

**MAESTRÍA EN CIENCIA DE DATOS**

**COHORTE 2024-2025**

**BUENOS AIRES**

**INTRODUCCIÓN A DATA WAREHOUSING**

**TRABAJO PRÁCTICO FINAL**

Docentes: - Mg. Eduardo A. Poggi - CC. Esteban J. Alonso - Dr. Hernán Etiennot

**Alumnos:**

* **GALLARDO, Ezequiel**
* **FILIPUZZI, Juan Manuel**

Contenido

[TPG01 – Flujo de datos en un DWA 3](#_Toc197467112)

[Objetivos de trabajo y Entregables 3](#_Toc197467113)

[1. Arquitectura de la solución 5](#_Toc197467114)

[2. Carga de Schema TMP 6](#_Toc197467115)

[2.1 Proceso Schema TMP 1: CATEGORIES 6](#_Toc197467116)

[2.2 Proceso Schema TMP 2: REGIONS 8](#_Toc197467117)

[2.3 Proceso Schema TMP 3: SHIPPERS 10](#_Toc197467118)

[2.4 Proceso Schema TMP 4: TERRITORIES 12](#_Toc197467119)

[2.5 Proceso Schema TMP 5: CUSTOMERS 15](#_Toc197467120)

[2.6 Proceso Schema TMP 6: EMPLOYEE TERRITORIES 17](#_Toc197467121)

[2.7 Proceso Schema TMP 7: EMPLOYEE 19](#_Toc197467122)

[2.8 Proceso Schema TMP 8: EMPLOYEE TERRITORIES 21](#_Toc197467123)

[2.9 Proceso Schema TMP 9: ORDERS 23](#_Toc197467124)

[2.10 Proceso Schema TMP 10: ORDERS 25](#_Toc197467125)

[2.11 Proceso Schema TMP 11: SUPPLIERS 27](#_Toc197467126)

[2.12 Proceso Schema TMP 11: COUNTRIES 30](#_Toc197467127)

[3. Carga de Schema STG 32](#_Toc197467128)

[3.1 Proceso Schema STG 1: CATEGORIES 32](#_Toc197467129)

[3.2 Proceso Schema STG 2: REGIONS 35](#_Toc197467130)

[3.3 Proceso Schema STG 3: SHIPPERS 37](#_Toc197467131)

[3.4 Proceso Schema STG 4: TERRITORIES 40](#_Toc197467132)

[4. Carga de Schema DWH 42](#_Toc197467133)

[4.1 Proceso Schema DWH 1: CATEGORIES 42](#_Toc197467134)

[4.2 Proceso Schema DWH 2: REGIONS 45](#_Toc197467135)

[4.3 Proceso Schema DWH 3: SHIPPERS 47](#_Toc197467136)

[4.4 Proceso Schema DWH 4: TERRITORIES 49](#_Toc197467137)

# TPG01 – Flujo de datos en un DWA

## Objetivos de trabajo y Entregables

El objetivo de este trabajo práctico es desarrollar todas las capas de datos y ejecutar los procesos correspondientes del flujo end-to-end en un DWA (Data Warehouse Analítico), desde la ingesta hasta la publicación y la explotación.

El material básico para la elaboración del presente trabajo se encuentra publicado en la plataforma del curso, además de lo expuesto en clase.

**Aclaraciones**

• Este trabajo debe elaborarse por equipos según los grupos establecidos para la materia. Los grupos de más de tres integrantes serán penalizados.

• La entrega de este TP consiste en publicar un documento resumiendo lo realizado según se especifica más abajo, además de los componentes desarrollados.

• Cada grupo deberá exponer en clase una síntesis del trabajo realizado con una duración máxima de 10’. Podría reemplazarse con un video.

• Las fechas de publicación y presentación serán indicadas en la plataforma

• Incluyan en los archivos a entregar la lista de los integrantes. Se recomienda considerar una carátula en donde se identifique el posgrado, la materia, el título del informe, los integrantes del equipo y la fecha.

• La evaluación se realizará según la rúbrica descrita más abajo.

• Los integrantes de cada grupo obtendrán la misma calificación.

