

**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 4](#_Toc198763905)

[Objetivos de trabajo y Entregables 4](#_Toc198763906)

[1. Arquitectura de la solución 6](#_Toc198763907)

[1.1 Estructura de Pipelines 7](#_Toc198763908)

[1.2 Estrategia de Control de Cambios 8](#_Toc198763909)

[1.3 Orquestación de procesos 9](#_Toc198763910)

[2. Procesos de ingesta de Schema TMP 10](#_Toc198763911)

[2.1 Proceso Schema TMP 1: CATEGORIES 10](#_Toc198763912)

[2.2 Proceso Schema TMP 2: REGIONS 12](#_Toc198763913)

[2.3 Proceso Schema TMP 3: SHIPPERS 14](#_Toc198763914)

[2.4 Proceso Schema TMP 4: TERRITORIES 16](#_Toc198763915)

[2.5 Proceso Schema TMP 5: CUSTOMERS 19](#_Toc198763916)

[2.6 Proceso Schema TMP 6: EMPLOYEE TERRITORIES 21](#_Toc198763917)

[2.7 Proceso Schema TMP 7: EMPLOYEE 23](#_Toc198763918)

[2.8 Proceso Schema TMP 8: EMPLOYEE TERRITORIES 25](#_Toc198763919)

[2.9 Proceso Schema TMP 9: ORDERS 27](#_Toc198763920)

[2.10 Proceso Schema TMP 10: ORDERS 29](#_Toc198763921)

[2.11 Proceso Schema TMP 11: SUPPLIERS 31](#_Toc198763922)

[2.12 Proceso Schema TMP 12: COUNTRIES 34](#_Toc198763923)

[2.13 Proceso Schema TMP 13: PRODUCTS 36](#_Toc198763924)

[3. Procesos de ingesta de Schema STG 38](#_Toc198763925)

[3.1 Proceso Schema STG 1: CATEGORIES 38](#_Toc198763926)

[3.2 Proceso Schema STG 2: REGIONS 41](#_Toc198763927)

[3.3 Proceso Schema STG 3: SHIPPERS 43](#_Toc198763928)

[3.4 Proceso Schema STG 4: TERRITORIES 46](#_Toc198763929)

[3.5 Proceso Schema STG 5: COUNTRIES 48](#_Toc198763930)

[3.6 Proceso Schema STG 6: EMPLOYEES 50](#_Toc198763931)

[3.7 Proceso Schema STG 7: SUPPLIERS 54](#_Toc198763932)

[3.8 Proceso Schema STG 8: SUPPLIERS DATOS CONTACTO 57](#_Toc198763933)

[3.9 Proceso Schema STG 9: PRODUCTS 60](#_Toc198763934)

[3.10 Proceso Schema STG 10: EMPLOYEE\_TERRITORIES 63](#_Toc198763935)

[3.11 Proceso Schema STG 11: CUSTOMERS 65](#_Toc198763936)

[3.12 Proceso Schema STG 12: ORDERS 68](#_Toc198763937)

[3.13 Proceso Schema STG 13: ORDERS DETAILS 72](#_Toc198763938)

[4. Procesos de ingesta de Schema DWH (Dimensiones) 75](#_Toc198763939)

[4.1 Proceso Schema DWH 1: CATEGORIES 75](#_Toc198763940)

[4.2 Proceso Schema DWH 2: REGIONS 77](#_Toc198763941)

[4.3 Proceso Schema DWH 3: SHIPPERS 80](#_Toc198763942)

[4.4 Proceso Schema DWH 4: TERRITORIES 82](#_Toc198763943)

[4.5 Proceso Schema DWH 5: COUNTRIES 84](#_Toc198763944)

[4.6 Proceso Schema DWH 6: EMPLOYEES 87](#_Toc198763945)

[4.7 Proceso Schema DWH 7: SUPPLIERS 91](#_Toc198763946)

[4.8 Proceso Schema DWH 8: SUPPLIERS DATOS CONTACTO 94](#_Toc198763947)

[4.9 Proceso Schema DWH 9: PRODUCTS 97](#_Toc198763948)

[4.10 Proceso Schema DWH 10: EMPLOYEES TERRITORIES 100](#_Toc198763949)

[4.11 Proceso Schema DWH 11: LK\_TIEMPO 102](#_Toc198763950)

[5. Procesos de ingesta de Schema ODS 104](#_Toc198763951)

[5.1 Proceso Schema ODS 1: CUSTOMERS 104](#_Toc198763952)

[5.2 Proceso Schema ODS 2: ORDERS 107](#_Toc198763953)

[5.3 Proceso Schema ODS 3: ORDERS DETAILS 110](#_Toc198763954)

[6. Gestión de METADATA & Memoria institucional 113](#_Toc198763955)

[7. DQM Gestión de Logs 114](#_Toc198763956)

# 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).

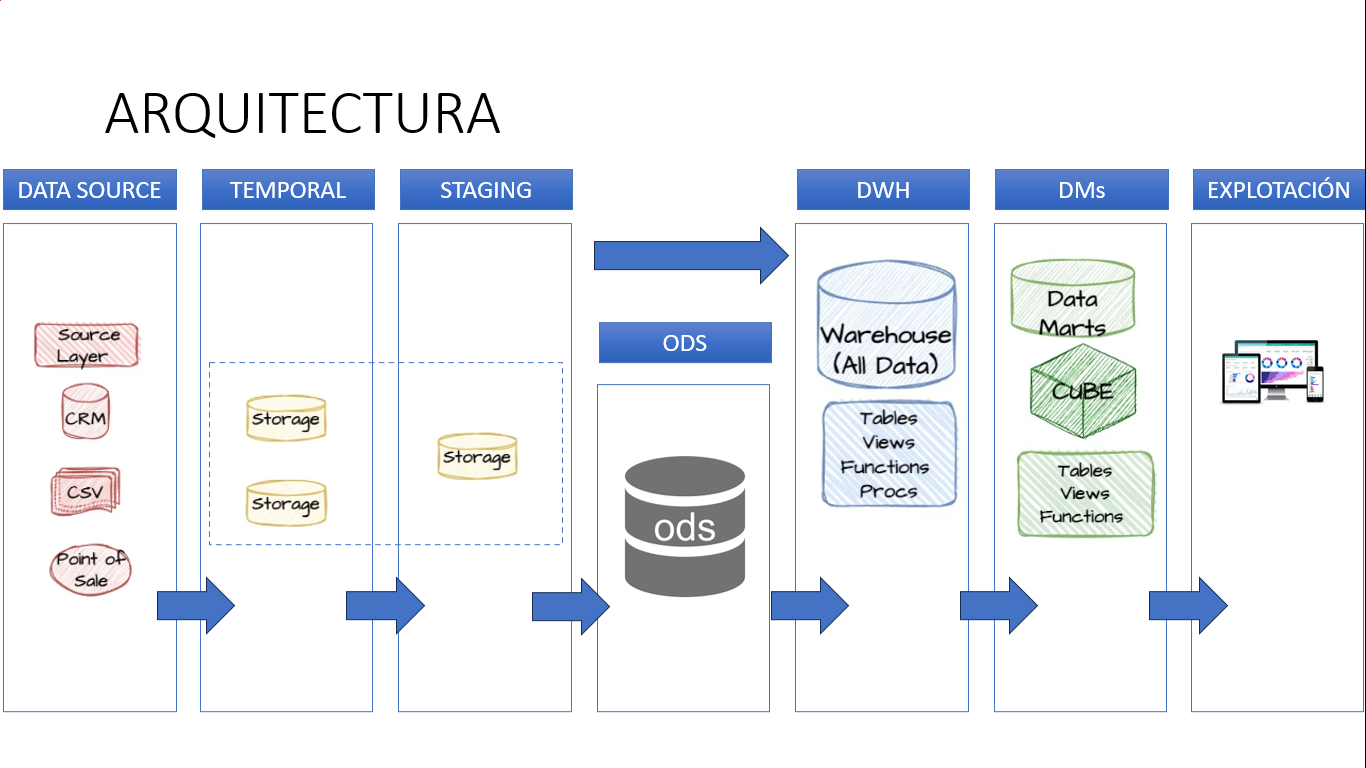
5. En la presentación en clase deberán ejecutar los tableros desarrollados.

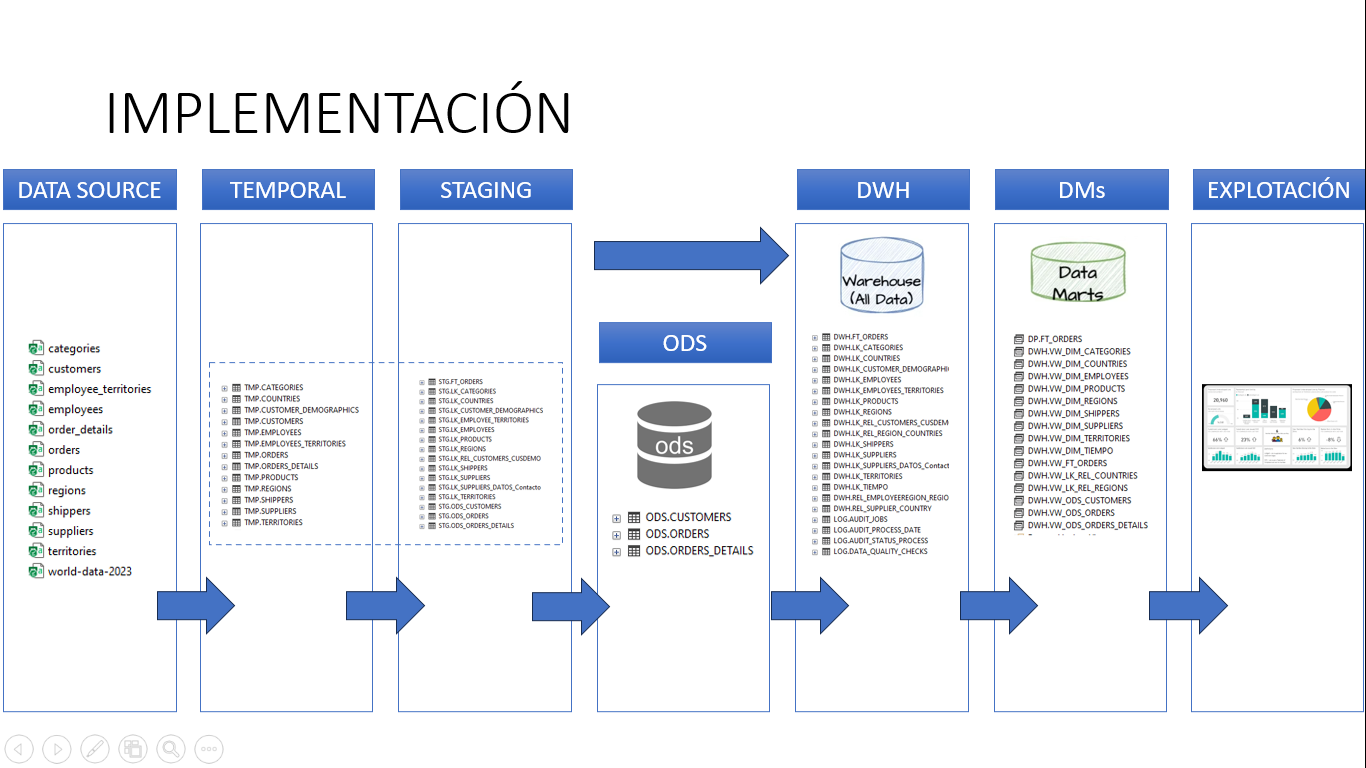
6. Salvo el informe que debe ser publicado en el aula viryual, los demás objetos pueden ser publicados en un drive con libre acceso.

