Whitepaper

How to Manage Slowly Changing Dimensions in Centerprise

Spending too much time modeling your Slowly Changing Dimensions to track historical attributes?

Experience Astera's built-in automated Slowly Changing Dimensions transformation.

sales@astera.com | 888-77-ASTERA

Astera
DATA INTEGRATION MADE EASY

Table of Contents

"The data warehouse must accept the responsibility of accurately describing the past. By doing so, the data warehouse simplifies the responsibilities of the OLTP system. Not only does the data warehouse relieve the OLTP system of almost all forms of reporting, but the data warehouse contains special structures that have several ways of tracking historical data."

-Ralph Kimball

Introduction

Data warehouses, by definition, maintain both current and historical data. Maintaining history requires considerable planning though, because dimensional tables are constrained in terms of change, and data is typically not modified. This presents a problem when the business must change entity descriptions, whether it's a true change like Primary Contact name in Supplier or Customer dimensional entities for instance, or a correction in the data.

Changes like the Primary Contact example used above are rare, happening perhaps once a year or less, and do not have a regular schedule. In database, dimensions that change as slowly are referred to as "Slowly Changing Dimensions (SCD)". The two most common SCD implementations are:

- SCD Type 1: Existing dimensional attribute is overwritten and old data is replaced completely.
- SCD Type 2: New record is created for new data and history is preserved in the original record.

This white paper will cover challenges that businesses face while handling SCD Type 1 and Type 2 in their dimensional data and how Astera addresses these challenges using its specialized SCD transformation.

Why Handling Slowly Changing Dimensions is a Challenge

Ralph Kimball introduced the concept of SCD and has thoroughly expounded on it in his book, but developing the implementation remains complex. The design and development overhead of incorporating SCDs is considerable.

Creating an SCD mapping requires a lot of hand-coding for every action. The number of moving parts involved in mapping SCD is huge; developing scripts for each action is time-consuming and leaves a lot of room for error, especially because the mapping becomes subject to each developer's interpretation of the concept. Efficiency in developing these mappings, and performance achieved are two important considerations.

Centerprise makes the process simpler with a built-in SCD transformation.

How Centerprise Simplifies Implementation of Slowly Changing Dimensions

To minimize the effort involved in the process illustrated above, Astera's Centerprise offers a built-in SCD component that can be dragged into the visual builder and mapped directly to the source.

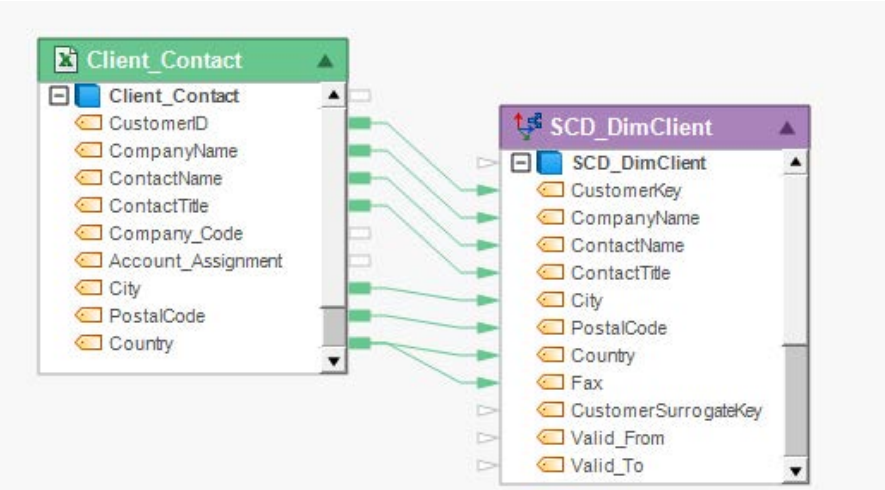


Figure 1 Mapping Your Slowly Changing Dimension

In the above example, we used an operational Excel sheet containing details for a company's client businesses, specifically, details about the company's primary contact. Changes to this sheet are updated in the DimClient dimensional table in our data warehouse[1]. Mappings are created by performing a drag-and-drop action from one field in the source table, to its corresponding field in the SCD transformation, as seen above. The SCD object here contains a connection to our DimClient dimension and posts changes directly to the data warehouse destination.