• Los docentes evalúan el trabajo realizado por lo que se manifiesta en la presentación y en los documentos entregados, por lo tanto se recomienda una elaboración cuidada y comentada. El contenido debe transmitir las tareas realizadas con la especificidad suficiente para comprenderlas pero sin entrar en detalles irrelevantes. No copien textos externos, si fuera necesario, citen la fuente.

**Informe y presentación exponiendo:**

1. Entrega de un informe y/o presentación (.PDF/.PPTX) con un resumen de lo realizado. Esto permitirá evaluar el resultado sin necesariamente abrir ningún entorno de base de datos.

2. Se deben incluir como anexos todos los scripts desarrollados, los DER y estructuras correspondientes.

3. Entregar como .ZIP la base resultante con todos los componentes (.db, .sql, etc. y los tableros) para verificación de autoría si fuera necesario.

4. Entregar el tablero desarrollado (por ejemplo, Tablero.PBIX).

5. En la presentación en clase deberán ejecutar los tableros desarrollados.

6. Salvo el informe que debe ser publicado en el aula viryual, los demás objetos pueden ser publicados en un drive con libre acceso.

## Arquitectura de la solución

La arquitectura de la solución, será la siguiente

## Carga de Schema TMP

En ésta sección de trabajo, tenemos como objetivo alimentar todas las tablas del schema TEMPORAL a partir de los files de INGESTA. Tomamos como patrón de diseño lo siguiente:

1. El input serán los flat files del directorio de INGESTA
2. Se crea un registro de Log propio de la entidad a correr.
3. Las Tablas Temporales se truncan al momento inicial.
4. Se envían todos los registros del archivo de INPUT, hacia la tabla en el schema TMP.
   1. Como premisa, todos los registros y sus respectivos campos, no reciben manipulación, para tener una copia exacta del transaccional; Motivo por el cual, todos los campos se almacenan como VARCHAR.
   2. En el caso de ser rejectada alguna fila; se almacena en un registro de Auditoría.
5. Se registra nivel auditoría el status del proceso.

Correremos todos los procesos con los FILES DE INGESTA 1.

### 2.1 Proceso Schema TMP 1: CATEGORIES

|  |  |
| --- | --- |
| JobName | TMP\_Categories.dtsx |
| Input |  |
| Control Flow |  |
| Data Flow |  |
| Data |  |

### 2.2 Proceso Schema TMP 2: REGIONS

|  |  |
| --- | --- |
| JobName | TMP\_Regions.dtsx |
| Input |  |
| Control Flow |  |
| Data Flow |  |
| Data |  |

### 2.3 Proceso Schema TMP 3: SHIPPERS

|  |  |
| --- | --- |
| JobName | TMP\_Shippers.dtsx |
| Input |  |
| Control Flow |  |
| Data Flow |  |
| Data |  |

### 2.4 Proceso Schema TMP 4: TERRITORIES

|  |  |
| --- | --- |
| JobName | TMP\_Territories.dtsx |
| Input |  |
| Control Flow |  |
| Data Flow |  |
| Data |  |

### 2.5 Proceso Schema TMP 5: CUSTOMERS

|  |  |
| --- | --- |
| JobName | TMP\_Customers.dtsx |
| Input |  |
| Control Flow |  |
| Data Flow |  |
| Data |  |

### 2.6 Proceso Schema TMP 6: EMPLOYEE TERRITORIES

|  |  |
| --- | --- |
| JobName | TMP\_Employee\_Territories.dtsx |
| Input |  |
| Control Flow |  |
| Data Flow |  |
| Data |  |

### 2.7 Proceso Schema TMP 7: EMPLOYEE

|  |  |
| --- | --- |
| JobName | TMP\_Employee.dtsx |
| Input |  |
| Control Flow |  |
| Data Flow |  |
| Data |  |

### 2.8 Proceso Schema TMP 8: EMPLOYEE TERRITORIES

|  |  |
| --- | --- |
| JobName | TMP\_Order\_Details.dtsx |
| Input |  |
| Control Flow |  |
| Data Flow |  |
| Data |  |