## Arquitectura de la solución

La arquitectura de la solución, será la siguiente. Los puntos importantes de la misma son:

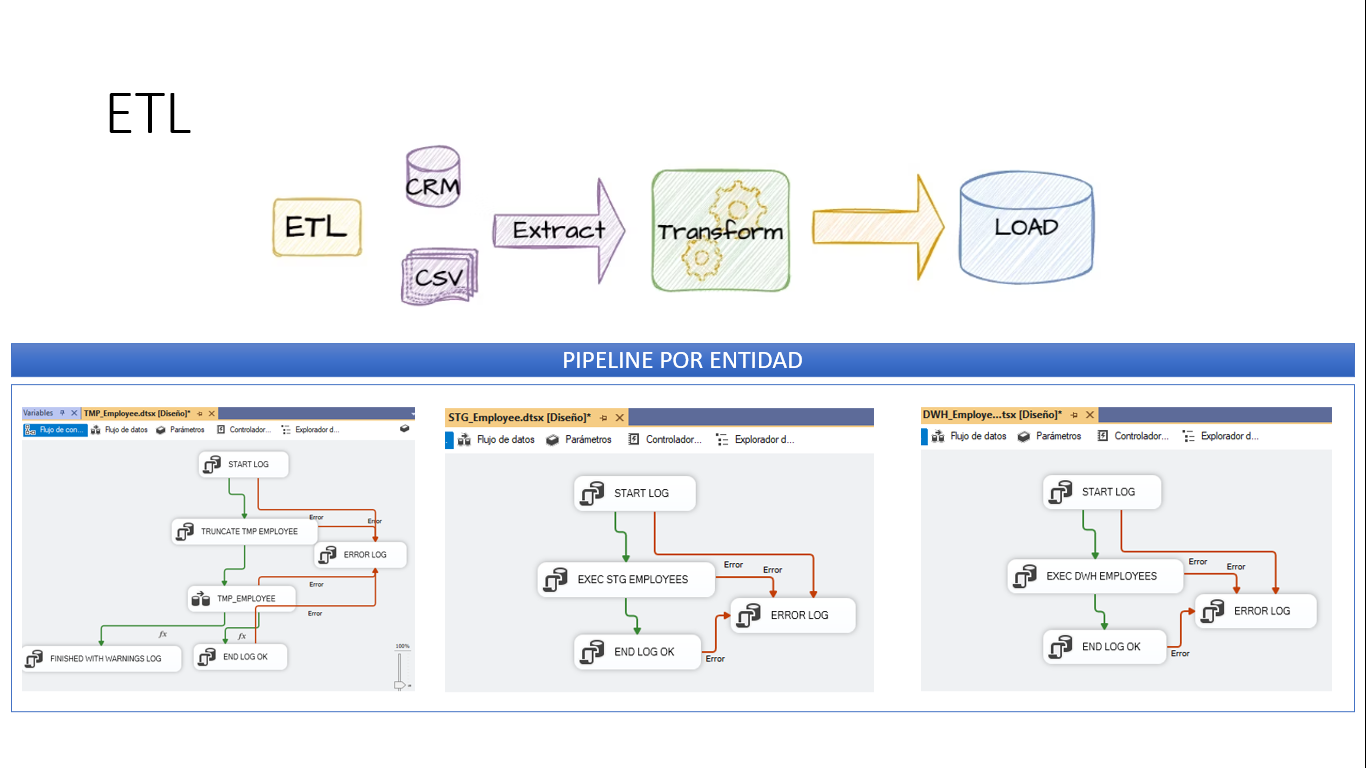
1. Esquema TEMPORAL: “TMP”; con el objeto de desconectar la fuente; Es una copia intacta del origen.
2. Esquema STAGING; “STG” área donde se almacenan en forma temporal los datos, en la cual ,se aplican transformaciones de limpieza y join con otros objetos, con el objeto de agregar información.
3. Esquema ODS;
4. Esquema DWH; Donde se almacenan las dimensiones y fact tables.
5. Esquema DP, para los Data Marts.





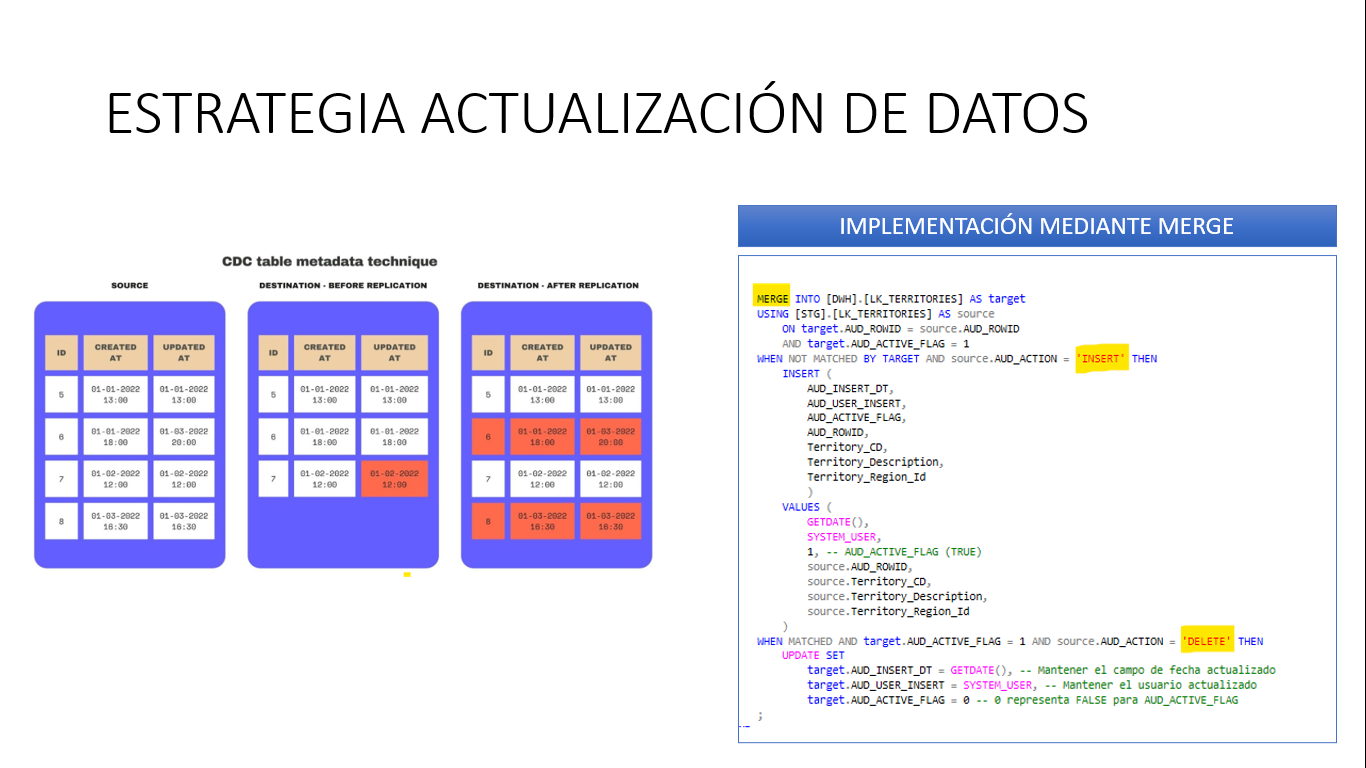
### 1.1 Estructura de Pipelines

Los ETLs se desarrollan con SSIS. La estructura es la siguiente.



### Estrategia de Control de Cambios

Para asegurar la integridad y trazabilidad de los datos en mi Data Warehouse, implemento un robusto sistema de control de cambios. Utilizo la detección de novedades mediante la comparación del ROWID generado por el HASH de todos los campos relevantes. Este enfoque me permite identificar cambios de manera eficiente y precisa. Además, para mantener un histórico completo, utilizo un FLAG de ACTIVO que indica la versión actual de cada registro, mientras que las versiones anteriores se desactivan, asegurando que siempre disponga de la información más reciente y precisa para análisis y reporting.



### Orquestación de procesos

Se han desarrollado 3 Orquestadores, encargados de la invocación de los procesos TEMPORAL, STAGING y DWH.



## Procesos de ingesta 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 12: COUNTRIES

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

### 2.13 Proceso Schema TMP 13: PRODUCTS

|  |  |
| --- | --- |
| JobName | TMP\_Products.dtsx |
| Input |  |
| Control Flow |  |
| Data Flow |  |
| Data | select \* from [TMP].[PRODUCTS] |

