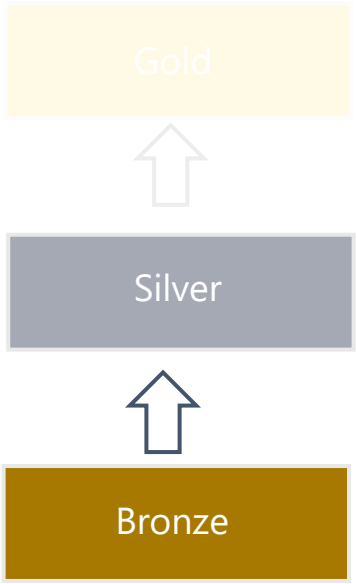
Historized Tables

- Individual tables are cleansed, deduplicated and historized (SCD2)
- Each source table is transformed independently
- Maintains ValidFrom, ValidTo, IsCurrent for tracking changes

Raw Data

- Stores raw data from various sources without transformations
- Could be ingested from databases, APIs, logs, etc.
- Stored as delta tables

Architecture



Data Movement: Bronze → Silver (Full Load)

Bronze Layer (Raw Data)

- Change Data Feed (CDF) is enabled → Tracks all inserts, updates, and deletes
- Reading from the first to the latest version of the source table using CDF
- Filter out "update_preimage" records (removes old values before an update)

Metadata Captured from CDF

- CommitTimestamp → The exact time the change occurred in Bronze
- CommitVersion → A unique incremental version number for every change
- ChangeType → Indicates whether a record was "insert", "update_postimage", or "delete„
- WatermarkCommitVersion → Stores the highest commit version processed (used for incremental loads later)

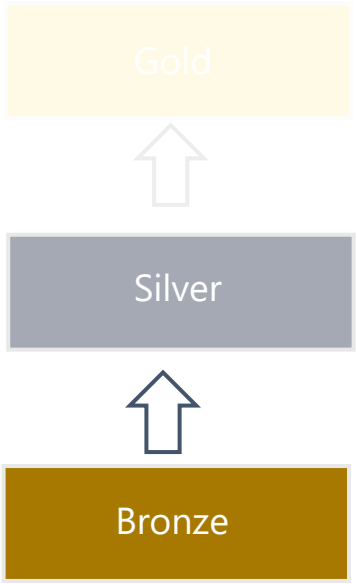
Silver Layer (Historization of Individual Tables)

- Column standardization (renaming from source format to standardized format)
- Hash keys generated for change tracking (based on key business attributes)
- Deduplication to ensure only the first occurrence of each change is stored

Slowly Changing Dimension Type 2 (SCD2)

- ValidFrom → Taken from CommitTimestamp to mark when a record became active
- ValidTo → Next record's CommitTimestamp (or 9999-12-31 for active records)
- IsCurrent → 1 for currently active records, 0 for historical records
- IsDelete → 1 if the record was deleted
- Reordered for schema consistency before insertion into Silver

Architecture



Data Movement: Bronze → Silver (Incremental Load)

Bronze Layer (Raw Data)

- Check for new data → Compare latest commit version in Bronze vs. last processed version in Silver. If no changes, exit.
- Read incremental changes → Use Change Data Feed (CDF) to get only new/updated records
- Filter out "update_preimage" records (removes previous values before an update)

Silver Layer (Historization of Individual Tables)

- Assign the latest CommitVersion as WatermarkCommitVersion (tracks the latest processed version)
- Identify new vs. updated records
  - New → CustomerID not in Silver
  - Updated → Hash has changed for existing CustomerID
- Close out old records → Update ValidTo and IsCurrent for old record
- Insert new and updated records

Slowly Changing Dimension Type 2 (SCD2)

- ValidFrom → Taken from CommitTimestamp to mark when a record became active
- ValidTo → Next record's CommitTimestamp (or 9999-12-31 for active records)
- IsCurrent → 1 for currently active records, 0 for historical records
- IsDelete → 1 if the record was deleted
- Reordered for schema consistency before insertion into Silver

Architecture



Data Movement: Silver → Gold (Full Load)