### 2.9 Proceso Schema TMP 9: ORDERS

|  |  |
| --- | --- |
| JobName | TMP\_Orders.dtsx |
| Input |  |
| Control Flow |  |
| Data Flow |  |
| Data |  |

### 2.10 Proceso Schema TMP 10: ORDERS

|  |  |
| --- | --- |
| JobName | TMP\_Orders.dtsx |
| Input |  |
| Control Flow |  |
| Data Flow |  |
| Data |  |

### 2.11 Proceso Schema TMP 11: SUPPLIERS

|  |  |
| --- | --- |
| JobName | TMP\_Suppliers.dtsx |
| Input |  |
| Control Flow |  |
| Data Flow |  |
| Data |  |

### 2.12 Proceso Schema TMP 11: COUNTRIES

|  |  |
| --- | --- |
| JobName | TMP\_Countries.dtsx |
| Input |  |
| Control Flow |  |
| Data Flow |  |
| Data |  |

## Carga de Schema STG

En ésta sección de trabajo, tenemos como objetivo alimentar todas las tablas del schema STAGING a partir de las tablas TMP. Ya nos hemos desconectado del origen, y ahora podemos trabajar a partir de las tablas IMPORT.

Consideraciones en éste proceso:

1. Se utiliza CONTROL DE VERSIONES, por lo cual se tiene un campo CHECKSUM para determinar si es registro nuevo o existente.
2. En la tabla de Staging, tenemos el campo AUD\_ACTION para determinar si es un INSERT, o un DELETE (registro que existe en DWH, pero ya no viene desde la tabla TMP; Borrado Lógico).
3. Se Aplica limpieza de Datos.
4. Se identifican IDs de otras dimensiones.

### 3.1 Proceso Schema STG 1: CATEGORIES

|  |  |
| --- | --- |
| JobName | STG\_Categories.dtsx |
| Input | select \* from [TMP].[CATEGORIES] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_STG\_Categories]  AS  BEGIN  SET NOCOUNT ON;  -- Truncar la tabla STG  TRUNCATE TABLE [STG].[LK\_CATEGORIES];  -- Insertar los datos desde TMP con AUD\_ROW\_ID y acción INSERT  WITH STG\_CATEGORIES AS (  SELECT  LTRIM(RTRIM(Category\_CD)) AS Category\_CD,  LTRIM(RTRIM(Category\_Name)) AS Category\_Name,  LTRIM(RTRIM(Category\_Description)) AS Category\_Description,  LTRIM(RTRIM(Category\_Picture)) AS Category\_Picture  FROM [TMP].[CATEGORIES]  WHERE ISNULL(LTRIM(RTRIM(Category\_CD)), '') <> ''  )  -- Mecanismo de Control de Versiones  INSERT INTO [STG].[LK\_CATEGORIES] (  Category\_CD,  Category\_Name,  Category\_Description,  Category\_Picture, -- Mantener el nombre correcto "Category\_Picture"  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  bt.Category\_CD,  bt.Category\_Name,  bt.Category\_Description,  bt.Category\_Picture, -- Mantener el nombre correcto "Category\_Picture"  GETDATE() AS AUD\_INSERT\_DT,  SYSTEM\_USER AS AUD\_USER\_INSERT,  'INSERT' AS AUD\_ACTION,  -- Uso de CHECKSUM sobre la concatenación de cadenas  CHECKSUM(  ISNULL(bt.Category\_CD, ''),  ISNULL(bt.Category\_Name, ''),  ISNULL(bt.Category\_Description, ''),  ISNULL(bt.Category\_Picture, '')  ) AS AUD\_ROW\_ID  FROM STG\_CATEGORIES bt;  -- Inserto los BORRADOS LÓGICOS  INSERT INTO [STG].[LK\_CATEGORIES] (  Category\_CD,  Category\_Name,  Category\_Description,  Category\_Picture,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  dwh.Category\_CD,  dwh.Category\_Name,  dwh.Category\_Description,  dwh.Category\_Picture, -- Mantener el nombre correcto "Category\_Picture"  GETDATE() AS AUD\_INSERT\_DT,  SYSTEM\_USER AS AUD\_USER\_INSERT,  'DELETE' AS AUD\_ACTION,  -- Uso de CHECKSUM sobre la concatenación de cadenas  dwh.AUD\_ROWID  FROM [DWH].[LK\_CATEGORIES] dwh  LEFT JOIN [STG].[LK\_CATEGORIES] s  ON dwh.AUD\_ROWID = s.AUD\_ROWID  WHERE 1 = 1  AND dwh.AUD\_ACTIVE\_FLAG = 1  AND s.Category\_CD IS NULL;  END; |
| OUTPUT | select \* from [STG].[LK\_CATEGORIES] |

