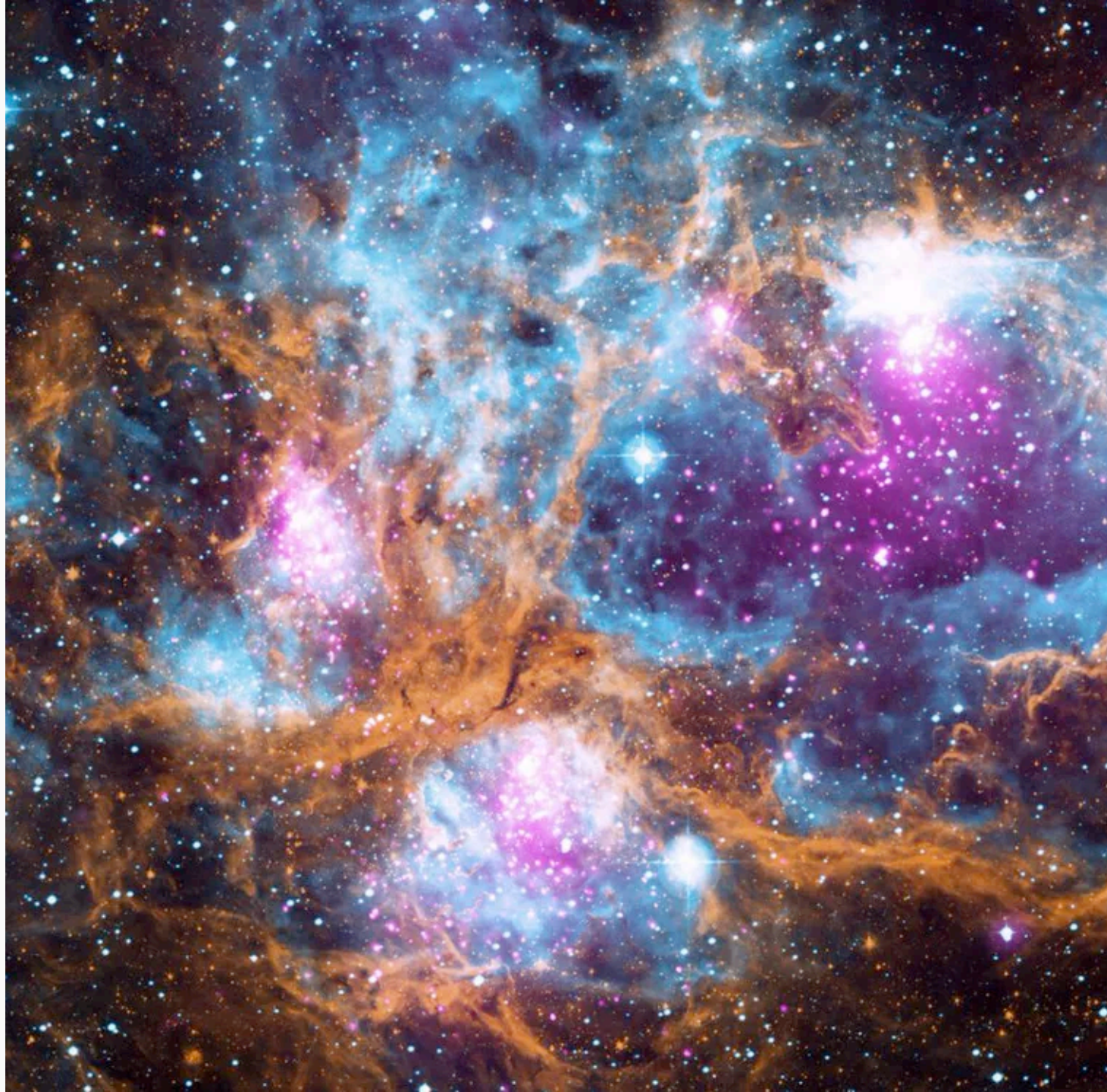
Esoon Ko
IT Högskolan

Slow Changing Dimensions And Data Versioning



Data Versioning

Data versioning refers to the process of keeping track of different versions or states of a dataset over time. This concept is similar to version control in software development but applied to data, allowing you to manage and retrieve past states of your data.