To set SCD options, we can go to the Properties menu of the transformation and define the attributes for each field according to business requirements. Centerprise will handle the update strategy, performance considerations, routing, and complex joins automatically on the backend, as long as the SCD Field Types in the screen below are defined correctly.

Push-Down Optimization

Centerprise further increases performance when loading and managing large dimensions by using push-down optimization (ELT), leveraging the massive processing power of modern databases and data warehouse appliances. The platform increases efficiency by auto-generating SQL statements necessary for your desired layout in the Slowly Changing Dimension and sending them to the database for processing within the destination's own engine.

[1] Microsoft demo database WideWorldImporters has been used as destination for the SCD object in this example

SCD_DimClient : Layout Fields

	Name	Column Name	SCD Field Type	Active Value	Inactive Value
	CustomerKey	Customer Key	Business Key		
	CompanyName	Company Name	Insert Only		
	ContactName	Contact Name	SCD2 - Update and Insert		
	ContactTitle	ContactTitle	SCD1 - Update		
	CurrentCity	City	SCD6 - Current Value		
	PreviousCity	City History	SCD6 - Previous Value		
	PostalCode	Postal Code	Insert Only		
	CurrentCountry	Country	SCD3 - Current Value		
	OriginalCountry	Original Country	SCD3 - Original Value		
	Phone	Phone	Insert Only		
	CustomerSurroga...	Customer Surrogate Key	Surrogate Key		
	ValidFrom	Valid From	SCD2 Effective Date		
	ValidTo	Valid To	SCD2 Expiration Date		
	Version	Version	Version Number		
	Date_Created	Date Created	Audit - Created		
	Date_Changed	Date Changed	Audit - Last Changed		
	Date_SCD1_Cha...	Date SCD1 Changed	Audit - SCD1 Change		
	Date_SCD2_Cha...	Date SCD2 Changed	Audit - SCD2 Change		
	CurrentRecord	Current Record	Current Record Designator	1	0
	PrevSurrogateKey	PrevSurrogateKey	Previous Surrogate Key		
▶*					

Figure 2 SCD Object Properties menu in Centerprise

Using SCD Field Types

SCD1 – Update

The Type 1 method is used when no historical data is required on how data changed in the SCD field over time. Type 1 method simply updates a record in the dimension table, overwriting an old SCD value with a new value.

In our fictitious scenario, the business is only interested in preserving history for ContactName for their primary contact at each customer business, while associated details for any ContactName require an overwrite because the business is only interested in tracking historical data for their primary contact. SCD1 – Update option is therefore selected for ContactTitle and Fax columns.

Before Update

CustomerKey	Customer Surrogate Key	CompanyName	ContactName	ContactTitle
ANATR	1	The Digital	Aaron Higgs	Sales Representative

After Update

CustomerKey	Customer Surrogate Key	CompanyName	ContactName	ContactTitle
ANATR	1	The Digital	Aaron Higgs	Business Development Manager

We have updated only the ContactTitle here and are not interested in preserving history here. As this column has SCD field type set to 'SCD1 – Update', Centerprise has overwritten the previous value with the new one.

SCD2 – Update and Insert

The Type 2 method makes it possible to track historical data by creating multiple records in the dimension table. The records are identified by using an extra surrogate key in addition to any other keys already present in the table. With Type 2, you have unlimited history as a new record is inserted each time a change to the SCD value is made. Additional fields, such as EffectiveDate, ExpirationDate, or Version may be used to track the timeline of SCD value updates, as well as identify the current SCD value.

We have used SCD2 for the ContactName field in our scenario. Now, when any value in this column is updated in the source table, Centerprise will create a copy of the record and the changes will be reflected in this new copy. The original copy will be Expired.