### 3.2 Proceso Schema STG 2: REGIONS

|  |  |
| --- | --- |
| JobName | STG\_Regions.dtsx |
| Input | select \* from [TMP].[REGIONS] |
| Control Flow |  |
| Proceso | CREATE PROCEDURE [dbo].[SP\_STG\_Regions]  AS  BEGIN  SET NOCOUNT ON;  -- Truncar la tabla STG  TRUNCATE TABLE [STG].[LK\_REGIONS];  -- Insertar los datos desde TMP con AUD\_ROW\_ID y acción INSERT  WITH STG\_REGIONS AS (  SELECT  LTRIM(RTRIM(Region\_CD)) AS Region\_CD,  LTRIM(RTRIM(Region\_Description)) Region\_Description  FROM [TMP].[REGIONS]  WHERE ISNULL(LTRIM(RTRIM(Region\_CD)), '') <> ''  )  -- Mecanismo de Control de Versiones  INSERT INTO [STG].[LK\_REGIONS] (  Region\_CD,  Region\_Description,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  bt.Region\_CD,  bt.Region\_Description,  GETDATE() AS AUD\_INSERT\_DT,  SYSTEM\_USER AS AUD\_USER\_INSERT,  'INSERT' AS AUD\_ACTION,  -- Uso de CHECKSUM sobre la concatenación de cadenas  CHECKSUM(  ISNULL(bt.Region\_CD, ''),  ISNULL(bt.Region\_Description, '')  ) AS AUD\_ROW\_ID  FROM STG\_REGIONS bt;  -- Inserto los BORRADOS LÓGICOS  INSERT INTO [STG].[LK\_REGIONS] (  Region\_CD,  Region\_Description,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  dwh.Region\_CD,  dwh.Region\_Description,  GETDATE() AS AUD\_INSERT\_DT,  SYSTEM\_USER AS AUD\_USER\_INSERT,  'DELETE' AS AUD\_ACTION,  dwh.AUD\_ROWID  FROM [DWH].[LK\_REGIONS] dwh  LEFT JOIN [STG].[LK\_REGIONS] s  ON dwh.AUD\_ROWID = s.AUD\_ROWID  WHERE 1 = 1  AND dwh.AUD\_ACTIVE\_FLAG = 1  AND s.Region\_CD IS NULL;  END; |
| OUTPUT | select \* from [STG].[LK\_REGIONS] |