Why Data Versioning?

Traceability: Helps in tracking the history of changes made to the data, making it possible to understand when, how, and why certain changes were made.

Reproducibility: Essential for ensuring that analyses and results can be reproduced accurately, especially in research or regulated industries.

Data Integrity: Allows for the detection of errors or inconsistencies in data by comparing different versions.

Collaboration: Multiple teams can work on the same dataset without overwriting each other's changes, supporting better collaboration.

Auditability: Provides a clear audit trail, which is crucial for compliance with regulations like GDPR, HIPAA, or financial reporting standards.

What can we do when we have Data Versioning? - Examples

Regulatory Compliance: For example, in financial services or healthcare, where laws require organizations to retain and be able to reproduce historical data exactly as it was at a certain point in time.

Data Recovery: If data corruption or loss occurs, data versioning allows for restoring data to a previous, uncorrupted state.

Model Training in Machine Learning: Versioning training datasets ensures that models can be retrained on the exact same data, ensuring consistency in results.

Tracking Data Pipelines: When building data pipelines, versioning helps in tracking how data transforms over time and ensures that any issues can be traced back through the pipeline.

So What methods are there for Data Versioning?

File-based Versioning

- **Manual Versioning:** Saving multiple copies of files with timestamps or version numbers (e.g., data_v1.csv, data_v2.csv).
- **Automated Versioning:** Tools like Git LFS, DVC (Data Version Control), or Quilt that automatically manage versions of data files, even large ones.

Database Versioning

- **Temporal Tables:** Some databases (e.g., SQL Server, Oracle) support temporal tables that automatically track changes to data over time.
- **Change Data Capture (CDC):** Techniques that track and store changes made to a database (e.g., Debezium for streaming CDC).
- **Slow Changing Dimensions (SDC)**: Methods to keep track and store changes of data itself.

Challenges in Data Versioning?

Challenges in Data Versioning?

Storage Overhead: Keeping multiple versions of large datasets can lead to significant storage requirements. Strategies like delta storage or compression can help mitigate this.

Performance Impact: Querying historical versions of data can be slower than querying current data, especially in large datasets.

Complexity in Data Management: Implementing data versioning adds complexity to the data management process, requiring careful planning and tooling.

Consistency Across Systems: Ensuring consistency in data versions across different systems (e.g., databases, data lakes, and filesystems) can be challenging.

Slow Changing Dimensions (SCD)

Slowly Changing Dimensions (SCD) refer to a concept in data warehousing where the attributes of a dimension (e.g., customer, product) change slowly over time. SCDs track historical data and handle changes in **dimension tables**.

SCDs are separated as **types** with each offering different methods for handling changes in dimension data.