Before Update

Customer Key	Customer Surrogate Key	CompanyName	ContactName	Valid_From	Valid_To	Current
ANATR	4	The Digital	Aaron Higgs	2018-02-20 08:02:22.201	NULL	1

After Update

Customer Key	Customer Surrogate Key	CompanyName	ContactName	Valid_From	Valid_To	Current
ANATR	4	The Digital	Aaron Higgs	2018-02-20 08:02:22.201	2018-02-22 09:14:36.903	0
ANATR	5	The Digital	Michael Freeman	2018-02-22 09:14:36.903	NULL	1

In the 'After Update' table, Centerprise has automatically created a new record for the same CustomerKey. The CustomerSurrogateKey has been updated sequentially by the system, and the new record contains the updated ContactName. The Effective Date has been updated in Valid_From column for the new record. This is also the date the old record was Expired – visible in the Valid_To column. In the Current column, Active record status can be seen (1 for Active, 0 for Inactive).

Business Key

This designates the field holding the key that is normally used to identify records in the table.

In the Excel source table, the CustomerID column contains unique identifiers for each record, which is the customer code the company uses in their CRM. This is mapped to 'CustomerKey' in our dimensional data and SCD Field Type is set as 'Business Key' for the column.

In the table above, Business Key for both records is the same but Surrogate Key has changed to associate all copies of the same record with the same Business Key while tracking SCD copies with the Surrogate Key.

Insert Only

This is the default field type, and if selected, will insert new values. Updates will be disregarded.

In our fictitious scenario, the business is only interested in updating contact details for each customer, not the rest of the details that the source table contains about the customer. The 'Insert Only' field type is therefore selected for columns that hold details about the customer's business other than contact details. Selecting this field type will ensure that values in all columns specified as 'Insert Only' are inserted only when the flow is run for the first time to load the dimensional table. Any updates in the source table following the initial load will not be reflected in the target data warehouse.

Surrogate Key

This designates the field holding an extra key that identifies versions of the SCD value with the same business key. Surrogate Key is system-generated.

This column does not exist in our source table but has been created (CustomerSurrogateKey) in our dimensional data to track copies of records made by the SCD transformation. If, for instance, ContactName is updated using SCD 2 write strategy, the destination table will contain a copy of the original record. Both versions will have the same Business Key, but different Surrogate Keys.

Previous Surrogate Key

This column stores the previous surrogate key for the same business key.

When several copies of the same record are created via SCD Type 2, users may want to perform operations like Join on latest and previous value. This designated column will hold the Previous Surrogate Key for such operations. We created a column with the same name in our destination dimensional table to track the Previous Surrogate Key.

Before Update

CustomerKey	Customer SurrogateKey	ContactName	PreviousSurrogate Key	Current
ANATR	4	Aaron Higgs	NULL	1

After Update

CustomerKey	Customer SurrogateKey	ContactName	PreviousSurrogate Key	Current
ANATR	4	Aaron Higgs	NULL	0
ANATR	5	Michael Freeman	4	1

Current Record Designator

The Current Record Designator field type is used as a Boolean flag to indicate whether the record is current or not. Users can specify any combination of true or false value, according to their database naming conventions. We usually see 1 for Active and 0 for Inactive in our customers' integration flows. A search for '1' in Current column will fetch the latest records.

In our example, we created a separate column named Current in our dimensional table to keep track of Active records. Now Active records will have value '1' in this column and Inactive ones will be marked as '0'. This can be seen in the tables shown before.

SCD2 Effective Date

The SCD2 Effective Date field will contain the validity date of each record. When the table is populated for the first time, all Effective Date values will be set as NULL but are updated when SCD copy is created. See below if you want to initialize records with current date.

SCD2 Effective Date with Initial Value

By default, 'SCD2 Effective Date' field type will be generated by the system and its value is set as NULL when the table is initialized. If the business wants to use current date as the initial value of SCD2 Effective Date field, the 'SCD2 Effective Date with Initial Value' field type must be used.

For the DimClient table, the ContactName column is marked as SCD2, so when a new record is created, the Effective Date field is updated. We created the Valid_From column to hold Effective Date value. This field contains a value in the original record too, because field type was selected as 'SCD2 Effective Date with Initial Value', as seen in the table under SCD2 – Update and Insert.

SCD2 Expiration Date

Stores the expiration date of the SCD value in the record. Expiration Date is Null in the record storing the current SCD value. If any records are expired, this field will contain the date of expiration. The new record created to handle the change will then have expiration date as NULL, as seen in the table under SCD2 – Update and Insert.