### 3.3 Proceso Schema STG 3: SHIPPERS

|  |  |
| --- | --- |
| JobName | STG\_Shippers.dtsx |
| Input | select \* from [TMP].[SHIPPERS] |
| Control Flow |  |
| Proceso | CREATE PROCEDURE [dbo].[SP\_STG\_Shippers]  AS  BEGIN  SET NOCOUNT ON;  -- Truncar la tabla STG  TRUNCATE TABLE [STG].[LK\_SHIPPERS];  -- Insertar los datos desde TMP con AUD\_ROW\_ID y acción INSERT  WITH STG\_SHIPPERS AS (  SELECT  LTRIM(RTRIM(Shipper\_CD)) AS Shipper\_CD,  LTRIM(RTRIM(Company\_Name)) Company\_Name,  LTRIM(RTRIM(Phone)) Phone  FROM [TMP].[SHIPPERS]  WHERE ISNULL(LTRIM(RTRIM(Shipper\_CD)), '') <> ''  )  -- Mecanismo de Control de Versiones  INSERT INTO [STG].[LK\_SHIPPERS] (  Ship\_Shipper\_CD,  Ship\_Company\_Name,  Ship\_Phone,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  bt.Shipper\_CD,  bt.Company\_Name,  bt.Phone,  GETDATE() AS AUD\_INSERT\_DT,  SYSTEM\_USER AS AUD\_USER\_INSERT,  'INSERT' AS AUD\_ACTION,  -- Uso de CHECKSUM sobre la concatenación de cadenas  CHECKSUM(  ISNULL(bt.Shipper\_CD, ''),  ISNULL(bt.Company\_Name, ''),  ISNULL(bt.Phone, '')  ) AS AUD\_ROW\_ID  FROM STG\_SHIPPERS bt;  -- Inserto los BORRADOS LÓGICOS  INSERT INTO [STG].[LK\_SHIPPERS] (  Ship\_Shipper\_CD,  Ship\_Company\_Name,  Ship\_Phone,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  dwh.Ship\_Shipper\_CD,  dwh.Ship\_Company\_Name,  dwh.Ship\_Phone,  GETDATE() AS AUD\_INSERT\_DT,  SYSTEM\_USER AS AUD\_USER\_INSERT,  'DELETE' AS AUD\_ACTION,  dwh.AUD\_ROWID  FROM [DWH].[LK\_SHIPPERS] dwh  LEFT JOIN [STG].[LK\_SHIPPERS] s  ON dwh.AUD\_ROWID = s.AUD\_ROWID  WHERE 1 = 1  AND dwh.AUD\_ACTIVE\_FLAG = 1  AND s.Ship\_Shipper\_CD IS NULL;  END; |
| OUTPUT | select \* from [STG].[LK\_SHIPPERS] |

### 3.4 Proceso Schema STG 4: TERRITORIES

|  |  |
| --- | --- |
| JobName | STG\_Territories.dtsx |
| Input | select \* from [TMP].[TERRITORIES] |
| Control Flow |  |
| Proceso | CREATE PROCEDURE [dbo].[SP\_STG\_Territories]  AS  BEGIN  SET NOCOUNT ON;  -- Truncar la tabla STG  TRUNCATE TABLE [STG].[LK\_TERRITORIES];  -- Insertar los datos desde TMP con AUD\_ROW\_ID y acción INSERT  WITH STG\_TERRITORIES AS (  SELECT  LTRIM(RTRIM(Territory\_CD)) AS Territory\_CD,  LTRIM(RTRIM(Territory\_Description)) Territory\_Description,  COALESCE(d.Region\_Id,-1) Region\_Id  FROM [TMP].[TERRITORIES] s  LEFT JOIN [DWH].[LK\_REGIONS] d  ON LTRIM(RTRIM(s.Region\_CD))= d.Region\_CD  WHERE ISNULL(LTRIM(RTRIM(Territory\_CD)), '') <> ''  AND d.AUD\_ACTIVE\_FLAG = 1  )  -- Mecanismo de Control de Versiones  INSERT INTO [STG].[LK\_TERRITORIES] (  Territory\_CD,  Territory\_Description,  Territory\_Region\_Id,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  bt.Territory\_CD,  bt.Territory\_Description,  bt.Region\_Id,  GETDATE() AS AUD\_INSERT\_DT,  SYSTEM\_USER AS AUD\_USER\_INSERT,  'INSERT' AS AUD\_ACTION,  -- Uso de CHECKSUM sobre la concatenación de cadenas  CHECKSUM(  ISNULL(bt.Territory\_CD, ''),  ISNULL(bt.Territory\_Description, ''),  ISNULL(bt.Region\_Id, '')  ) AS AUD\_ROW\_ID  FROM STG\_TERRITORIES bt;  -- Inserto los BORRADOS LÓGICOS  INSERT INTO [STG].[LK\_TERRITORIES] (  Territory\_CD,  Territory\_Description,  Territory\_Region\_Id,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  dwh.Territory\_CD,  dwh.Territory\_Description,  dwh.Territory\_Region\_Id,  GETDATE() AS AUD\_INSERT\_DT,  SYSTEM\_USER AS AUD\_USER\_INSERT,  'DELETE' AS AUD\_ACTION,  dwh.AUD\_ROWID  FROM [DWH].[LK\_TERRITORIES] dwh  LEFT JOIN [STG].[LK\_TERRITORIES] s  ON dwh.AUD\_ROWID = s.AUD\_ROWID  WHERE 1 = 1  AND dwh.AUD\_ACTIVE\_FLAG = 1  AND s.Territory\_CD IS NULL;  END; |
| OUTPUT | select \* from [STG].[LK\_TERRITORIES] |

## Carga de Schema DWH

En éste apartado, cargaremos las tablas del schema DWH en función de los datos de STAGING.

