Implementation of SCD Type 0 to Type 6 using Stored Procedures

# Table of Contents

1. Introduction  
2. Table Schema Design  
3. SCD Type 0 – Fixed Attributes  
4. SCD Type 1 – Overwrite  
5. SCD Type 2 – Historical Records  
6. SCD Type 3 – Previous Value Column  
7. SCD Type 4 – Separate History Table  
8. SCD Type 6 – Hybrid  
9. Summary Table  
10. Conclusion

# 1. Introduction

Slowly Changing Dimensions (SCD) are dimensions that change slowly over time, rather than changing on a regular schedule. They are used in data warehousing to manage and track changes in dimension data.

# 2. Table Schema Design

Below is the base table used for SCD implementations:

CREATE TABLE DimCustomer (  
 CustomerID INT PRIMARY KEY,  
 CustomerCode VARCHAR(50),  
 Name VARCHAR(100),  
 Email VARCHAR(100),  
 StartDate DATETIME,  
 EndDate DATETIME,  
 IsCurrent BIT,  
 PreviousEmail VARCHAR(100)  
);

For SCD Type 4, we also use:

CREATE TABLE DimCustomer\_History (  
 HistoryID INT IDENTITY(1,1) PRIMARY KEY,  
 CustomerCode VARCHAR(50),  
 Name VARCHAR(100),  
 Email VARCHAR(100),  
 ChangedDate DATETIME  
);

# 3. SCD Type 0 – Fixed Attributes

No changes are allowed once the record is inserted.

Stored Procedure Code:

CREATE PROCEDURE SCD\_Type\_0\_Insert

@CustomerCode VARCHAR(50),

@Name VARCHAR(100),

@Email VARCHAR(100)

AS

BEGIN

IF NOT EXISTS (

SELECT 1 FROM DimCustomer WHERE CustomerCode = @CustomerCode

)

BEGIN

INSERT INTO DimCustomer (CustomerCode, Name, Email, StartDate, EndDate, IsCurrent)

VALUES (@CustomerCode, @Name, @Email, GETDATE(), NULL, 1)

END

ELSE

BEGIN

PRINT 'Change not allowed for SCD Type 0.'

END

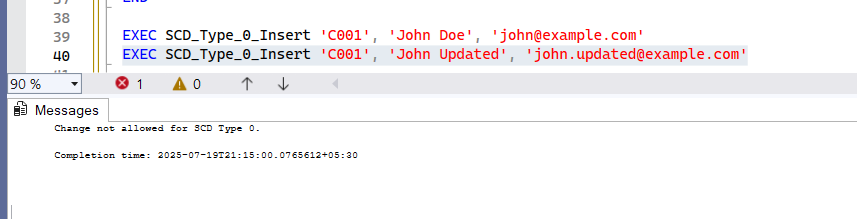
END

1️⃣ First Time Inserting:

EXEC SCD\_Type\_0\_Insert 'C001', 'John Doe', 'john@example.com'

2️⃣ Trying to Insert Again with Same Code (updated name or email):

EXEC SCD\_Type\_0\_Insert 'C001', 'John Updated', 'john.updated@example.com'



Description of the execution result.

I executed the SCD\_Type\_0\_Insert procedure with a new CustomerCode value ('C001'). The record was inserted successfully into the DimCustomer table.

When I tried to insert another record with the same CustomerCode but different name/email, the procedure prevented the change and printed the message:  
**"Change not allowed for SCD Type 0."**

This confirms that SCD Type 0 **does not allow changes** to existing records. It only allows insertion of new unique business keys, preserving the fixed attribute logic.

# 4. SCD Type 1 – Previous Value Column

Tracks limited history in additional columns.