## Procesos de ingesta 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] |

### 3.5 Proceso Schema STG 5: COUNTRIES

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

### 3.6 Proceso Schema STG 6: EMPLOYEES

|  |  |
| --- | --- |
| JobName | STG\_Employees.dtsx |
| Input | select \* from [TMP].[EMPLOYEES] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_STG\_Employees] Script Date: 17/5/2025 14:33:36 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_STG\_Employees]  AS  BEGIN  SET NOCOUNT ON;  -- Truncar la tabla STG  TRUNCATE TABLE [STG].[LK\_EMPLOYEES];  -- Insertar los datos desde TMP con AUD\_ROW\_ID y acción INSERT  WITH EMPLOYEES\_REPORT AS  ( SELECT \*  FROM [DWH].[LK\_EMPLOYEES]  WHERE AUD\_ACTIVE\_FLAG = 1  ),  REGIONS AS  ( SELECT \*  FROM [DWH].VW\_LK\_REL\_REGIONS  ),  STG\_EMPLOYEES AS (  SELECT  LTRIM(RTRIM(Employee\_CD)) AS Emp\_Employee\_CD,  LTRIM(RTRIM(Last\_name)) Emp\_Last\_name,  LTRIM(RTRIM(First\_name)) Emp\_First\_name,  LTRIM(RTRIM(title)) Emp\_title,  LTRIM(RTRIM(title\_of\_courtesy)) Emp\_title\_of\_courtesy,  CONVERT(DATE,LTRIM(RTRIM(birth\_date))) Emp\_birth\_date,  CONVERT(DATE,LTRIM(RTRIM(hire\_date))) Emp\_hire\_date,  LTRIM(RTRIM(address)) Emp\_address,  LTRIM(RTRIM(city)) Emp\_city,  COALESCE(r.Region\_id,-1) Emp\_Region\_id,  LTRIM(RTRIM(postal\_Code)) Emp\_postal\_Code,  LTRIM(RTRIM(home\_phone)) Emp\_home\_phone,  LTRIM(RTRIM(extension)) Emp\_extension,  LTRIM(RTRIM(photo)) Emp\_photo,  LTRIM(RTRIM(notes)) Emp\_notes,  COALESCE(er.Emp\_Employee\_id,-1) Emp\_report\_to\_id,  LTRIM(RTRIM(photo\_path)) Emp\_photo\_path,  CONVERT(int, CASE WHEN [report\_to] = 'NULL' THEN -1 else [report\_to] END) Emp\_report\_to\_cd  FROM [TMP].[EMPLOYEES] s  LEFT JOIN EMPLOYEES\_REPORT d  ON LTRIM(RTRIM(s.Employee\_CD))= d.Emp\_Employee\_CD  LEFT JOIN REGIONS r  ON LTRIM(RTRIM(s.region\_cd))= r.Region\_CD  LEFT JOIN EMPLOYEES\_REPORT er  ON LTRIM(RTRIM(s.report\_to))= er.Emp\_Employee\_CD  WHERE ISNULL(LTRIM(RTRIM(Employee\_CD)), '') <> ''  )  -- Mecanismo de Control de Versiones  INSERT INTO [STG].[LK\_EMPLOYEES] (  Emp\_Employee\_CD,  Emp\_Last\_name,  Emp\_First\_name,  Emp\_title,  Emp\_title\_of\_courtesy,  Emp\_birth\_date,  Emp\_hire\_date,  Emp\_address,  Emp\_city,  Emp\_Region\_id,  Emp\_postal\_Code,  Emp\_home\_phone,  Emp\_extension,  Emp\_photo,  Emp\_notes,  Emp\_report\_to\_id,  Emp\_photo\_path,  Emp\_report\_to\_cd,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  bt.Emp\_Employee\_CD,  bt.Emp\_Last\_name,  bt.Emp\_First\_name,  bt.Emp\_title,  bt.Emp\_title\_of\_courtesy,  bt.Emp\_birth\_date,  bt.Emp\_hire\_date,  bt.Emp\_address,  bt.Emp\_city,  bt.Emp\_Region\_id,  bt.Emp\_postal\_Code,  bt.Emp\_home\_phone,  bt.Emp\_extension,  bt.Emp\_photo,  bt.Emp\_notes,  bt.Emp\_report\_to\_id,  bt.Emp\_photo\_path,  bt.Emp\_report\_to\_cd,  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.Emp\_Employee\_CD, ''),  ISNULL(bt.Emp\_Employee\_CD,''),  ISNULL(bt.Emp\_Last\_name,''),  ISNULL(bt.Emp\_First\_name,''),  ISNULL(bt.Emp\_title,''),  ISNULL(bt.Emp\_title\_of\_courtesy,''),  ISNULL(bt.Emp\_birth\_date,''),  ISNULL(bt.Emp\_hire\_date,''),  ISNULL(bt.Emp\_address,''),  ISNULL(bt.Emp\_city,''),  ISNULL(bt.Emp\_Region\_id,''),  ISNULL(bt.Emp\_postal\_Code,''),  ISNULL(bt.Emp\_home\_phone,''),  ISNULL(bt.Emp\_extension,''),  ISNULL(bt.Emp\_photo,''),  ISNULL(bt.Emp\_notes,''),  ISNULL(bt.Emp\_report\_to\_id,''),  ISNULL(bt.Emp\_photo\_path,'')  ) AS AUD\_ROW\_ID  FROM STG\_EMPLOYEES bt;  -- Inserto los BORRADOS LÓGICOS  INSERT INTO [STG].[LK\_EMPLOYEES] (  Emp\_Employee\_CD,  Emp\_Last\_name,  Emp\_First\_name,  Emp\_title,  Emp\_title\_of\_courtesy,  Emp\_birth\_date,  Emp\_hire\_date,  Emp\_address,  Emp\_city,  Emp\_Region\_id,  Emp\_postal\_Code,  Emp\_home\_phone,  Emp\_extension,  Emp\_photo,  Emp\_notes,  Emp\_report\_to\_id,  Emp\_photo\_path,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  dwh.Emp\_Employee\_CD,  dwh.Emp\_Last\_name,  dwh.Emp\_First\_name,  dwh.Emp\_title,  dwh.Emp\_title\_of\_courtesy,  dwh.Emp\_birth\_date,  dwh.Emp\_hire\_date,  dwh.Emp\_address,  dwh.Emp\_city,  dwh.Emp\_Region\_id,  dwh.Emp\_postal\_Code,  dwh.Emp\_home\_phone,  dwh.Emp\_extension,  dwh.Emp\_photo,  dwh.Emp\_notes,  dwh.Emp\_report\_to\_id,  dwh.Emp\_photo\_path,  GETDATE() AS AUD\_INSERT\_DT,  SYSTEM\_USER AS AUD\_USER\_INSERT,  'DELETE' AS AUD\_ACTION,  dwh.AUD\_ROWID  FROM [DWH].[LK\_EMPLOYEES] dwh  LEFT JOIN [STG].[LK\_EMPLOYEES] s  ON dwh.AUD\_ROWID = s.AUD\_ROWID  WHERE 1 = 1  AND dwh.AUD\_ACTIVE\_FLAG = 1  AND s.Emp\_Employee\_CD IS NULL  AND dwh.Emp\_Employee\_id<>-1;  END; |
| OUTPUT | select \* from [STG].[LK\_EMPLOYEES] |

### 3.7 Proceso Schema STG 7: SUPPLIERS

|  |  |
| --- | --- |
| JobName | STG\_Suppliers.dtsx |
| Input | select \* from [TMP].[SUPPLIERS] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_STG\_Suppliers] Script Date: 17/5/2025 14:15:15 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_STG\_Suppliers]  AS  BEGIN  SET NOCOUNT ON;  -- Truncar la tabla STG  TRUNCATE TABLE [STG].[LK\_SUPPLIERS];  -- Insertar los datos desde TMP con AUD\_ROW\_ID y acción INSERT  WITH REGIONS AS  (SELECT \* from [DWH].[LK\_REGIONS]  WHERE AUD\_ACTIVE\_FLAG = 1),  COUNTRIES AS  (SELECT \* from DWH.VW\_LK\_REL\_COUNTRIES  ),  STG\_SUPPLIERS AS (  SELECT  LTRIM(RTRIM(Supp\_CD)) AS Supp\_CD,  LTRIM(RTRIM(Supp\_companyName)) Supp\_companyName,  LTRIM(RTRIM(Supp\_contactName)) Supp\_contactName,  LTRIM(RTRIM(Supp\_contactTitle)) Supp\_contactTitle,  LTRIM(RTRIM(Supp\_address)) Supp\_address,  LTRIM(RTRIM(Supp\_City)) Supp\_City,  COALESCE(d.Region\_Id,-1) Supp\_Id\_Region,  LTRIM(RTRIM(Supp\_PostalCode))Supp\_PostalCode,  COALESCE(c.Ctry\_Country\_Id,-1)Supp\_Id\_Country,  CONVERT(integer, '-1') Supp\_ID\_DatoContacto  FROM [TMP].[SUPPLIERS] s  LEFT JOIN REGIONS d  ON LTRIM(RTRIM(s.Supp\_Region))= d.Region\_CD  LEFT JOIN COUNTRIES c  ON LTRIM(RTRIM(s.Supp\_Country))= c.Country\_CD  WHERE ISNULL(LTRIM(RTRIM(Supp\_CD)), '') <> ''  )  -- Mecanismo de Control de Versiones  INSERT INTO [STG].[LK\_SUPPLIERS] (  Supp\_CD,  Supp\_companyName,  Supp\_contactName,  Supp\_contactTitle,  Supp\_address,  Supp\_City,  Supp\_Id\_Region,  Supp\_PostalCode,  Supp\_Id\_Country,  Supp\_ID\_DatoContacto,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  bt.Supp\_CD,  bt.Supp\_companyName,  bt.Supp\_contactName,  bt.Supp\_contactTitle,  bt.Supp\_address,  bt.Supp\_City,  Supp\_Id\_Region,  Supp\_PostalCode,  Supp\_Id\_Country,  Supp\_ID\_DatoContacto,  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.Supp\_CD, ''),  ISNULL(bt.Supp\_companyName, ''),  ISNULL(bt.Supp\_contactName, ''),  ISNULL(bt.Supp\_contactTitle, ''),  ISNULL(bt.Supp\_address, ''),  ISNULL(bt.Supp\_City, ''),  ISNULL(bt.Supp\_Id\_Region, ''),  ISNULL(bt.Supp\_PostalCode, ''),  ISNULL(bt.Supp\_Id\_Country, ''),  ISNULL(bt.Supp\_ID\_DatoContacto, '')  ) AS AUD\_ROW\_ID  FROM STG\_SUPPLIERS bt;  -- Inserto los BORRADOS LÓGICOS  INSERT INTO [STG].[LK\_SUPPLIERS] (  Supp\_CD,  Supp\_companyName,  Supp\_contactName,  Supp\_contactTitle,  Supp\_address,  Supp\_City,  Supp\_Id\_Region,  Supp\_PostalCode,  Supp\_Id\_Country,  Supp\_ID\_DatoContacto,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  dwh.Supp\_CD,  dwh.Supp\_companyName,  dwh.Supp\_contactName,  dwh.Supp\_contactTitle,  dwh.Supp\_address,  dwh.Supp\_City,  dwh.Supp\_Id\_Region,  dwh.Supp\_PostalCode,  dwh.Supp\_Id\_Country,  dwh.Supp\_ID\_DatoContacto,  GETDATE() AS AUD\_INSERT\_DT,  SYSTEM\_USER AS AUD\_USER\_INSERT,  'DELETE' AS AUD\_ACTION,  dwh.AUD\_ROWID  FROM [DWH].[LK\_SUPPLIERS] dwh  LEFT JOIN [STG].[LK\_SUPPLIERS] s  ON dwh.AUD\_ROWID = s.AUD\_ROWID  WHERE 1 = 1  AND dwh.AUD\_ACTIVE\_FLAG = 1  AND s.Supp\_CD IS NULL  AND dwh.Supp\_Supplier\_Id <> -1;  END; |
| OUTPUT | select \* from [STG].[LK\_EMPLOYEES] |

### 3.8 Proceso Schema STG 8: SUPPLIERS DATOS CONTACTO

|  |  |
| --- | --- |
| JobName | STG\_Suppliers.dtsx |
| Input | select \* from [TMP].[SUPPLIERS] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_STG\_Suppliers\_Datos\_Contacto] Script Date: 16/5/2025 19:04:54 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_STG\_Suppliers\_Datos\_Contacto]  AS  BEGIN  SET NOCOUNT ON;  -- Truncar la tabla STG  TRUNCATE TABLE [STG].LK\_SUPPLIERS\_DATOS\_Contacto;  -- Insertar los datos desde TMP con AUD\_ROW\_ID y acción INSERT  WITH SUPPLIERS AS  (SELECT \* from [DWH].[LK\_SUPPLIERS]  WHERE AUD\_ACTIVE\_FLAG = 1),  STG\_SUPPLIERS AS (  SELECT  d.Supp\_Supplier\_Id AS Dat\_SupplierId,  LTRIM(RTRIM(Supp\_Phone)) Dat\_Phone,  LTRIM(RTRIM(Supp\_Fax)) Dat\_Fax,  LTRIM(RTRIM(Supp\_HomePage)) Dat\_HomePage  FROM [TMP].[SUPPLIERS] s  INNER JOIN SUPPLIERS d  ON LTRIM(RTRIM(s.Supp\_CD))= d.Supp\_CD  WHERE ISNULL(LTRIM(RTRIM(s.Supp\_CD)), '') <> ''  )  -- Mecanismo de Control de Versiones  INSERT INTO [STG].LK\_SUPPLIERS\_DATOS\_Contacto (  Dat\_SupplierId,  Dat\_Phone,  Dat\_Fax,  Dat\_HomePage,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  bt.Dat\_SupplierId,  bt.Dat\_Phone,  bt.Dat\_Fax,  bt.Dat\_HomePage,  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.Dat\_SupplierId, ''),  ISNULL(bt.Dat\_Phone, ''),  ISNULL(bt.Dat\_Fax, ''),  ISNULL(bt.Dat\_HomePage, '')  ) AS AUD\_ROW\_ID  FROM STG\_SUPPLIERS bt;  -- Inserto los BORRADOS LÓGICOS  INSERT INTO [STG].[LK\_SUPPLIERS\_DATOS\_Contacto] (  Dat\_SupplierId,  Dat\_Phone,  Dat\_Fax,  Dat\_HomePage,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  dwh.Dat\_SupplierId,  dwh.Dat\_Phone,  dwh.Dat\_Fax,  dwh.Dat\_HomePage,  GETDATE() AS AUD\_INSERT\_DT,  SYSTEM\_USER AS AUD\_USER\_INSERT,  'DELETE' AS AUD\_ACTION,  dwh.AUD\_ROWID  FROM [DWH].[LK\_SUPPLIERS\_DATOS\_Contacto] dwh  LEFT JOIN [STG].[LK\_SUPPLIERS\_DATOS\_Contacto] s  ON dwh.AUD\_ROWID = s.AUD\_ROWID  WHERE 1 = 1  AND dwh.AUD\_ACTIVE\_FLAG = 1  AND s.Dat\_SupplierId IS NULL  AND dwh.DatC\_Contacto\_Id <> -1;  END; |
| OUTPUT | select \* from [STG].[ LK\_SUPPLIERS\_DATOS\_Contacto] |