Se implementa mediante el uso de MERGE.

### 4.1 Proceso Schema DWH 1: CATEGORIES

|  |  |
| --- | --- |
| JobName | DWH\_Categories.dtsx |
| Input | select \* from [STG].[CATEGORIES] |
| Control Flow |  |
| Proceso | CREATE PROCEDURE [dbo].[SP\_DWH\_Categories]  AS  BEGIN  SET NOCOUNT ON;  -- Realizando el MERGE en la tabla de destino con la tabla fuente  MERGE INTO [DWH].[LK\_CATEGORIES] AS target  USING [STG].[LK\_CATEGORIES] AS source  ON target.AUD\_ROWID = source.AUD\_ROWID  AND target.AUD\_ACTIVE\_FLAG = 1 -- 1 representa TRUE para AUD\_ACTIVE\_FLAG en SQL Server  WHEN NOT MATCHED BY TARGET AND source.AUD\_ACTION = 'INSERT' THEN  INSERT (  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID,  Category\_CD,  Category\_Name,  Category\_Description,  Category\_Picture  )  VALUES (  GETDATE(), -- AUD\_INSERT\_DT  SYSTEM\_USER, -- AUD\_USER\_INSERT  1, -- AUD\_ACTIVE\_FLAG (TRUE)  source.AUD\_ROWID,  source.Category\_CD,  source.Category\_Name,  source.Category\_Description,  source.Category\_Picture  )  WHEN MATCHED AND target.AUD\_ACTIVE\_FLAG = 1 AND source.AUD\_ACTION = 'DELETE' THEN  UPDATE SET  target.AUD\_INSERT\_DT = GETDATE(), -- Mantener el campo de fecha actualizado  target.AUD\_USER\_INSERT = SYSTEM\_USER, -- Mantener el usuario actualizado  target.AUD\_ACTIVE\_FLAG = 0 -- 0 representa FALSE para AUD\_ACTIVE\_FLAG  ;  END; |
| OUTPUT | select \* from [DWH].[LK\_CATEGORIES] |

### 4.2 Proceso Schema DWH 2: REGIONS

|  |  |
| --- | --- |
| JobName | DWH\_Regions.dtsx |
| Input | select \* from [STG].[REGIONS] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_DWH\_Regions] Script Date: 6/5/2025 23:01:26 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_DWH\_Regions]  AS  BEGIN  SET NOCOUNT ON;  -- Realizando el MERGE en la tabla de destino con la tabla fuente  MERGE INTO [DWH].[LK\_REGIONS] AS target  USING [STG].[LK\_REGIONS] AS source  ON target.AUD\_ROWID = source.AUD\_ROWID  AND target.AUD\_ACTIVE\_FLAG = 1  WHEN NOT MATCHED BY TARGET AND source.AUD\_ACTION = 'INSERT' THEN  INSERT (  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID,  Region\_CD,  Region\_Description  )  VALUES (  GETDATE(), -- AUD\_INSERT\_DT  SYSTEM\_USER, -- AUD\_USER\_INSERT  1, -- AUD\_ACTIVE\_FLAG (TRUE)  source.AUD\_ROWID,  source.Region\_CD,  source.Region\_Description  )  WHEN MATCHED AND target.AUD\_ACTIVE\_FLAG = 1 AND source.AUD\_ACTION = 'DELETE' THEN  UPDATE SET  target.AUD\_INSERT\_DT = GETDATE(), -- Mantener el campo de fecha actualizado  target.AUD\_USER\_INSERT = SYSTEM\_USER, -- Mantener el usuario actualizado  target.AUD\_ACTIVE\_FLAG = 0 -- 0 representa FALSE para AUD\_ACTIVE\_FLAG  ;  END; |
| OUTPUT | select \* from [DWH].[LK\_REGIONS] |