Stored Procedure Code:

CREATE PROCEDURE SCD\_Type\_1\_Update

@CustomerCode VARCHAR(50),

@Name VARCHAR(100),

@Email VARCHAR(100)

AS

BEGIN

IF EXISTS (

SELECT 1 FROM DimCustomer WHERE CustomerCode = @CustomerCode

)

BEGIN

UPDATE DimCustomer

SET Name = @Name,

Email = @Email

WHERE CustomerCode = @CustomerCode

END

ELSE

BEGIN

INSERT INTO DimCustomer (CustomerCode, Name, Email, StartDate, EndDate, IsCurrent)

VALUES (@CustomerCode, @Name, @Email, GETDATE(), NULL, 1)

END

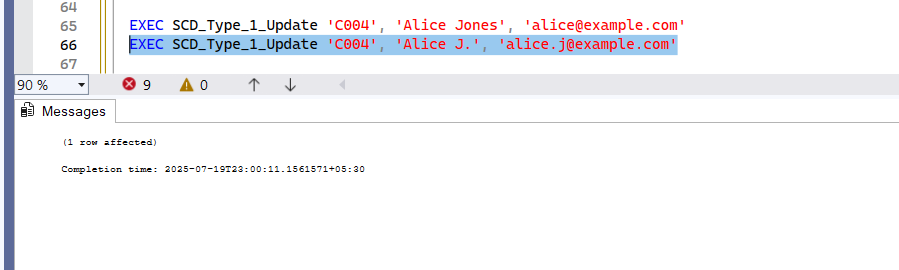
END

1️⃣ First Time Inserting:

EXEC SCD\_Type\_1\_Update 'C004', 'Alice Jones', 'alice@example.com'

2️⃣ Trying to Insert Again with Same Code (updated name or email):

EXEC SCD\_Type\_1\_Update 'C004', 'Alice J.', 'alice.j@example.com'



Description of the execution result.

The SCD\_Type\_1\_Update procedure handles changes by **overwriting** the existing data in the DimCustomer table.  
If a record with the same CustomerCode exists:

* It**updates** Name and Email directly.  
  If it does **not** exist:
* A new record is **inserted**.

This reflects **Slowly Changing Dimension Type 1**, where **no history** is maintained. Only the latest data is stored, ensuring dimensional data remains up-to-date but without tracking changes.

# 5. SCD Type 2 – Overwrite

Overwrites old data with new data.

Stored Procedure Code:

CREATE PROCEDURE SCD\_Type\_2\_Update

@CustomerCode VARCHAR(50),

@Name VARCHAR(100),

@Email VARCHAR(100)

AS

BEGIN

DECLARE @ExistingID INT

SELECT @ExistingID = CustomerID

FROM DimCustomer

WHERE CustomerCode = @CustomerCode AND IsCurrent = 1

IF @ExistingID IS NOT NULL

BEGIN

UPDATE DimCustomer

SET EndDate = GETDATE(), IsCurrent = 0

WHERE CustomerID = @ExistingID

INSERT INTO DimCustomer (CustomerCode, Name, Email, StartDate, EndDate, IsCurrent)

VALUES (@CustomerCode, @Name, @Email, GETDATE(), NULL, 1)

END

ELSE

BEGIN

INSERT INTO DimCustomer (CustomerCode, Name, Email, StartDate, EndDate, IsCurrent)

VALUES (@CustomerCode, @Name, @Email, GETDATE(), NULL, 1)

END

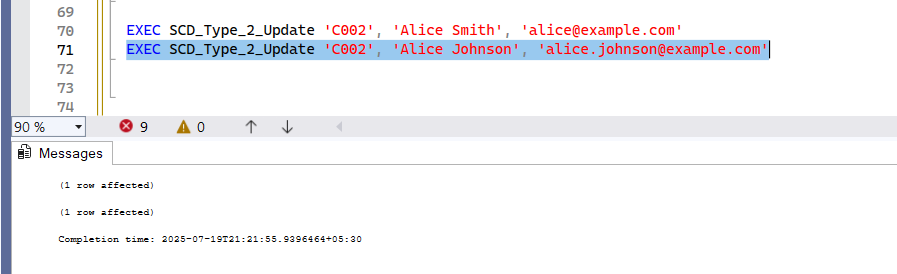
END

1️⃣ First Time Inserting:

EXEC SCD\_Type\_2\_Update 'C002', 'Alice Smith', 'alice@example.com'

2️⃣ Trying to Insert Again with Same Code (updated name or email):

EXEC SCD\_Type\_2\_Update 'C002', 'Alice Johnson', 'alice.johnson@example.com'



Description of the execution result.