### 3.9 Proceso Schema STG 9: PRODUCTS

|  |  |
| --- | --- |
| JobName | STG\_Products.dtsx |
| Input | select \* from [TMP].[PRODUCTS] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_STG\_Products] Script Date: 17/5/2025 11:57:16 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_STG\_Products]  AS  BEGIN  SET NOCOUNT ON;  -- Truncar la tabla STG  TRUNCATE TABLE [STG].[LK\_PRODUCTS];  -- Insertar los datos desde TMP con AUD\_ROW\_ID y acción INSERT  WITH SUPPLIERS AS  (SELECT \* from [DWH].[LK\_SUPPLIERS]  WHERE AUD\_ACTIVE\_FLAG = 1),  CATEGORIES AS  (SELECT \* from [DWH].[LK\_CATEGORIES]  WHERE AUD\_ACTIVE\_FLAG = 1),  STG\_PRODUCTS AS (  SELECT  LTRIM(RTRIM(product\_CD)) AS prod\_product\_CD,  LTRIM(RTRIM(product\_name)) prod\_product\_name,  COALESCE(d.Supp\_Supplier\_Id,-1) prod\_supplier\_id,  COALESCE(c.Category\_Id,-1) prod\_category\_id,  LTRIM(RTRIM(Quantity\_per\_unit)) prod\_Quantity\_per\_unit,  CAST(LTRIM(RTRIM(unit\_price)) AS DECIMAL(10,2)) prod\_unit\_price,  CAST(LTRIM(RTRIM(units\_in\_stock)) AS DECIMAL(10,2)) prod\_units\_in\_stock,  CAST(LTRIM(RTRIM(units\_on\_order)) AS DECIMAL(10,2)) prod\_units\_on\_order,  CAST(LTRIM(RTRIM(reorder\_level)) AS INT) prod\_reorder\_level,  CAST(LTRIM(RTRIM(discontinued)) AS INT) prod\_discontinued  FROM [TMP].[PRODUCTS] s  LEFT JOIN SUPPLIERS d  ON LTRIM(RTRIM(s.supplier\_cd))= d.Supp\_CD  LEFT JOIN CATEGORIES c  ON LTRIM(RTRIM(s.category\_cd))= c.Category\_CD  WHERE ISNULL(LTRIM(RTRIM(product\_CD)), '') <> ''  )  -- Mecanismo de Control de Versiones  INSERT INTO [STG].[LK\_PRODUCTS] (  prod\_product\_CD,  prod\_product\_name,  prod\_supplier\_id,  prod\_category\_id,  prod\_Quantity\_per\_unit,  prod\_unit\_price,  prod\_units\_in\_stock,  prod\_units\_on\_order,  prod\_reorder\_level,  prod\_discontinued,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  bt.prod\_product\_CD,  bt.prod\_product\_name,  bt.prod\_supplier\_id,  bt.prod\_category\_id,  bt.prod\_Quantity\_per\_unit,  bt.prod\_unit\_price,  bt.prod\_units\_in\_stock,  bt.prod\_units\_on\_order,  bt.prod\_reorder\_level,  bt.prod\_discontinued,  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.prod\_product\_CD, ''),  ISNULL(bt.prod\_product\_name, ''),  ISNULL(bt.prod\_supplier\_id, ''),  ISNULL(bt.prod\_category\_id, ''),  ISNULL(bt.prod\_Quantity\_per\_unit, ''),  ISNULL(bt.prod\_unit\_price, 0),  ISNULL(bt.prod\_units\_in\_stock, 0),  ISNULL(bt.prod\_units\_on\_order, 0),  ISNULL(bt.prod\_reorder\_level, 0),  ISNULL(bt.prod\_discontinued, 0)  ) AS AUD\_ROW\_ID  FROM STG\_PRODUCTS bt;  -- Inserto los BORRADOS LÓGICOS  INSERT INTO [STG].[LK\_PRODUCTS] (  prod\_product\_CD,  prod\_product\_name,  prod\_supplier\_id,  prod\_category\_id,  prod\_Quantity\_per\_unit,  prod\_unit\_price,  prod\_units\_in\_stock,  prod\_units\_on\_order,  prod\_reorder\_level,  prod\_discontinued,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  dwh.prod\_product\_CD,  dwh.prod\_product\_name,  dwh.prod\_supplier\_id,  dwh.prod\_category\_id,  dwh.prod\_Quantity\_per\_unit,  dwh.prod\_unit\_price,  dwh.prod\_units\_in\_stock,  dwh.prod\_units\_on\_order,  dwh.prod\_reorder\_level,  dwh.prod\_discontinued,  GETDATE() AS AUD\_INSERT\_DT,  SYSTEM\_USER AS AUD\_USER\_INSERT,  'DELETE' AS AUD\_ACTION,  dwh.AUD\_ROWID  FROM [DWH].[LK\_PRODUCTS] dwh  LEFT JOIN [STG].[LK\_PRODUCTS] s  ON dwh.AUD\_ROWID = s.AUD\_ROWID  WHERE 1 = 1  AND dwh.AUD\_ACTIVE\_FLAG = 1  AND s.prod\_product\_CD IS NULL  AND dwh.prod\_product\_id <> -1;  END; |
| OUTPUT | select \* from [STG].[ LK\_PRODUCTS] |

### 3.10 Proceso Schema STG 10: EMPLOYEE\_TERRITORIES

|  |  |
| --- | --- |
| JobName | STG\_Employees\_Territories.dtsx |
| Input | select \* from [TMP].[EMPLOYEES\_TERRITORIES] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_STG\_Employee\_Territories] Script Date: 17/5/2025 17:37:37 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_STG\_Employee\_Territories]  AS  BEGIN  SET NOCOUNT ON;  -- Truncar la tabla STG  TRUNCATE TABLE [STG].[LK\_EMPLOYEE\_TERRITORIES];  -- Insertar los datos desde TMP con AUD\_ROW\_ID y acción INSERT  WITH  TERRITORIES AS (  Select \* from [DWH].[LK\_TERRITORIES]  WHERE AUD\_ACTIVE\_FLAG = 1  ),  EMPLOYEES AS  (  SELECT \* from [DWH].[LK\_EMPLOYEES]  WHERE AUD\_ACTIVE\_FLAG = 1  ),  STG\_EMPLOYEE\_TERRITORIES AS (  SELECT  COALESCE(e.Emp\_Employee\_id,-1)AS EmpTer\_Employee\_id,  COALESCE(t.Territory\_Id,-1) [EmpTer\_Territory\_id]  FROM [TMP].[EMPLOYEES\_TERRITORIES] s  LEFT JOIN EMPLOYEES e  ON LTRIM(RTRIM(s.Employee\_CD))= e.Emp\_Employee\_CD  LEFT JOIN TERRITORIES t  ON LTRIM(RTRIM(s.Territory\_CD))= t.Territory\_CD  WHERE ISNULL(LTRIM(RTRIM(Employee\_CD)), '') <> ''  )  -- Mecanismo de Control de Versiones  INSERT INTO [STG].[LK\_EMPLOYEE\_TERRITORIES] (  EmpTer\_Employee\_id,  EmpTer\_Territory\_id,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  bt.EmpTer\_Employee\_id,  bt.EmpTer\_Territory\_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.EmpTer\_Employee\_id, ''),  ISNULL(bt.EmpTer\_Territory\_id, '')  ) AS AUD\_ROW\_ID  FROM STG\_EMPLOYEE\_TERRITORIES bt;  -- Inserto los BORRADOS LÓGICOS  INSERT INTO [STG].[LK\_EMPLOYEE\_TERRITORIES] (  EmpTer\_Employee\_id,  EmpTer\_Territory\_id,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  dwh.EmpTer\_Employee\_id,  dwh.EmpTer\_Territory\_id,  GETDATE() AS AUD\_INSERT\_DT,  SYSTEM\_USER AS AUD\_USER\_INSERT,  'DELETE' AS AUD\_ACTION,  dwh.AUD\_ROWID  FROM [DWH].[LK\_EMPLOYEES\_TERRITORIES] dwh  LEFT JOIN [STG].[LK\_EMPLOYEE\_TERRITORIES] s  ON dwh.AUD\_ROWID = s.AUD\_ROWID  WHERE 1 = 1  AND dwh.AUD\_ACTIVE\_FLAG = 1  AND s.EmpTer\_Employee\_id IS NULL  AND dwh.EmpTer\_Employee\_id <> -1;  END; |
| OUTPUT | select \* from [STG].[ LK\_EMPLOYEE\_TERRITORIES] |