### 4.3 Proceso Schema DWH 3: SHIPPERS

|  |  |
| --- | --- |
| JobName | DWH\_Shippers.dtsx |
| Input | select \* from [STG].[SHIPPERS] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_DWH\_Regions] Script Date: 6/5/2025 23:05:21 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE PROCEDURE [dbo].[SP\_DWH\_Shippers]  AS  BEGIN  SET NOCOUNT ON;  -- Realizando el MERGE en la tabla de destino con la tabla fuente  MERGE INTO [DWH].[LK\_SHIPPERS] AS target  USING [STG].[LK\_SHIPPERS] AS source  ON target.AUD\_ROWID = source.AUD\_ROWID  AND target.AUD\_ACTIVE\_FLAG = 1  WHEN NOT MATCHED BY TARGET AND source.AUD\_ACTION = 'INSERT' THEN  INSERT (  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID,  Ship\_Shipper\_CD,  Ship\_Company\_Name,  Ship\_Phone  )  VALUES (  GETDATE(), -- AUD\_INSERT\_DT  SYSTEM\_USER, -- AUD\_USER\_INSERT  1, -- AUD\_ACTIVE\_FLAG (TRUE)  source.AUD\_ROWID,  source.Ship\_Shipper\_CD,  source.Ship\_Company\_Name,  source.Ship\_Phone  )  WHEN MATCHED AND target.AUD\_ACTIVE\_FLAG = 1 AND source.AUD\_ACTION = 'DELETE' THEN  UPDATE SET  target.AUD\_INSERT\_DT = GETDATE(), -- Mantener el campo de fecha actualizado  target.AUD\_USER\_INSERT = SYSTEM\_USER, -- Mantener el usuario actualizado  target.AUD\_ACTIVE\_FLAG = 0 -- 0 representa FALSE para AUD\_ACTIVE\_FLAG  ;  END; |
| OUTPUT | select \* from [DWH].[LK\_SHIPPERS] |

### 4.4 Proceso Schema DWH 4: TERRITORIES

|  |  |
| --- | --- |
| JobName | DWH\_Territories.dtsx |
| Input | select \* from [STG].[Territories] |
| Control Flow |  |
| Proceso | CREATE PROCEDURE [dbo].[SP\_DWH\_Territories]  AS  BEGIN  SET NOCOUNT ON;  -- Realizando el MERGE en la tabla de destino con la tabla fuente  MERGE INTO [DWH].[LK\_TERRITORIES] AS target  USING [STG].[LK\_TERRITORIES] AS source  ON target.AUD\_ROWID = source.AUD\_ROWID  AND target.AUD\_ACTIVE\_FLAG = 1  WHEN NOT MATCHED BY TARGET AND source.AUD\_ACTION = 'INSERT' THEN  INSERT (  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID,  Territory\_CD,  Territory\_Description,  Territory\_Region\_Id  )  VALUES (  GETDATE(), -- AUD\_INSERT\_DT  SYSTEM\_USER, -- AUD\_USER\_INSERT  1, -- AUD\_ACTIVE\_FLAG (TRUE)  source.AUD\_ROWID,  source.Territory\_CD,  source.Territory\_Description,  source.Territory\_Region\_Id  )  WHEN MATCHED AND target.AUD\_ACTIVE\_FLAG = 1 AND source.AUD\_ACTION = 'DELETE' THEN  UPDATE SET  target.AUD\_INSERT\_DT = GETDATE(), -- Mantener el campo de fecha actualizado  target.AUD\_USER\_INSERT = SYSTEM\_USER, -- Mantener el usuario actualizado  target.AUD\_ACTIVE\_FLAG = 0 -- 0 representa FALSE para AUD\_ACTIVE\_FLAG  ;  END; |
| OUTPUT | select \* from [DWH].[LK\_TERRITORIES] |