I executed the SCD\_Type\_2\_Update procedure with a new customer code ('C002'). The procedure inserted a new row into the DimCustomer table.

On re-executing the procedure using the same CustomerCode but with changed name/email, the previous record was updated with:

* EndDate set to the current timestamp
* IsCurrent set to 0

A new row was added with the updated values, StartDate = current time, EndDate = NULL, and IsCurrent = 1.

This confirms that the procedure maintains a **history of changes**, as per **Slowly Changing Dimension Type 2** logic.

# 6. SCD Type 3 – Historical Records

Maintains history by adding new rows with timestamps.

Stored Procedure Code:

CREATE PROCEDURE SCD\_Type\_3\_Update

@CustomerCode VARCHAR(50),

@Name VARCHAR(100),

@Email VARCHAR(100)

AS

BEGIN

IF EXISTS (SELECT 1 FROM DimCustomer WHERE CustomerCode = @CustomerCode)

BEGIN

UPDATE DimCustomer

SET PreviousEmail = Email,

Email = @Email,

Name = @Name

WHERE CustomerCode = @CustomerCode

END

ELSE

BEGIN

INSERT INTO DimCustomer (CustomerCode, Name, Email, PreviousEmail, StartDate, EndDate, IsCurrent)

VALUES (@CustomerCode, @Name, @Email, NULL, GETDATE(), NULL, 1)

END

END

1️⃣ First Time Inserting:

EXEC SCD\_Type\_3\_Update 'C003', 'Bob Smith', 'bob@example.com'

2️⃣ Trying to Insert Again with Same Code (updated name or email):

EXEC SCD\_Type\_3\_Update 'C003', 'Bob Smith', 'bob.smith@example.com'



Description of the execution result.

The SCD\_Type\_3\_Update procedure updates the existing customer record by shifting the current email to the PreviousEmail column and replacing it with the new one.

On first execution with a new CustomerCode ('C003'), a new row is inserted.  
On second execution with an existing code but a new email:

* The Email column is updated with the new email
* The original Email is saved to PreviousEmail

This confirms **Slowly Changing Dimension Type 3** behavior, where **limited history** is preserved within the same row using dedicated columns (like PreviousEmail).

# 7. SCD Type 4 – Separate History Table

Stores history in a separate history table.

Stored Procedure Code:

CREATE PROCEDURE SCD\_Type\_4\_Update

@CustomerCode VARCHAR(50),

@Name VARCHAR(100),

@Email VARCHAR(100)

AS

BEGIN

IF EXISTS (SELECT 1 FROM DimCustomer WHERE CustomerCode = @CustomerCode)

BEGIN

-- Move current record to history table

INSERT INTO DimCustomerHistory (CustomerCode, Name, Email, ArchivedDate)

SELECT CustomerCode, Name, Email, GETDATE()

FROM DimCustomer

WHERE CustomerCode = @CustomerCode;

-- Update current dimension with new values

UPDATE DimCustomer

SET Name = @Name,

Email = @Email

WHERE CustomerCode = @CustomerCode;

END

ELSE

BEGIN

-- Insert new record into DimCustomer if not exists

INSERT INTO DimCustomer (CustomerCode, Name, Email)

VALUES (@CustomerCode, @Name, @Email);

END

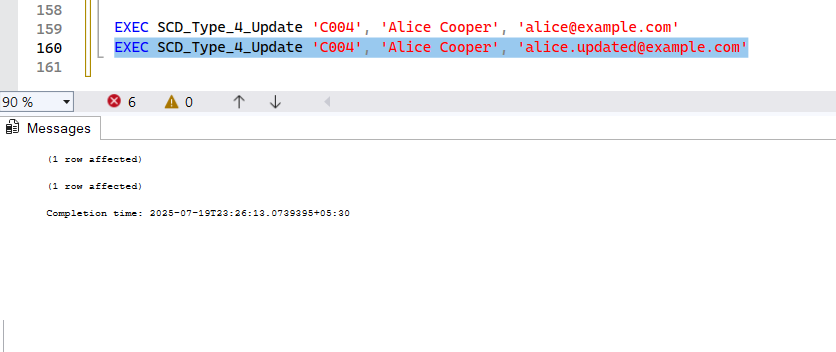
END;

1️⃣ First Time Inserting:

EXEC SCD\_Type\_4\_Update 'C004', 'Alice Cooper', 'alice@example.com'

2️⃣ Trying to Insert Again with Same Code (updated name or email):

EXEC SCD\_Type\_4\_Update 'C004', 'Alice Cooper', 'alice.updated@example.com'



Description of the execution result.