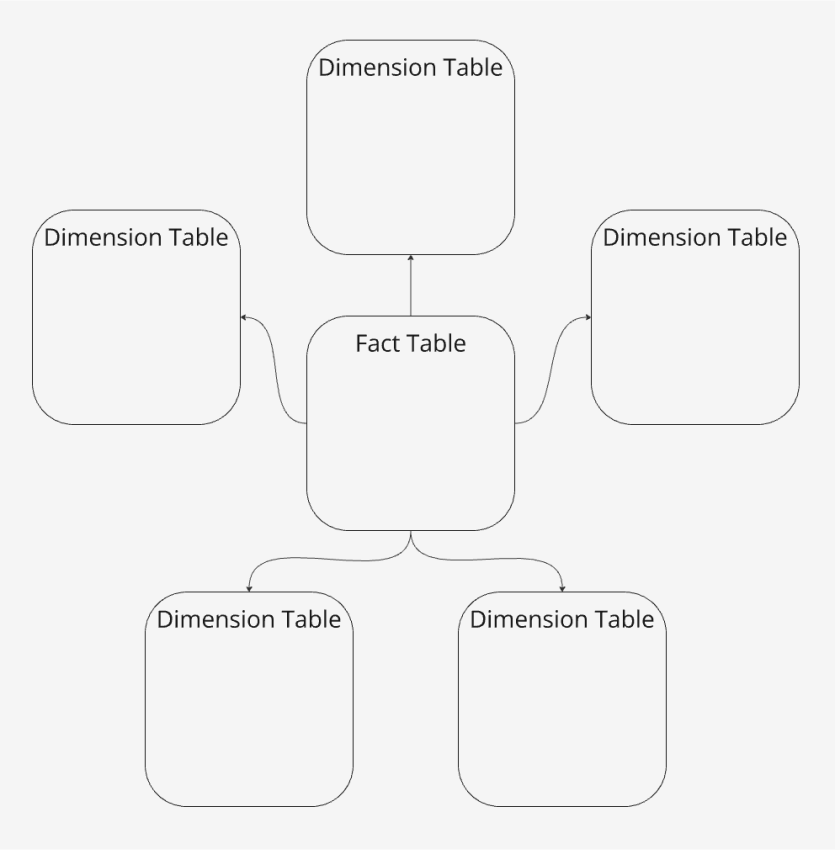
But before we go deeper into SCD types, we must go over what differentiates **Fact tables and Dimension tables.**

Star Schema

Introduced in 1996 by Ralph Kimball, **Star Schema** is a database modeling technique so called due to the diagram of the schema resembling a star shape, with a central "fact table" at the center and several "dimension tables" radiating outwards like the points of a star.

Fact Tables: Store data that captures events, such as a transaction at a retail store, a reservation for a guest at a hotel, or patient visits to a doctor. It contains the quantitative data (measures) that you want to analyze, such as sales revenue, number of units sold, etc.

Dimension Table: Store information that enriches data in fact tables. They contain descriptive attributes (dimensions) related to the data in the fact table. For example, a dimension table might include information about products (like product name, category, and brand), customers (like name, location, and demographic data), or time (like year, quarter, and month).



Slow Changing Dimensions (SCD) Type 0

No changes are allowed after the initial data load. The data remains static.

Slow Changing Dimensions (SCD) Type 1

Overwrites the existing data with new data, with no history of previous data retained.

Slow Changing Dimensions (SCD) Type 1 - Example

Row	item_id	item_name	item_price	item_row
1	1	potato chips	39	12
2	2	cookies	29	10

We want to change the price and placement of potato chips to 35 kr and 11 respectively

Row	item_id	item_name	item_price	item_row
1	1	potato chips	35	11
2	2	cookies	29	10

Slow Changing Dimensions (SCD) Type 2

Adds a new row with the updated data and keeps the old data, usually with an effective date range to indicate the validity of each record.

Slow Changing Dimensions (SCD) Type 2 - Example

Row	item_id	item_name	item_price	item_row	is_current
1	1	potato chips	39	12	true
2	2	cookies	29	10	true

We want to change the price and placement of potato chips to 35 kr and 11 respectively

Row	item_id	item_name	item_price	item_row	is_current
1	1	potato chips	35	11	true
2	1	potato chips	39	12	false
3	2	cookies	29	10	true

Slow Changing Dimensions (SCD) Type 2 - Example

Another method is the use of time

Row	item_id	item_name	item_price	item_row	start_time	end_time
1	1	potato chips	39	12	2024-08-01	2024-09-01
2	2	cookies	29	10	2024-08-01	2024-10-01

We want to change the price and placement of potato chips to 35 kr and 11 respectively

Row	item_id	item_name	item_price	item_row	start_time	end_time
1	1	potato chips	35	11	2024-09-01	2024-10-01
2	1	potato chips	39	12	2024-08-01	2024-09-01
3	2	cookies	29	10	2024-08-01	2024-10-01

Slow Changing Dimensions (SCD) Type 3

Adds a new column to store the new data, keeping a limited history by retaining both old and new data in the same row.