For reporting purposes, the most recent record can be obtained by fetching records with Valid_To = NULL in our scenario.

SCD1 – Update Current Record

This is generally used in cases when the business initially started maintaining history for a particular column, but at some point, decided not to maintain any further history. Here, the column in question would originally be marked as 'SCD2 – Update and Insert', and would later be changed 'SCD1 – Update Current Record'.

This change will result in only the most recent SCD2 values being updated in the column, while the original history will remain as is. If we had changed field type to 'SCD1 – Update' instead, all historical values would have been replaced with the most recent value as well. The business doesn't want historical values to be changed though, and therefore, will use 'SCD1 – Update Current Record'.

'SCD1 – Update Current Record' is also used for performance reasons when many SCD2 versions for a record exist and you want to do an SCD1 update for a column like CompanyName. Using 'SCD1 – Update' for this column would update CompanyName in all SCD2 copies, which could potentially result in hundreds of records being updated. Instead, we can use 'SCD1 – Update Current Record' to change CompanyName in only the most recent SCD2 copy.

Version Number

The Version Number SCD field will automatically increment version for each copy of a record. This makes it easier to keep track of different historical copies of the original record. Version Data Mapping is usually used in conjunction with Effective Date Range Mapping. A search for maximum version number will yield the most recent records.

To track each record copy by version, we have created a new column (Version) in the DimClient table and set SCD field type as 'Version Number'.

Before Update

CustomerKey	CustomerSurrogateKey	ContactName	Version
ANATR	4	Aaron Higgs	1

After Update

CustomerKey	CustomerSurrogateKey	ContactName	Version
ANATR	4	Aaron Higgs	1
ANATR	5	Michael Freeman	2

SCD3 - Current Value and SCD3 - Original Value

The Type 3 method allows users to maintain history in the dimension table, but just of a single record. It doesn't increase the number of rows – the changes are reflected in two columns that are updated as a result.

It creates two versions of the attribute: Current Value and Original Value. The former shows the most recent record, while the latter shows the first-ever record made for that particular attribute. However, the history of changes made between the first and latest record is lost.

Before Update

Customer Key	Customer Surrogate Key	Company Name	Contact Name	Valid From	Country
ANATR	4	The Digital	Aaron Higgs	2018-01-10 07:09:25:100	Germany

After Update

Customer Key	Customer Surrogate Key	Company Name	Contact Name	Valid From	Country	Original Country
ANATR	4	The Digital	Aaron Higgs	2018-02-22 07:09:25:100	Turkey	Germany

Customer Key	Customer Surrogate Key	Company Name	Contact Name	Valid From	Country	Original Country
ANATR	4	The Digital	Aaron Higgs	2018-04-13 12:25:09:067	Malaysia	Germany

In our example scenario, we have applied Type 3 method on the Country field. We have configured our SCD Transformation with Original Country, Country, and Valid From to enable Centerprise to insert values from the SCD3 implementations. When the Country field is updated in the source table, the Country column, which is set to SCD - Current Value, will show the new value, indicating that this is the most recent value in the source table.

The Original Country column is set to field type SCD3 - Original Value. When the record is updated in the source table, it will show the first value that has been inserted initially. The Valid From column will show the date the source table was updated. In case the Country value is changed from Turkey to Malaysia, the record for Turkey will be lost, as the Type 3 method allows to keep the history of the initial and latest values only.

SCD6 - Current Value and SCD6 - Previous Value

The Type 6 method, also known as the SCD Hybrid approach, is a combination of Type 1, Type 2, and Type 3. It is applicable when a business wants to keep track of historical changes while being able to analyze historical performance data based on the current values of specific attributes.