### 3.11 Proceso Schema STG 11: CUSTOMERS

|  |  |
| --- | --- |
| JobName | STG\_Customers.dtsx |
| Input | select \* from [TMP].[CUSTOMERS] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_STG\_Customers] Script Date: 18/5/2025 21:31:36 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_STG\_Customers]  AS  BEGIN  SET NOCOUNT ON;  -- Truncar la tabla STG  TRUNCATE TABLE [STG].[ODS\_CUSTOMERS];  -- Insertar los datos desde TMP con AUD\_ROW\_ID y acción INSERT  WITH REGIONS AS  (SELECT \* from [DWH].[VW\_LK\_REL\_REGIONS]),  COUNTRIES AS  (SELECT \* from DWH.VW\_LK\_REL\_COUNTRIES  ),  STG\_CUSTOMERS AS (  SELECT  LTRIM(RTRIM(Customer\_CD)) Cus\_Customer\_CD,  LTRIM(RTRIM(Company\_Name)) Cus\_Company\_Name,  LTRIM(RTRIM(Contact\_Name)) Cus\_Contact\_Name,  LTRIM(RTRIM(Contact\_Title)) Cus\_Contact\_Title,  LTRIM(RTRIM(address)) Cus\_address,  LTRIM(RTRIM(City)) Cus\_City,  LTRIM(RTRIM(Region)) Cus\_Region\_cd,  COALESCE(r.Region\_Id,-1) as Cus\_Region\_id,  LTRIM(RTRIM(Postal\_Code)) Cus\_Postal\_Code,  LTRIM(RTRIM(Country)) Cus\_Country\_cd,  COALESCE(co.Ctry\_Country\_Id,-1) as Cus\_Country\_id,  LTRIM(RTRIM(Phone)) Cus\_Phone,  LTRIM(RTRIM(fax))Cus\_fax  FROM [TMP].[CUSTOMERS] c  LEFT JOIN REGIONS r  ON LTRIM(RTRIM(c.Region)) = r.Region\_CD  LEFT JOIN COUNTRIES co  ON LTRIM(RTRIM(c.Country)) = co.Country\_CD  WHERE ISNULL(LTRIM(RTRIM(Customer\_CD)), '') <> ''  )  -- Mecanismo de Control de Versiones  INSERT INTO [STG].[ODS\_CUSTOMERS] (  [Cus\_Customer\_CD],  [Cus\_Company\_Name],  [Cus\_Contact\_Name],  [Cus\_Contact\_Title],  [Cus\_address],  [Cus\_City],  [Cus\_Region\_cd],  [Cus\_Region\_id],  [Cus\_Postal\_Code],  [Cus\_Country\_cd],  [Cus\_Country\_id],  [Cus\_Phone],  [Cus\_fax],  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  Cus\_Customer\_CD,  Cus\_Company\_Name,  Cus\_Contact\_Name,  Cus\_Contact\_Title,  Cus\_address,  Cus\_City,  Cus\_Region\_cd,  Cus\_Region\_id,  Cus\_Postal\_Code,  Cus\_Country\_cd,  Cus\_Country\_id,  Cus\_Phone,  Cus\_fax,  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.Cus\_Customer\_CD, ''),  ISNULL(bt.Cus\_Company\_Name, ''),  ISNULL(bt.Cus\_Contact\_Name, ''),  ISNULL(bt.Cus\_Contact\_Title, ''),  ISNULL(bt.Cus\_address, ''),  ISNULL(bt.Cus\_City, ''),  ISNULL(bt.Cus\_Region\_cd, ''),  ISNULL(bt.Cus\_Region\_id, ''),  ISNULL(bt.Cus\_Postal\_Code, ''),  ISNULL(bt.Cus\_Country\_cd, ''),  ISNULL(bt.Cus\_Country\_id, ''),  ISNULL(bt.Cus\_Phone, ''),  ISNULL(bt.Cus\_fax, '')  ) AS AUD\_ROW\_ID  FROM STG\_CUSTOMERS bt;  -- Inserto los BORRADOS LÓGICOS  INSERT INTO [STG].[ODS\_Customers] (  Cus\_Customer\_CD,  Cus\_Company\_Name,  Cus\_Contact\_Name,  Cus\_Contact\_Title,  Cus\_address,  Cus\_City,  Cus\_Region\_cd,  Cus\_Region\_id,  Cus\_Postal\_Code,  Cus\_Country\_cd,  Cus\_Country\_id,  Cus\_Phone,  Cus\_fax,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  dwh.Cus\_Customer\_CD,  dwh.Cus\_Company\_Name,  dwh.Cus\_Contact\_Name,  dwh.Cus\_Contact\_Title,  dwh.Cus\_address,  dwh.Cus\_City,  dwh.Cus\_Region\_cd,  dwh.Cus\_Region\_id,  dwh.Cus\_Postal\_Code,  dwh.Cus\_Country\_cd,  dwh.Cus\_Country\_id,  dwh.Cus\_Phone,  dwh.Cus\_fax,  GETDATE() AS AUD\_INSERT\_DT,  SYSTEM\_USER AS AUD\_USER\_INSERT,  'DELETE' AS AUD\_ACTION,  dwh.AUD\_ROWID  FROM [ODS].[CUSTOMERS] dwh  LEFT JOIN [STG].[ODS\_CUSTOMERS] s  ON dwh.AUD\_ROWID = s.AUD\_ROWID  WHERE 1 = 1  AND dwh.AUD\_ACTIVE\_FLAG = 1  AND s.Cus\_Customer\_CD IS NULL  AND dwh.Cus\_Customer\_id<>-1;  END; |
| OUTPUT | select \* from [STG].[ ODS\_CUSTOMERSS] |

### 3.12 Proceso Schema STG 12: ORDERS

|  |  |
| --- | --- |
| JobName | STG\_Orders.dtsx |
| Input | select \* from [TMP].[ORDERS] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_STG\_Customers] Script Date: 18/5/2025 22:46:13 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_STG\_Orders]  AS  BEGIN  SET NOCOUNT ON;  -- Truncar la tabla STG  TRUNCATE TABLE [STG].[ODS\_ORDERS];  -- Insertar los datos desde TMP con AUD\_ROW\_ID y acción INSERT  WITH REGIONS AS  (SELECT \* from [DWH].[VW\_LK\_REL\_REGIONS]),  COUNTRIES AS  (SELECT \* from DWH.VW\_LK\_REL\_COUNTRIES  ),  CUSTOMERS AS  ( SELECT \* FROM ODS.CUSTOMERS  WHERE AUD\_ACTIVE\_FLAG = 1),  EMPLOYEE AS  ( SELECT \* from DWH.LK\_EMPLOYEES  WHERE AUD\_ACTIVE\_FLAG = 1),  STG\_ORDERS AS (  SELECT  LTRIM(RTRIM(Order\_CD)) Ord\_Order\_CD,  LTRIM(RTRIM(Customer\_cd)) Ord\_Customer\_cd,  COALESCE(cu.Cus\_Customer\_id,-1) Ord\_Customer\_id,  LTRIM(RTRIM(Employee\_cd)) Ord\_Employee\_cd,  LTRIM(RTRIM(e.Emp\_Employee\_id)) Ord\_Employee\_id,  COALESCE(TRY\_CAST(  CASE  WHEN LTRIM(RTRIM(Order\_Date)) IS NULL OR LTRIM(RTRIM(Order\_Date)) = ''  THEN '1900-01-01'  ELSE LTRIM(RTRIM(Order\_Date))  END  AS DATE), CAST('1900-01-01' AS DATE)) AS Ord\_Order\_Date,  COALESCE(TRY\_CAST(  CASE  WHEN LTRIM(RTRIM(Required\_Date)) IS NULL OR LTRIM(RTRIM(Required\_Date)) = ''  THEN '1900-01-01'  ELSE LTRIM(RTRIM(Required\_Date))  END  AS DATE), CAST('1900-01-01' AS DATE)) AS Ord\_Required\_Date,  COALESCE(TRY\_CAST(  CASE  WHEN LTRIM(RTRIM(Shipped\_Date)) IS NULL OR LTRIM(RTRIM(Shipped\_Date)) = ''  THEN '1900-01-01'  ELSE LTRIM(RTRIM(Shipped\_Date))  END  AS DATE), CAST('1900-01-01' AS DATE)) AS Ord\_Shipped\_Date,  LTRIM(RTRIM(Ship\_Via)) Ord\_Ship\_Via,  LTRIM(RTRIM(Freight)) Ord\_Freight,  LTRIM(RTRIM(Ship\_name)) as Ord\_Ship\_name,  LTRIM(RTRIM(Ship\_Address)) Ord\_Ship\_Address,  LTRIM(RTRIM(Ship\_City))Ord\_Ship\_City,  LTRIM(RTRIM(Ship\_Region))Ord\_Ship\_Region\_cd,  COALESCE(r.Region\_Id ,-1) as Ord\_Ship\_Region\_Id,  LTRIM(RTRIM(Ship\_PostalCode))Ord\_Ship\_PostalCode,  LTRIM(RTRIM(Ship\_Country))Ord\_Ship\_Country\_cd,  COALESCE (coshp.Ctry\_Country\_id, -1) Ord\_Ship\_Country\_id  FROM [TMP].[ORDERS] c  LEFT JOIN REGIONS r  ON LTRIM(RTRIM(c.Ship\_Region)) = r.Region\_CD  LEFT JOIN CUSTOMERS cu  ON LTRIM(RTRIM(c.Customer\_cd)) = cu.Cus\_Customer\_cd  LEFT JOIN EMPLOYEE e  ON LTRIM(RTRIM(c.Employee\_cd)) = e.Emp\_Employee\_Cd  LEFT JOIN COUNTRIES coshp  ON LTRIM(RTRIM(c.Ship\_Country)) = coshp.Country\_CD  WHERE ISNULL(LTRIM(RTRIM(Order\_CD)), '') <> ''  )  -- Mecanismo de Control de Versiones  INSERT INTO [STG].[ODS\_ORDERS] (  Ord\_Order\_CD,  Ord\_Customer\_cd,  Ord\_Customer\_id,  Ord\_Employee\_cd,  Ord\_Employee\_id,  Ord\_Order\_Date,  Ord\_Required\_Date,  Ord\_Shipped\_Date,  Ord\_Ship\_Via,  Ord\_Freight,  Ord\_Ship\_name,  Ord\_Ship\_Address,  Ord\_Ship\_City,  Ord\_Ship\_Region\_cd,  Ord\_Ship\_Region\_Id,  Ord\_Ship\_PostalCode,  Ord\_Ship\_Country\_cd,  Ord\_Ship\_Country\_id,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  Ord\_Order\_CD,  Ord\_Customer\_cd,  Ord\_Customer\_id,  Ord\_Employee\_cd,  Ord\_Employee\_id,  Ord\_Order\_Date,  Ord\_Required\_Date,  Ord\_Shipped\_Date,  Ord\_Ship\_Via,  Ord\_Freight,  Ord\_Ship\_name,  Ord\_Ship\_Address,  Ord\_Ship\_City,  Ord\_Ship\_Region\_cd,  Ord\_Ship\_Region\_Id,  Ord\_Ship\_PostalCode,  Ord\_Ship\_Country\_cd,  Ord\_Ship\_Country\_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.Ord\_Order\_CD, ''),  ISNULL(bt.Ord\_Customer\_cd, ''),  ISNULL(bt.Ord\_Customer\_id, ''),  ISNULL(bt.Ord\_Employee\_id, ''),  ISNULL(bt.Ord\_Order\_Date, ''),  ISNULL(bt.Ord\_Required\_Date, ''),  ISNULL(bt.Ord\_Shipped\_Date, ''),  ISNULL(bt.Ord\_Ship\_Via, ''),  ISNULL(bt.Ord\_Freight, ''),  ISNULL(bt.Ord\_Ship\_name, ''),  ISNULL(bt.Ord\_Ship\_Address, ''),  ISNULL(bt.Ord\_Ship\_City, ''),  ISNULL(bt.Ord\_Ship\_Region\_cd, ''),  ISNULL(bt.Ord\_Ship\_Region\_Id, ''),  ISNULL(bt.Ord\_Ship\_PostalCode, ''),  ISNULL(bt.Ord\_Ship\_Country\_cd, ''),  ISNULL(bt.Ord\_Ship\_Country\_id, '')  ) AS AUD\_ROW\_ID  FROM STG\_ORDERS bt;  -- Inserto los BORRADOS LÓGICOS  INSERT INTO [STG].[ODS\_ORDERS] (  Ord\_Order\_CD,  Ord\_Customer\_cd,  Ord\_Customer\_id,  Ord\_Employee\_cd,  Ord\_Employee\_id,  Ord\_Order\_Date,  Ord\_Required\_Date,  Ord\_Shipped\_Date,  Ord\_Ship\_Via,  Ord\_Freight,  Ord\_Ship\_name,  Ord\_Ship\_Address,  Ord\_Ship\_City,  Ord\_Ship\_Region\_cd,  Ord\_Ship\_Region\_Id,  Ord\_Ship\_PostalCode,  Ord\_Ship\_Country\_cd,  Ord\_Ship\_Country\_id,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  dwh.Ord\_Order\_CD,  dwh.Ord\_Customer\_cd,  dwh.Ord\_Customer\_id,  dwh.Ord\_Employee\_cd,  dwh.Ord\_Employee\_id,  dwh.Ord\_Order\_Date,  dwh.Ord\_Required\_Date,  dwh.Ord\_Shipped\_Date,  dwh.Ord\_Ship\_Via,  dwh.Ord\_Freight,  dwh.Ord\_Ship\_name,  dwh.Ord\_Ship\_Address,  dwh.Ord\_Ship\_City,  dwh.Ord\_Ship\_Region\_cd,  dwh.Ord\_Ship\_Region\_Id,  dwh.Ord\_Ship\_PostalCode,  dwh.Ord\_Ship\_Country\_cd,  dwh.Ord\_Ship\_Country\_id,  GETDATE() AS AUD\_INSERT\_DT,  SYSTEM\_USER AS AUD\_USER\_INSERT,  'DELETE' AS AUD\_ACTION,  dwh.AUD\_ROWID  FROM [ODS].[ORDERS] dwh  LEFT JOIN [STG].[ODS\_ORDERS] s  ON dwh.AUD\_ROWID = s.AUD\_ROWID  WHERE 1 = 1  AND dwh.AUD\_ACTIVE\_FLAG = 1  AND s.Ord\_Order\_CD IS NULL  AND dwh.Ord\_Order\_id<>-1;  END; |
| OUTPUT | select \* from [STG].[ ODS\_ORDERS] |