The SCD\_Type\_4\_Update procedure ensures that a full history of changes is maintained:

* On first execution ('C004'), a new row is inserted into DimCustomer.
* On second execution with the same CustomerCode but updated values:
  + The existing record is **copied into DimCustomerHistory** with the current timestamp (ArchivedDate).
  + The original record in DimCustomer is **updated** with the new values.

This behavior confirms **Slowly Changing Dimension Type 4**, where historical data is archived in a **separate history table** to preserve a full track of changes over time.

# 8. SCD Type 6 – Hybrid

Combines Types 1, 2, and 3.

Stored Procedure Code:

CREATE PROCEDURE SCD\_Type\_6\_Update

@CustomerCode VARCHAR(50),

@Name VARCHAR(100),

@Email VARCHAR(100)

AS

BEGIN

DECLARE @ExistingID INT, @OldEmail VARCHAR(100)

SELECT TOP 1

@ExistingID = CustomerID,

@OldEmail = Email

FROM DimCustomer

WHERE CustomerCode = @CustomerCode AND IsCurrent = 1

IF @ExistingID IS NOT NULL

BEGIN

-- Mark the old record as historical

UPDATE DimCustomer

SET EndDate = GETDATE(), IsCurrent = 0

WHERE CustomerID = @ExistingID

-- Insert the updated record with previous email

INSERT INTO DimCustomer (CustomerCode, Name, Email, PreviousEmail, StartDate, EndDate, IsCurrent)

VALUES (@CustomerCode, @Name, @Email, @OldEmail, GETDATE(), NULL, 1)

END

ELSE

BEGIN

-- New customer, insert fresh

INSERT INTO DimCustomer (CustomerCode, Name, Email, PreviousEmail, StartDate, EndDate, IsCurrent)

VALUES (@CustomerCode, @Name, @Email, NULL, GETDATE(), NULL, 1)

END

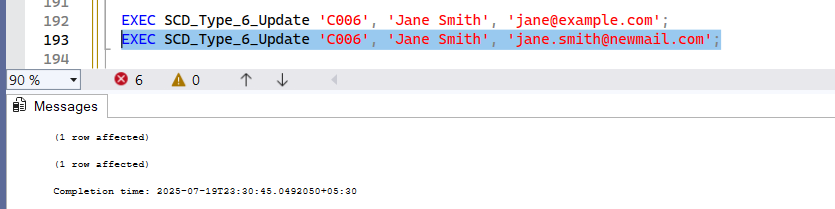
END;

1️⃣ First Time Inserting:

EXEC SCD\_Type\_6\_Update 'C006', 'Jane Smith', 'jane@example.com';

2️⃣ Trying to Insert Again with Same Code (updated name or email):

EXEC SCD\_Type\_6\_Update 'C006', 'Jane Smith', 'jane.smith@newmail.com';



Description of the execution result.

* One row will be marked as IsCurrent = 0 with an EndDate.
* The new row will have IsCurrent = 1, the updated Email, and the PreviousEmail stored.

# 9. Summary Table

Below is a summary comparison of all SCD Types.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SCD Type | Description | Tracks History? | Overwrites Data? | Separate Table? |
| 0 | Fixed | No | No | No |
| 1 | Overwrite | No | Yes | No |
| 2 | Full History | Yes | No | No |
| 3 | Partial History | Yes (1 value) | Yes | No |
| 4 | External History | Yes | Yes | Yes |
| 6 | Hybrid | Yes | Yes | No |

# 10. Conclusion

Through this documentation, we implemented and explored all six types of Slowly Changing Dimensions (SCD). Each method serves a specific business case and requirement depending on how historical data should be maintained. This document also provides a clear reference for developers working on dimensional modeling.

Sucharita Gorai