Before Update

Customer Key	Customer Surrogate Key	Company Name	Contact Name	Valid From	Valid To	City
ANATR	4	The Digital	Aaron Higgs	2018-02-22 07:09:25:100	9999-12-31 23:59:59:000	Chicago

After Update

Customer Key	Customer Surrogate Key	Company Name	Contact Name	Valid From	Valid To	City	City History	Current Record
ANATR	4	The Digital	Aaron Higgs	2018-02-22 07:09:25:100	2018-03-02 03:25:10:252	Seattle	Chicago	0
ANATR	5	The Digital	Aaron Higgs	2018-03-02 03:26:25:065	9999-12-31 23:59:59:000	Seattle	Seattle	1

Customer Key	Customer Surrogate Key	Company Name	Contact Name	Valid From	Valid To	City	City History	Current Record
ANATR	4	The Digital	Aaron Higgs	2018-02-22 07:09:25:100	2018-03-02 03:25:10:252	California	Chicago	0
ANATR	5	The Digital	Aaron Higgs	2018-03-02 03:26:25:065	2018-06-10:45:15:125	California	Seattle	0
ANATR	6	The Digital	Aaron Higgs	2018-02-22 10:46:50:025	9999-12-31 23:59:59:000	California	California	1

Each of the fundamental SCD types performs a specific function in the Type 6 method:

- Type 1: It overwrites new values on the Type 3 attribute to provide the latest snapshot for a record.
- Type 2: Rows are added to track historical changes as new records are inserted into the source table.
- Type 3: It updates the data warehouse table with the 'Current' attribute.

We have applied the Type 6 method on the City field in our scenario. When the record is updated in the source table, a new row is added, reflecting the updated information in several fields of the data warehouse table, bearing a unique surrogate key (SCD Type 2). The City column is updated to show the latest value, while the City History column shows the previous value (Type 1 and Type 3). Valid From and Valid To columns are updated to show the timeline of each record. The Current Record column is also updated, showing the active record status on the latest entry.

Similarly, when the source table is updated with another record, the data warehouse table is again updated, changing the values in the columns in the same manner as above. However, notice that the City column shows the same value for all three records. This is because it follows the SCD Type 1 properties, overwriting the previous values with the latest ones.

Audit - Created

Stores the date and time when a new record with a new SCD value was created. The value is stored in DateCreated column in our scenario.

Before Update

Customer Key	Customer Surrogate Key	CompanyName	ContactName	DateCreated	DateChanged	Current
ANATR	4	The Digital	Aaron Higgs	2018-02-20 08:02:22.201	NULL	1

After Update

Customer Key	Customer Surrogate Key	CompanyName	ContactName	DateCreated	DateChanged	Current
ANATR	4	The Digital	Aaron Higgs	2018-02-20 08:02:22.201	2018-02-22 09:14:36.903	0
ANATR	5	The Digital	Michael Freeman	2018-02-22 09:14:36.910	NULL	1

Audit - Last Changed

Stores the date and time when the record with an SCD value was last updated, as seen in the table above. The value is stored in DateChanged column in our scenario.

Audit - SCD1 Change

Stores the date and time when the record with the SCD Type 1 value was last updated. The value is stored in Date SCD1 Change column in our scenario.

Audit - SCD2 Change

Stores the date and time when the record with an SCD Type 2 value was last updated. The value is stored in Date SCD2 Change column in our scenario.

Configuring the SCD Layout

While the 'SCD Field Type' is the most important aspect of the Centerprise Slowly Changing Dimension component, there are a few other fields that need to be considered when building the layout:

DB Type

This field identifies the data type defined originally in your database table. Centerprise will automatically load these up when connection is established initially.

We recommend keeping storage and performance optimization in mind when choosing the data type for each column. For instance, for a column like CompanyName, set data type as VARCHAR (20) instead of CHAR (20). This way, the SQL Server will use only take up the space that the input requires; the full 20-character length will not be pre-allocated. But if we use CHAR (20), the database will take up the full 20 bytes, regardless of the number of characters that will actually be stored in each field of the column. These are small considerations that can result in improved database performance.

Remember, the data types must be defined within the database rather than Centerprise.

Recommendations:

Storage: Always use a data type that fulfills the requirements of your expected data in each column. Don't exceed it. See the CHAR (20) and VARCHAR (20) example above.

Data Type: For business keys and surrogate keys, integer data types can be used in most cases, rather than string. Besides optimizing for storage, this can help avoid possible leading and trailing spaces, and encoding compatibility issues.