### 3.13 Proceso Schema STG 13: ORDERS DETAILS

|  |  |
| --- | --- |
| JobName | STG\_Order\_Details.dtsx |
| Input | select \* from [TMP].[ORDERS\_DETAILS] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_STG\_Orders] Script Date: 18/5/2025 23:48:06 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_STG\_Orders\_Details]  AS  BEGIN  SET NOCOUNT ON;  -- Truncar la tabla STG  TRUNCATE TABLE [STG].[ODS\_ORDERS\_DETAILS];  -- Insertar los datos desde TMP con AUD\_ROW\_ID y acción INSERT  WITH PRODUCTS AS  ( SELECT \* FROM DWH.LK\_PRODUCTS  WHERE AUD\_ACTIVE\_FLAG = 1),  ORDERS AS  ( SELECT \* from ODS.ORDERS  WHERE AUD\_ACTIVE\_FLAG = 1),  STG\_ORDERS\_DETAILS AS (  SELECT  COALESCE(ord\_order\_id,-1) [OrdDet\_Order\_id],  LTRIM(RTRIM(Product\_cd)) [OrdDet\_Product\_cd],  COALESCE(prod\_product\_id,-1) [OrdDet\_Product\_id],  TRY\_CAST(LTRIM(RTRIM(Unit\_Price)) as DECIMAL(10,2)) [OrdDet\_Unit\_Price],  TRY\_CAST(LTRIM(RTRIM(Quantity)) as DECIMAL(10,2)) [OrdDet\_Quantity],  TRY\_CAST(LTRIM(RTRIM(Discount)) as DECIMAL(10,2)) [OrdDet\_Discount]  FROM [TMP].[ORDERS\_DETAILS] c  LEFT JOIN PRODUCTS r  ON LTRIM(RTRIM(c.Product\_cd)) = r.prod\_product\_CD  LEFT JOIN ORDERS o  ON LTRIM(RTRIM(c.Order\_cd)) = o.Ord\_Order\_cd  WHERE ISNULL(LTRIM(RTRIM(Order\_CD)), '') <> ''  )  -- Mecanismo de Control de Versiones  INSERT INTO [STG].[ODS\_ORDERS\_DETAILS] (  [OrdDet\_Order\_id],  [OrdDet\_Product\_cd],  [OrdDet\_Product\_id],  [OrdDet\_Unit\_Price],  [OrdDet\_Quantity],  [OrdDet\_Discount],  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  [OrdDet\_Order\_id],  [OrdDet\_Product\_cd],  [OrdDet\_Product\_id],  [OrdDet\_Unit\_Price],  [OrdDet\_Quantity],  [OrdDet\_Discount],  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.OrdDet\_Order\_id, ''),  ISNULL(bt.OrdDet\_Product\_cd, ''),  ISNULL(bt.OrdDet\_Product\_id, ''),  ISNULL(bt.OrdDet\_Unit\_Price, 0),  ISNULL(bt.OrdDet\_Quantity, 0),  ISNULL(bt.OrdDet\_Discount, 0)  ) AS AUD\_ROW\_ID  FROM STG\_ORDERS\_DETAILS bt;  -- Inserto los BORRADOS LÓGICOS  INSERT INTO [STG].[ODS\_ORDERS\_DETAILS] (  [OrdDet\_Order\_id],  [OrdDet\_Product\_cd],  [OrdDet\_Product\_id],  [OrdDet\_Unit\_Price],  [OrdDet\_Quantity],  [OrdDet\_Discount],  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTION,  AUD\_ROWID  )  SELECT  dwh.OrdDet\_Order\_id,  dwh.OrdDet\_Product\_cd,  dwh.OrdDet\_Product\_id,  dwh.OrdDet\_Unit\_Price,  dwh.OrdDet\_Quantity,  dwh.OrdDet\_Discount,  GETDATE() AS AUD\_INSERT\_DT,  SYSTEM\_USER AS AUD\_USER\_INSERT,  'DELETE' AS AUD\_ACTION,  dwh.AUD\_ROWID  FROM [ODS].[ORDERS\_DETAILS] dwh  LEFT JOIN [STG].[ODS\_ORDERS\_DETAILS] s  ON dwh.AUD\_ROWID = s.AUD\_ROWID  WHERE 1 = 1  AND dwh.AUD\_ACTIVE\_FLAG = 1  AND s.OrdDet\_Order\_id IS NULL  AND dwh.OrdDet\_Order\_Detail\_id<>-1;  END; |
| OUTPUT | select \* from [STG].[ODS\_ORDERS\_DETAILS]; |

## Procesos de ingesta de Schema DWH (Dimensiones)

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 | ALTER PROCEDURE [dbo].[SP\_DWH\_Categories]  AS  BEGIN  SET NOCOUNT ON;  -- Inicializar Dimensión con valor NA  -- Verificar si el valor NA (-1) ya existe en la tabla LK\_REGIONS  IF NOT EXISTS (SELECT 1 FROM [DWH].[LK\_CATEGORIES] WHERE Category\_Id = -1)  BEGIN  -- Habilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_CATEGORIES] ON;  -- Insertar el valor NA (-1)  INSERT INTO [DWH].[LK\_CATEGORIES] (  Category\_Id, -- Identity, por eso ahora se puede insertar explícitamente  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID,  Category\_CD,  Category\_Name,  Category\_Description  )  VALUES (  -1,  GETDATE(),  SYSTEM\_USER,  1,  0, -- o algún valor identificador especial si tu lógica lo requiere  '-1',  'Not Applicable',  'Not Applicable'  );  -- Deshabilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_CATEGORIES] OFF;  END;  -- 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

Se ha considerado hacer una tabla de relaciones, como en Countries; Para mapear regiones que vienen desde Employees.

|  |  |
| --- | --- |
| 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  BEGIN  SET NOCOUNT ON;  -- Inicializar Dimensión con valor NA  -- Verificar si el valor NA (-1) ya existe en la tabla LK\_REGIONS  IF NOT EXISTS (SELECT 1 FROM [DWH].[LK\_REGIONS] WHERE Region\_Id = -1)  BEGIN  -- Habilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_REGIONS] ON;  -- Insertar el valor NA (-1)  INSERT INTO [DWH].[LK\_REGIONS] (  Region\_Id, -- Identity, por eso ahora se puede insertar explícitamente  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID,  Region\_CD,  Region\_Description  )  VALUES (  -1,  GETDATE(),  SYSTEM\_USER,  1,  0, -- o algún valor identificador especial si tu lógica lo requiere  '-1',  'Not Applicable'  );  -- Deshabilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_REGIONS] OFF;  END;  -- 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] |
| Registros de Relacion |  |
| View FInal | CREATE VIEW [DWH].VW\_LK\_REL\_REGIONS AS  SELECT  c.Region\_Id,  map.MAP\_External\_RegionName Region\_Cd,  c.Region\_Description  FROM  [DWH].[REL\_EMPLOYEEREGION\_REGIONS] map  LEFT JOIN DWH.LK\_REGIONS c  ON UPPER(LTRIM(RTRIM(c.Region\_Description))) = UPPER(LTRIM(RTRIM(map.MAP\_Region\_NAme)))  WHERE  map.AUD\_ACTIVE\_FLAG = 1  AND c.AUD\_ACTIVE\_FLAG = 1  UNION ALL  SELECT  c.Region\_Id,  c.Region\_Cd,  c.Region\_Description  FROM  DWH.LK\_REGIONS c  WHERE c.AUD\_ACTIVE\_FLAG = 1  GO |
| OUTPUT VIEW |  |

### 4.3 Proceso Schema DWH 3: SHIPPERS

|  |  |
| --- | --- |
| JobName | DWH\_Shippers.dtsx |
| Input | select \* from [STG].[SHIPPERS] |
| Control Flow |  |
| Proceso | CREATE PROCEDURE [dbo].[SP\_DWH\_Shippers]  AS  -- Inicializar Dimensión con valor NA  -- Verificar si el valor NA (-1) ya existe en la tabla LK\_REGIONS  IF NOT EXISTS (SELECT 1 FROM [DWH].[LK\_SHIPPERS] WHERE Ship\_Shipper\_id = -1)  BEGIN  -- Habilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_SHIPPERS] ON;  -- Insertar el valor NA (-1)  INSERT INTO [DWH].[LK\_SHIPPERS] (  Ship\_Shipper\_id, -- Identity, por eso ahora se puede insertar explícitamente  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID,  Ship\_Shipper\_CD,  Ship\_Company\_Name  )  VALUES (  -1,  GETDATE(),  SYSTEM\_USER,  1,  0, -- o algún valor identificador especial si tu lógica lo requiere  '-1',  'Not Applicable'  );  -- Deshabilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_SHIPPERS] OFF;  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;  -- Inicializar Dimensión con valor NA  -- Verificar si el valor NA (-1) ya existe en la tabla LK\_REGIONS  IF NOT EXISTS (SELECT 1 FROM [DWH].[LK\_TERRITORIES] WHERE Territory\_Id = -1)  BEGIN  -- Habilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_TERRITORIES] ON;  -- Insertar el valor NA (-1)  INSERT INTO [DWH].[LK\_TERRITORIES] (  Territory\_Id, -- Identity, por eso ahora se puede insertar explícitamente  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID,  Territory\_CD,  Territory\_Description,  Territory\_Region\_Id  )  VALUES (  -1,  GETDATE(),  SYSTEM\_USER,  1,  0, -- o algún valor identificador especial si tu lógica lo requiere  '-1',  'Not Applicable',  -1  );  -- Deshabilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_TERRITORIES] OFF;  END;  -- 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] |

### 4.5 Proceso Schema DWH 5: COUNTRIES

Particularmente, en ésta dimensión, se hace un tratamiento diferente; Puesto que es necesario una tabla de relación, para manejar Distintos nombres del mismo país

|  |  |
| --- | --- |
| JobName | DWH\_Countries.dtsx |
| Input | select \* from [STG].[COUNTRIES] |
| Control Flow |  |
| Proceso | ALTER PROCEDURE [dbo].[SP\_DWH\_Countries]  AS  BEGIN  SET NOCOUNT ON;  -- Inicializar Dimensión con valor NA  -- Verificar si el valor NA (-1) ya existe en la tabla LK\_REGIONS  IF NOT EXISTS (SELECT 1 FROM [DWH].[LK\_COUNTRIES] WHERE Ctry\_Country\_Id = -1)  BEGIN  -- Habilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_COUNTRIES] ON;  -- Insertar el valor NA (-1)  INSERT INTO [DWH].[LK\_COUNTRIES] (  Ctry\_Country\_Id, -- Identity, por eso ahora se puede insertar explícitamente  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID,  Country\_CD,  Ctry\_Name  )  VALUES (  -1,  GETDATE(),  SYSTEM\_USER,  1,  0, -- o algún valor identificador especial si tu lógica lo requiere  '-1',  'Not Applicable'  );  -- Deshabilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_COUNTRIES] OFF;  END;  -- Realizando el MERGE en la tabla de destino con la tabla fuente  MERGE INTO [DWH].[LK\_COUNTRIES] AS target  USING [STG].[LK\_COUNTRIES] 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,  Country\_CD,  Ctry\_Name  )  VALUES (  GETDATE(), -- AUD\_INSERT\_DT  SYSTEM\_USER, -- AUD\_USER\_INSERT  1, -- AUD\_ACTIVE\_FLAG (TRUE)  source.AUD\_ROWID,  source.Country\_CD,  source.Country\_Name  )  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\_COUNTRIES] |
| TABLA DE RELACION | select \* from DWH.REL\_SUPPLIER\_COUNTRY |
| VIEW | CREATE VIEW DWH.VW\_LK\_REL\_COUNTRIES as  SELECT  c.Ctry\_Country\_Id,  c.Ctry\_Name,  map.MAP\_External\_Name Country\_CD  FROM  DWH.REL\_SUPPLIER\_COUNTRY map  LEFT JOIN DWH.LK\_COUNTRIES c  ON UPPER(LTRIM(RTRIM(c.Ctry\_Name))) = UPPER(LTRIM(RTRIM(map.MAP\_Ctry\_Country\_Name)))  WHERE  map.AUD\_ACTIVE\_FLAG = 1  AND c.AUD\_ACTIVE\_FLAG = 1  UNION ALL  SELECT  c.Ctry\_Country\_Id,  c.Ctry\_Name,  c.Ctry\_Name Country\_CD  FROM  DWH.LK\_COUNTRIES c  WHERE c.AUD\_ACTIVE\_FLAG = 1 |
| OUTPUT VIEW | De ésta forma complementa TABLA con RELACIONES |