Slow Changing Dimensions (SCD) Type 3 - Example

Row	item_id	item_name	current_item_price	previous_item_price	current_item_row	previous_item_row
1	1	potato chips	39	*null*	12	*null*
2	2	cookies	29	*null*	10	*null*

We want to change the price and placement of potato chips to 35 kr and 11 respectively

Row	item_id	item_name	current_item_price	previous_item_price	current_item_row	previous_item_row
1	1	potato chips	35	39	11	12
2	2	cookies	29	*null*	10	*null*

Slow Changing Dimensions (SCD) Type 4

Stores current data in one table and historical data in a separate table. Is a combination of type 1 and 2.

Slow Changing Dimensions (SCD) Type 4 - Example

Main Table:

Row	item_id	item_name	item_price	item_row
1	1	potato chips	39	12
2	2	cookies	29	10

History Table:

Row	item_id	item_name	item_price	item_row	start_time	end_time
1	1	potato chips	39	12	2024-08-01	2024-09-01
2	2	cookies	29	10	2024-08-01	2024-10-01

We want to change the price and placement of potato chips to 35 kr and 11 respectively

Main Table:

Row	item_id	item_name	item_price	item_row
1	1	potato chips	35	11
2	2	cookies	29	10

History Table:

Row	item_id	item_name	item_price	item_row	start_time	end_time
1	1	potato chips	35	11	2024-09-01	2024-10-01
2	1	potato chips	39	12	2024-08-01	2024-09-01
3	2	cookies	29	10	2024-08-01	2024-10-01

Slow Changing Dimensions (SCD) Type 6

Combines aspects of Types 1, 2, and 3. Typically, it adds a new row (like Type 2) and also includes columns for current and previous values (like Type 3). - Can be thought of as Type 6 (1+2+3)

Example:

Row	item_id	item_name	item_price	item_row	start_time	end_time	is_current
1	1	potato chips	35	11	2024-09-01	2024-10-01	true
2	1	potato chips	39	12	2024-08-01	2024-09-01	false
3	2	cookies	29	10	2024-08-01	2024-10-01	true

Lets guess the dimensions:

Before change:

Row	customer_id	name	address
1	1	John Doe	123 Elm St
2	2	Jane Smith	456 Oak St

After change:

Row	customer_id	name	address
1	1	John Doe	789 Pine St
2	2	Jane Smith	456 Oak St

Lets guess the dimensions:

Before change:

Row	customer_id	name	address
1	1	John Doe	123 Elm St
2	2	Jane Smith	456 Oak St

After change:

Row	customer_id	name	address
1	1	John Doe	789 Pine St
2	2	Jane Smith	456 Oak St

Answer: Type 1

Lets guess the dimensions:

Before change:

Row	customer_id	name	current_address	previous_address
1	1	John Doe	123 Elm St	*null*
2	2	Jane Smith	456 Oak St	*null*

After change:

Row	customer_id	name	current_address	previous_address
1	1	John Doe	789 Pine St	123 Elm St
2	2	Jane Smith	456 Oak St	*null*

Lets guess the dimensions:

Before change:

Row	customer_id	name	current_address	previous_address
1	1	John Doe	123 Elm St	*null*
2	2	Jane Smith	456 Oak St	*null*

After change:

Row	customer_id	name	current_address	previous_address
1	1	John Doe	789 Pine St	123 Elm St
2	2	Jane Smith	456 Oak St	*null*

Answer: Type 3

Lets guess the dimensions:

Before change:

Row	customer_id	name	address	effective_start_date	effective_end_date
1	1	John Doe	123 Elm St	2020-01-01	*null*
2	2	Jane Smith	456 Oak St	2021-05-15	*null*

After change:

Row	customer_id	name	address	effective_start_date	effective_end_date
1	1	John Doe	123 Elm St	2020-01-01	2023-08-01
2	1	John Doe	789 Pine St	2023-08-01	*null*
3	2	Jane Smith	456 Oak St	2021-05-15	*null*