Configuring the SCD Layout

While the 'SCD Field Type' is the most important aspect of the Centerprise Slowly Changing Dimension component, there are a few other fields that need to be considered when building the layout:

DB Type

This field identifies the data type defined originally in your database table. Centerprise will automatically load these up when connection is established initially.

We recommend keeping storage and performance optimization in mind when choosing the data type for each column. For instance, for a column like CompanyName, set data type as VARCHAR (20) instead of CHAR (20). This way, the SQL Server will use only take up the space that the input requires; the full 20-character length will not be pre-allocated. But if we use CHAR (20), the database will take up the full 20 bytes, regardless of the number of characters that will actually be stored in each field of the column. These are small considerations that can result in improved database performance.

Remember, the data types must be defined within the database rather than Centerprise.

Recommendations:

Storage: Always use a data type that fulfills the requirements of your expected data in each column. Don't exceed it. See the CHAR (20) and VARCHAR (20) example above.

Data Type: For business keys and surrogate keys, integer data types can be used in most cases, rather than string. Besides optimizing for storage, this can help avoid possible leading and trailing spaces, and encoding compatibility issues.

Data Type

Centerprise simplifies data type annotations within its interface. For instance, all integer types from your database (TINYINT, SMALLINT, INT, and BIGINT) are simply referred to as INTEGER within Centerprise. This layout field is filled out automatically in your SCD object.

Nullable

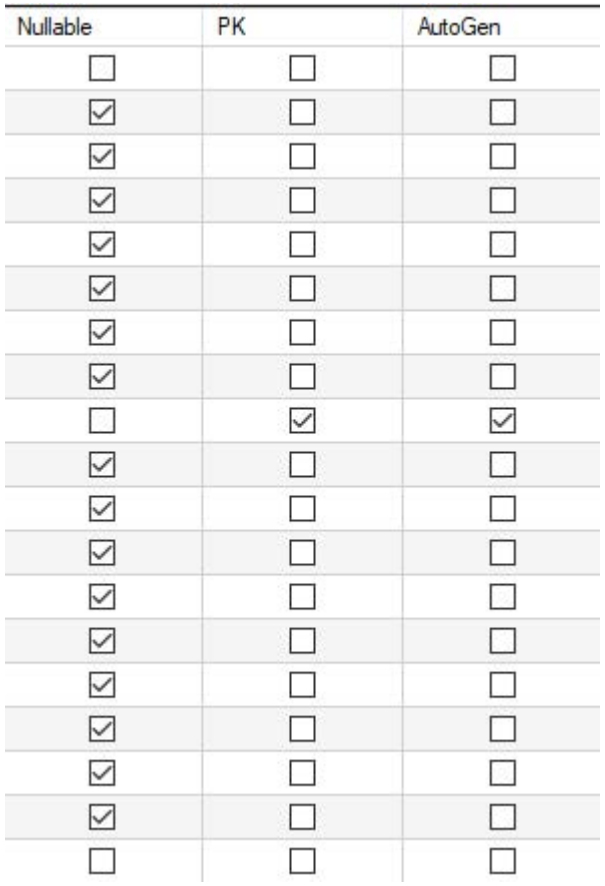
Set which column values you want to be able to set as NULL. You cannot set your Primary Key and your Business Key column as Nullable.

PK

You can specify your Primary Key column here. For SCD, we usually set the Surrogate Key as PK.

AutoGen

Use this property to auto generate values for the specified column using a sequence generator. This is especially useful for your Surrogate Key.



Conclusion

Slowly Changing Dimensions are an important part of every data warehouse. While the challenges are many, the key is to optimize the design of your data warehouse with a clear understanding of expected changes in data and the kind of history that the organization wants to preserve. Centerprise manages complex implementations like SCD for users without the need for hand-coding, freeing you up to focus on the design of your data warehouse.

Designed as an agile data integration platform for both technical and non-technical users, Centerprise is a complete Extract-Transform-Load (ETL) platform that is scalable enough to meet even the most demanding data warehouse needs. The built-in transformations allow users to visually apply complex operations on extracted data to transform it into the finished product, ready to be loaded to your data warehouse.

www.astera.com

Contact us for more information or to request a free trial
sales@astera.com | 888-77-ASTERA