### 4.6 Proceso Schema DWH 6: EMPLOYEES

|  |  |
| --- | --- |
| JobName | DWH\_Employees.dtsx |
| Input | select \* from [STG].[EMPLOYEES] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_DWH\_Employees] Script Date: 17/5/2025 15:31:06 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_DWH\_Employees]  AS  BEGIN  SET NOCOUNT ON;  -- Inicializar Dimensión con valor NA  -- Verificar si el valor NA (-1) ya existe en la tabla LK\_EMPLOYEES  IF NOT EXISTS (SELECT 1 FROM [DWH].[LK\_EMPLOYEES] WHERE Emp\_Employee\_id = -1)  BEGIN  -- Habilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_EMPLOYEES] ON;  -- Insertar el valor NA (-1)  INSERT INTO [DWH].[LK\_EMPLOYEES] (  Emp\_Employee\_id, -- Identity, por eso ahora se puede insertar explícitamente  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID,  Emp\_Employee\_CD,  Emp\_Last\_name,  Emp\_First\_name,  Emp\_report\_to\_id  )  VALUES (  -1,  GETDATE(),  SYSTEM\_USER,  1,  0, -- o algún valor identificador especial si tu lógica lo requiere  '-1',  'Not Applicable',  'Not Applicable',  -1  );  -- Deshabilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_EMPLOYEES] OFF;  END;  -- Realizando el MERGE en la tabla de destino con la tabla fuente  MERGE INTO [DWH].[LK\_EMPLOYEES] AS target  USING [STG].[LK\_EMPLOYEES] 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,  Emp\_Employee\_CD,  Emp\_Last\_name,  Emp\_First\_name,  Emp\_title,  Emp\_title\_of\_courtesy,  Emp\_birth\_date,  Emp\_hire\_date,  Emp\_address,  Emp\_city,  Emp\_Region\_id,  Emp\_postal\_Code,  Emp\_home\_phone,  Emp\_extension,  Emp\_photo,  Emp\_notes,  Emp\_report\_to\_id,  Emp\_photo\_path  )  VALUES (  GETDATE(), -- AUD\_INSERT\_DT  SYSTEM\_USER, -- AUD\_USER\_INSERT  1, -- AUD\_ACTIVE\_FLAG (TRUE)  source.AUD\_ROWID,  source.Emp\_Employee\_CD,  source.Emp\_Last\_name,  source.Emp\_First\_name,  source.Emp\_title,  source.Emp\_title\_of\_courtesy,  source.Emp\_birth\_date,  source.Emp\_hire\_date,  source.Emp\_address,  source.Emp\_city,  source.Emp\_Region\_id,  source.Emp\_postal\_Code,  source.Emp\_home\_phone,  source.Emp\_extension,  source.Emp\_photo,  source.Emp\_notes,  source.Emp\_report\_to\_id,  source.Emp\_photo\_path  )  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;  ;  -- RECALCULO DE JERARQUIAS  -- Tabla temporal para almacenar los IDs de empleados que fueron modificados  IF OBJECT\_ID('tempdb..#Empleados\_Modificados') IS NOT NULL DROP TABLE #Empleados\_Modificados;    Select  delta.\*,  r.Emp\_Employee\_id AS new\_mngr  INTO #Empleados\_Modificados  FROM  (  Select t.Emp\_Employee\_id,  stg.Emp\_report\_to\_cd,  t.Emp\_report\_to\_id  FROM  (  SELECT tgt.Emp\_Employee\_id,  Emp\_Employee\_CD,  tgt.Emp\_report\_to\_id  FROM [DWH].[LK\_EMPLOYEES] tgt  WHERE tgt.AUD\_ACTIVE\_FLAG = 1  ) t  INNER JOIN [STG].[LK\_EMPLOYEES] stg  ON t.Emp\_Employee\_CD = stg.Emp\_Employee\_CD  WHERE AUD\_ACTION = 'INSERT'  ) delta  INNER JOIN [DWH].[LK\_EMPLOYEES] r  ON delta.Emp\_report\_to\_cd = r.Emp\_Employee\_CD  WHERE r.AUD\_ACTIVE\_FLAG = 1  AND COALESCE(r.Emp\_Employee\_id,-1) <> COALESCE(delta.Emp\_report\_to\_id,-1);  -- Hacer el UPDATE solo sobre los que cambiaron  UPDATE tgt  SET tgt.Emp\_report\_to\_id = COALESCE(mgr.new\_mngr, -1)  FROM [DWH].[LK\_EMPLOYEES] tgt  INNER JOIN #Empleados\_Modificados mgr  ON tgt.Emp\_Employee\_id = mgr.Emp\_Employee\_id  WHERE AUD\_ACTIVE\_FLAG = 1;  -- Recalcular el AUD\_ROWID solo para los que fueron modificados  UPDATE e  SET AUD\_ROWID = CHECKSUM(  ISNULL(Emp\_Employee\_CD, ''),  ISNULL(Emp\_Employee\_CD, ''),  ISNULL(Emp\_Last\_name, ''),  ISNULL(Emp\_First\_name, ''),  ISNULL(Emp\_title, ''),  ISNULL(Emp\_title\_of\_courtesy, ''),  ISNULL(Emp\_birth\_date, ''),  ISNULL(Emp\_hire\_date, ''),  ISNULL(Emp\_address, ''),  ISNULL(Emp\_city, ''),  ISNULL(Emp\_Region\_id, ''),  ISNULL(Emp\_postal\_Code, ''),  ISNULL(Emp\_home\_phone, ''),  ISNULL(Emp\_extension, ''),  ISNULL(Emp\_photo, ''),  ISNULL(Emp\_notes, ''),  ISNULL(Emp\_report\_to\_id, ''),  ISNULL(Emp\_photo\_path, '')  )  FROM [DWH].[LK\_EMPLOYEES] e  WHERE e.Emp\_Employee\_id IN (SELECT Emp\_Employee\_id FROM #Empleados\_Modificados)  AND e.AUD\_ACTIVE\_FLAG = 1;  END; |
| OUTPUT | select \* from [DWH].[LK\_EMPLOYEES] |

### 4.7 Proceso Schema DWH 7: SUPPLIERS

|  |  |
| --- | --- |
| JobName | DWH\_Suppliers.dtsx |
| Input | select \* from [STG].[LK\_SUPPLIERS] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_DWH\_Suppliers] Script Date: 13/5/2025 23:20:01 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_DWH\_Suppliers]  AS  BEGIN  SET NOCOUNT ON;  -- Inicializar Dimensión con valor NA  -- Verificar si el valor NA (-1) ya existe en la tabla LK\_REGIONS  IF NOT EXISTS (SELECT 1 FROM [DWH].[LK\_SUPPLIERS] WHERE Supp\_Supplier\_Id = -1)  BEGIN  -- Habilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_SUPPLIERS] ON;  -- Insertar el valor NA (-1)  INSERT INTO [DWH].[LK\_SUPPLIERS] (  Supp\_Supplier\_Id, -- Identity, por eso ahora se puede insertar explícitamente  [Supp\_Id\_Region],  [Supp\_ID\_DatoContacto],  [Supp\_Id\_Country],  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID,  Supp\_CD,  Supp\_companyName  )  VALUES (  -1,  -1,  -1,  -1,  GETDATE(),  SYSTEM\_USER,  1,  0, -- o algún valor identificador especial si tu lógica lo requiere  '-1',  'Not Applicable'  );  -- Deshabilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_SUPPLIERS] OFF;  END;  -- Realizando el MERGE en la tabla de destino con la tabla fuente  MERGE INTO [DWH].[LK\_SUPPLIERS] AS target  USING [STG].[LK\_SUPPLIERS] 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,  Supp\_CD,  Supp\_companyName,  Supp\_contactName,  Supp\_contactTitle,  Supp\_address,  Supp\_City,  Supp\_Id\_Region,  Supp\_PostalCode,  Supp\_Id\_Country,  Supp\_ID\_DatoContacto  )  VALUES (  GETDATE(), -- AUD\_INSERT\_DT  SYSTEM\_USER, -- AUD\_USER\_INSERT  1, -- AUD\_ACTIVE\_FLAG (TRUE)  source.AUD\_ROWID,  source.Supp\_CD,  source.Supp\_companyName,  source.Supp\_contactName,  source.Supp\_contactTitle,  source.Supp\_address,  source.Supp\_City,  source.Supp\_Id\_Region,  source.Supp\_PostalCode,  source.Supp\_Id\_Country,  source.Supp\_ID\_DatoContacto  )  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\_SUPPLIERS] |

### 4.8 Proceso Schema DWH 8: SUPPLIERS DATOS CONTACTO

|  |  |
| --- | --- |
| JobName | DWH\_Suppliers\_Datos\_Contacto.dtsx |
| Input | select \* from [STG].[LK\_SUPPLIERS] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_DWH\_Suppliers\_Datos\_Contacto] Script Date: 16/5/2025 19:00:25 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_DWH\_Suppliers\_Datos\_Contacto]  AS  BEGIN  SET NOCOUNT ON;  -- Inicializar Dimensión con valor NA  -- Verificar si el valor NA (-1) ya existe en la tabla LK\_REGIONS  IF NOT EXISTS (SELECT 1 FROM [DWH].LK\_SUPPLIERS\_DATOS\_Contacto WHERE DatC\_Contacto\_Id = -1)  BEGIN  -- Habilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].LK\_SUPPLIERS\_DATOS\_Contacto ON;  -- Insertar el valor NA (-1)  INSERT INTO [DWH].LK\_SUPPLIERS\_DATOS\_Contacto (  DatC\_Contacto\_Id, -- Identity, por eso ahora se puede insertar explícitamente  Dat\_SupplierId,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID  )  VALUES (  -1,  -1,  GETDATE(),  SYSTEM\_USER,  1,  0 -- o algún valor identificador especial si tu lógica lo requiere  );  -- Deshabilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].LK\_SUPPLIERS\_DATOS\_Contacto OFF;  END;  MERGE INTO [DWH].LK\_SUPPLIERS\_DATOS\_Contacto AS target  USING [STG].LK\_SUPPLIERS\_DATOS\_Contacto 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,  Dat\_SupplierId,  Dat\_Phone,  Dat\_Fax,  Dat\_HomePage  )  VALUES (  GETDATE(), -- AUD\_INSERT\_DT  SYSTEM\_USER, -- AUD\_USER\_INSERT  1, -- AUD\_ACTIVE\_FLAG (TRUE)  source.AUD\_ROWID,  source.Dat\_SupplierId,  source.Dat\_Phone,  source.Dat\_Fax,  source.Dat\_HomePage  )  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;  ---------------------------------------------------------------------------------  --ACTUALIZACION DE SUPPLIERS  UPDATE SP  SET  SP.Supp\_ID\_DatoContacto = DC.DatC\_Contacto\_Id,  SP.AUD\_ROWID = CHECKSUM(  ISNULL(SP.Supp\_CD, ''),  ISNULL(SP.Supp\_companyName, ''),  ISNULL(SP.Supp\_contactName, ''),  ISNULL(SP.Supp\_contactTitle, ''),  ISNULL(SP.Supp\_address, ''),  ISNULL(SP.Supp\_City, ''),  ISNULL(SP.Supp\_Id\_Region, ''),  ISNULL(SP.Supp\_PostalCode, ''),  ISNULL(SP.Supp\_Id\_Country, ''),  ISNULL(DC.DatC\_Contacto\_Id, '')  )  FROM [DWH].[LK\_Suppliers] SP  INNER JOIN [DWH].[LK\_SUPPLIERS\_DATOS\_Contacto] DC  ON SP.Supp\_Supplier\_Id = DC.Dat\_SupplierId  WHERE ISNULL(SP.Supp\_ID\_DatoContacto, -1) <> DC.DatC\_Contacto\_Id;  END;  , |
| OUTPUT | select \* from [DWH].[ LK\_SUPPLIERS\_DATOS\_Contacto] |