Silver Layer (Historization of Individual Tables)

- Read individual tables
- Join the tables based on:
  - Primary Key (e.g. CustomerID) – Ensuring that we are merging data for the same customer
  - Validity Period (ValidFrom, ValidTo) – Ensuring that records overlap in their active time ranges
- Keep only relevant columns

Gold Layer (Dimension Tables)

- Appy SCD2 Logic:
  - Set ValidFrom/ValidTo → Take the greatest/minimum from both sources
  - Set IsCurrent = 1 for active records (ValidTo = 9999-12-31)
  - Generate Surrogate Key
- Assign the WatermarkCommitVersion for each source to the combined record (used for incremental loads later)
- Reordered for schema consistency before insertion into Gold

Architecture



Data Movement: Silver → Gold (Incremental Load)

Silver Layer (Historization of Individual Tables)

- Track changes → Retrieve the last processed version (WatermarkCommitVersion) from the target table for both source tables
- Filter new data: Extract only records with changes from both source tables

Gold Layer (Dimension Tables)

- Data processing scenarios:
  - **1) No new records in any source table**

Customer

CustomerID	Created	LastName	Name	Company	CustomerGroup	SalesTeam	ValidFrom	ValidTo

Customeraddress

CustomerID	Created	Country	City	Postal Code	Street	Number	ValidFrom	ValidTo

Do nothing, skip incremental load

Architecture



Data Movement: Silver → Gold (Incremental Load)

Gold Layer (Dimension Tables)

- Data processing scenarios:
  - **2) New records in both source tables**

Customer

CustomerID	Created	LastName	Name	Company	CustomerGroup	SalesTeam	ValidFrom	ValidTo
			New record					

Customeraddress

CustomerID	Created	Country	City	Postal Code	Street	Number	ValidFrom	ValidTo
			New record					

If records are entirely new in silver (e.g. CustomerID does not exist in gold layer)

INSERT

If records are updates of existing records in silver (e.g. CustomerID exists already in gold layer)

UPDATE existing combination
+
INSERT new combination

Architecture



Data Movement: Silver → Gold (Incremental Load)

Gold Layer (Dimension Tables)

- Data processing scenarios:
  - **3) New records only on one source table**

Customer

CustomerID	Created	LastName	Name	Company	CustomerGroup	SalesTeam	ValidFrom	ValidTo
			New record					

Customeraddress

CustomerID	Created	Country	City	Postal Code	Street	Number	ValidFrom	ValidTo
			No new record					

If record is entirely new in silver (e.g. CustomerID does not exist in gold layer)

INSERT
(Fields from other source are NULL)

If record is update of existing records in silver (e.g. CustomerID exists already in gold layer)

UPDATE existing combination
+
INSERT new record from source table 1 combined with last IsCurrent from source table 2

Architecture



Data Movement: Silver → Gold (Incremental Load)

Gold Layer (Dimension Tables)

- Data processing scenarios:
  - **4) New records only on one source table**

Customer

CustomerID	Created	LastName	Name	Company	CustomerGroup	SalesTeam	ValidFrom	ValidTo
			No new record					

Customeraddress

CustomerID	Created	Country	City	Postal Code	Street	Number	ValidFrom	ValidTo
			New record					

If record is entirely new in silver (e.g. CustomerID does not exist in gold layer)

INSERT
(Fields from other source are NULL)

If record is update of existing records in silver (e.g. CustomerID exists already in gold layer)

UPDATE existing combination
+
INSERT new record from source table 2 combined with last IsCurrent from source table 1

SCD in Action



Summary & Tips

- Track changes in bronze and silver layers
- Historize the silver layer
- Track changes in source tables using Watermark Commit Versions
- Handle different scenarios
  - No new data→ Skip processing
  - New data in both→ Insert new records, update existing ones by closing old versions (ValidTo) and setting the new ones as IsCurrent=1
  - New data only in one source→ Update records accordingly while preserving history
- Find the right trigger to start the process, ensuring execution only when necessary (e.g., based on CDF, event-driven triggers or schedule