Lets guess the dimensions:

Before change:

Row	customer_id	name	address	effective_start_date	effective_end_date
1	1	John Doe	123 Elm St	2020-01-01	*null*
2	2	Jane Smith	456 Oak St	2021-05-15	*null*

After change:

Row	customer_id	name	address	effective_start_date	effective_end_date
1	1	John Doe	123 Elm St	2020-01-01	2023-08-01
2	1	John Doe	789 Pine St	2023-08-01	*null*
3	2	Jane Smith	456 Oak St	2021-05-15	*null*

Answer: Type 2

Lets guess the dimensions:

Before change:

Row	customer_id	name	current_address	previous_address	effective_start_date	effective_end_date
1	1	John Doe	123 Elm St	*null*	2020-01-01	*null*
2	2	Jane Smith	456 Oak St	*null*	2021-05-15	*null*

After change:

Row	customer_id	name	current_address	previous_address	effective_start_date	effective_end_date
1	1	John Doe	789 Pine St	123 Elm St	2023-08-01	*null*
2	2	Jane Smith	456 Oak St	*null*	2021-05-15	*null*

Lets guess the dimensions:

Before change:

Row	customer_id	name	current_address	previous_address	effective_start_date	effective_end_date
1	1	John Doe	123 Elm St	*null*	2020-01-01	*null*
2	2	Jane Smith	456 Oak St	*null*	2021-05-15	*null*

After change:

Row	customer_id	name	current_address	previous_address	effective_start_date	effective_end_date
1	1	John Doe	789 Pine St	123 Elm St	2023-08-01	*null*
2	2	Jane Smith	456 Oak St	*null*	2021-05-15	*null*

Answer: Type 3

Lets guess the dimensions:

How about this one??

Row	customer_id	name	address	modified_datetime
1	1	John Doe	123 Elm St	2020-01-01
2	1	John Doe	789 Pine St	2023-08-01
3	2	Jane Smith	456 Oak St	2021-05-15

Lets guess the dimensions:

How about this one??

Row	customer_id	name	address	modified_datetime
1	1	John Doe	123 Elm St	2020-01-01
2	1	John Doe	789 Pine St	2023-08-01
3	2	Jane Smith	456 Oak St	2021-05-15

Answer: Type 2

Lets create a table with SCD 1:

Create a table structure with the following:

user_id, user_name, email_address
Add the following two users

{‘user_id’: 1, ‘user_name’: ‘Ali’, ‘email_address’: ‘ali@hotmail.com'}
{‘user_id’: 2, ‘user_name’: ‘Esoon’, ‘email_address’: ‘esoon@gmail.com’}

Ali vill byta email till ‘ali@gmail.com'

You don’t need to actually create the table, just the table schema

Före	user_id	user_name	email_address
	1	Ali	ali@hotmail.com
	2	Esoon	esoon@gmail.com
Efter	user_id	user_name	email_address
	1	Ali	ali@gmail.com
	2	Esoon	esoon@gmail.com

Lets create a table with SCD 2:

Create a table structure with the following:

user_id, user_name, email_address
Add the following two users

{‘user_id’: 1, ‘user_name’: ‘Ali’, ‘email_address’: ‘ali@hotmail.com'}
{‘user_id’: 2, ‘user_name’: ‘Esoon’, ‘email_address’: ‘esoon@gmail.com’}

Ali vill byta email till ‘ali@gmail.com'

You don’t need to actually create the table, just the table schema

Lets create a table with SCD 4:

Create a table structure with the following:

user_id, user_name, email_address
Add the following two users

{‘user_id’: 1, ‘user_name’: ‘Ali’, ‘email_address’: ‘ali@hotmail.com'}
{‘user_id’: 2, ‘user_name’: ‘Esoon’, ‘email_address’: ‘esoon@gmail.com’}

Ali vill byta email till ‘ali@gmail.com'

You don’t need to actually create the table, just the table schema