### 4.9 Proceso Schema DWH 9: PRODUCTS

|  |  |
| --- | --- |
| JobName | DWH\_Products.dtsx |
| Input | select \* from [STG].[LK\_PRODUCTS] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_DWH\_Products] Script Date: 17/5/2025 12:37:35 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_DWH\_Products]  AS  BEGIN  SET NOCOUNT ON;  -- Inicializar Dimensión con valor NA  -- Verificar si el valor NA (-1) ya existe en la tabla LK\_Products  IF NOT EXISTS (SELECT 1 FROM [DWH].[LK\_PRODUCTS] WHERE prod\_product\_id = -1)  BEGIN  -- Habilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_PRODUCTS] ON;  -- Insertar el valor NA (-1)  INSERT INTO [DWH].[LK\_PRODUCTS] (  prod\_product\_id, -- Identity, por eso ahora se puede insertar explícitamente  prod\_product\_CD,  prod\_product\_name,  prod\_supplier\_id,  prod\_category\_id,  prod\_discontinued,  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID  )  VALUES (  -1,  -1,  'Not Applicable',  -1,  -1,  0,  GETDATE(),  SYSTEM\_USER,  1,  0  );  -- Deshabilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_PRODUCTS] OFF;  END;  -- Realizando el MERGE en la tabla de destino con la tabla fuente  MERGE INTO [DWH].[LK\_PRODUCTS] AS target  USING [STG].[LK\_PRODUCTS] 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,  prod\_product\_CD,  prod\_product\_name,  prod\_supplier\_id,  prod\_category\_id,  prod\_Quantity\_per\_unit,  prod\_unit\_price,  prod\_units\_in\_stock,  prod\_units\_on\_order,  prod\_reorder\_level,  prod\_discontinued  )  VALUES (  GETDATE(), -- AUD\_INSERT\_DT  SYSTEM\_USER, -- AUD\_USER\_INSERT  1, -- AUD\_ACTIVE\_FLAG (TRUE)  source.AUD\_ROWID,  source.prod\_product\_CD,  source.prod\_product\_name,  source.prod\_supplier\_id,  source.prod\_category\_id,  source.prod\_Quantity\_per\_unit,  source.prod\_unit\_price,  source.prod\_units\_in\_stock,  source.prod\_units\_on\_order,  source.prod\_reorder\_level,  source.prod\_discontinued )  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\_PRODUCTS] |

### 4.10 Proceso Schema DWH 10: EMPLOYEES TERRITORIES

|  |  |
| --- | --- |
| JobName | DWH\_Employee\_Territories.dtsx |
| Input | select \* from [STG].[LK\_EMPLOYEE\_TERRITORIES] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_DWH\_Regions] Script Date: 17/5/2025 17:38:47 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE PROCEDURE [dbo].[SP\_DWH\_Employee\_Territories]  AS  BEGIN  SET NOCOUNT ON;  -- Inicializar Dimensión con valor NA  -- Verificar si el valor NA (-1) ya existe en la tabla LK\_REGIONS  IF NOT EXISTS (SELECT 1 FROM [DWH].[LK\_EMPLOYEES\_TERRITORIES] WHERE EmpTer\_Employee\_id = -1)  BEGIN  -- Habilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_EMPLOYEES\_TERRITORIES] ON;  -- Insertar el valor NA (-1)  INSERT INTO [DWH].[LK\_EMPLOYEES\_TERRITORIES] (  EmpTer\_EmployeeTerritory\_id, -- Identity, por eso ahora se puede insertar explícitamente  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID,  EmpTer\_Employee\_id,  EmpTer\_Territory\_Id  )  VALUES (  -1,  GETDATE(),  SYSTEM\_USER,  1,  0, -- o algún valor identificador especial si tu lógica lo requiere  -1,  -1  );  -- Deshabilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_EMPLOYEES\_TERRITORIES] OFF;  END;  -- Realizando el MERGE en la tabla de destino con la tabla fuente  MERGE INTO [DWH].[LK\_EMPLOYEES\_TERRITORIES] AS target  USING [STG].[LK\_EMPLOYEE\_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,  EmpTer\_Employee\_id,  EmpTer\_Territory\_Id  )  VALUES (  GETDATE(), -- AUD\_INSERT\_DT  SYSTEM\_USER, -- AUD\_USER\_INSERT  1, -- AUD\_ACTIVE\_FLAG (TRUE)  source.AUD\_ROWID,  source.EmpTer\_Employee\_id,  source.EmpTer\_Territory\_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\_EMPLOYEE\_TERRITORIES] |

### 4.11 Proceso Schema DWH 11: LK\_TIEMPO

|  |  |
| --- | --- |
| JobName | - |
| Input | - Tabla de Origen MANUAL |
| Proceso | CREATE OR ALTER PROCEDURE DWH.SP\_CREA\_DIM\_TIEMPO  AS  BEGIN  SET NOCOUNT ON;  -- Crear la tabla si no existe  IF OBJECT\_ID('DWH.LK\_TIEMPO', 'U') IS NULL  BEGIN  CREATE TABLE DWH.LK\_TIEMPO (  Fecha\_ID INT PRIMARY KEY, -- Clave surrogate: YYYYMMDD  Fecha DATE NOT NULL,  Anio INT NOT NULL,  Mes INT NOT NULL,  Dia INT NOT NULL,  Nombre\_Mes VARCHAR(20) NOT NULL,  Nombre\_Dia\_Semana VARCHAR(20) NOT NULL,  Numero\_Dia\_Semana INT NOT NULL,  Dia\_Anio INT NOT NULL,  Trimestre INT NOT NULL,  Es\_Fin\_Semana BIT NOT NULL,  Es\_Laborable BIT NOT NULL  );  END  -- Borrar datos existentes (opcional, según tu flujo de carga)  DELETE FROM DWH.LK\_TIEMPO;  -- Insertar fila para valores desconocidos (con fecha ficticia)  INSERT INTO DWH.LK\_TIEMPO (  Fecha\_ID,  Fecha,  Anio,  Mes,  Dia,  Nombre\_Mes,  Nombre\_Dia\_Semana,  Numero\_Dia\_Semana,  Dia\_Anio,  Trimestre,  Es\_Fin\_Semana,  Es\_Laborable  )  VALUES (  -1,  '19000101', -- Fecha ficticia  -1,  -1,  -1,  'Desconocido',  'Desconocido',  -1,  -1,  -1,  0,  0  );  -- Carga de fechas válidas desde 1995-01-01 hasta 2000-12-31  DECLARE @Fecha DATE = '19950101';  DECLARE @Fecha\_Fin DATE = '20001231';  WHILE @Fecha <= @Fecha\_Fin  BEGIN  INSERT INTO DWH.LK\_TIEMPO (  Fecha\_ID,  Fecha,  Anio,  Mes,  Dia,  Nombre\_Mes,  Nombre\_Dia\_Semana,  Numero\_Dia\_Semana,  Dia\_Anio,  Trimestre,  Es\_Fin\_Semana,  Es\_Laborable  )  VALUES (  CONVERT(INT, FORMAT(@Fecha, 'yyyyMMdd')),  @Fecha,  YEAR(@Fecha),  MONTH(@Fecha),  DAY(@Fecha),  DATENAME(MONTH, @Fecha),  DATENAME(WEEKDAY, @Fecha),  DATEPART(WEEKDAY, @Fecha),  DATEPART(DAYOFYEAR, @Fecha),  DATEPART(QUARTER, @Fecha),  CASE WHEN DATENAME(WEEKDAY, @Fecha) IN ('Saturday', 'Sunday') THEN 1 ELSE 0 END,  CASE WHEN DATENAME(WEEKDAY, @Fecha) IN ('Saturday', 'Sunday') THEN 0 ELSE 1 END  );  SET @Fecha = DATEADD(DAY, 1, @Fecha);  END  END;  GO |
| OUTPUT | select \* from [DWH].[LK\_TIEMPO] |

## Procesos de ingesta de Schema ODS

En éste capitulo, documentaremos la carga de los ODS

### 5.1 Proceso Schema ODS 1: CUSTOMERS

|  |  |
| --- | --- |
| JobName | DWH\_Customers.dtsx |
| Input | select \* from [STG].[ODS\_CUSTOMERS] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_ODS\_Customers] Script Date: 18/5/2025 21:43:10 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_ODS\_Customers]  AS  BEGIN  SET NOCOUNT ON;  -- Inicializar Dimensión con valor NA  -- Verificar si el valor NA (-1) ya existe en la tabla LK\_REGIONS  IF NOT EXISTS (SELECT 1 FROM [ODS].[CUSTOMERS] WHERE cus\_customer\_id = -1)  BEGIN  -- Habilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [ODS].[CUSTOMERS] ON;  -- Insertar el valor NA (-1)  INSERT INTO [ODS].[CUSTOMERS] (  cus\_customer\_id, -- Identity, por eso ahora se puede insertar explícitamente  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID,  Cus\_Customer\_CD,  Cus\_Company\_Name,  Cus\_Region\_id,  Cus\_Country\_id  )  VALUES (  -1,  GETDATE(),  SYSTEM\_USER,  1,  0, -- o algún valor identificador especial si tu lógica lo requiere  '-1',  'Not Applicable',  -1,  -1  );  -- Deshabilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [ODS].[CUSTOMERS] OFF;  END;  -- Realizando el MERGE en la tabla de destino con la tabla fuente  MERGE INTO [ODS].[CUSTOMERS] AS target  USING [STG].[ODS\_CUSTOMERS] 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,  Cus\_Customer\_CD,  Cus\_Company\_Name,  Cus\_Contact\_Name,  Cus\_Contact\_Title,  Cus\_address,  Cus\_City,  Cus\_Region\_cd,  Cus\_Region\_id,  Cus\_Postal\_Code,  Cus\_Country\_cd,  Cus\_Country\_id,  Cus\_Phone,  Cus\_fax  )  VALUES (  GETDATE(), -- AUD\_INSERT\_DT  SYSTEM\_USER, -- AUD\_USER\_INSERT  1, -- AUD\_ACTIVE\_FLAG (TRUE)  source.AUD\_ROWID,  source.Cus\_Customer\_CD,  source.Cus\_Company\_Name,  source.Cus\_Contact\_Name,  source.Cus\_Contact\_Title,  source.Cus\_address,  source.Cus\_City,  source.Cus\_Region\_cd,  source.Cus\_Region\_id,  source.Cus\_Postal\_Code,  source.Cus\_Country\_cd,  source.Cus\_Country\_id,  source.Cus\_Phone,  source.Cus\_fax  )  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 [ODS].[CUSTOMERS] |

### 5.2 Proceso Schema ODS 2: ORDERS

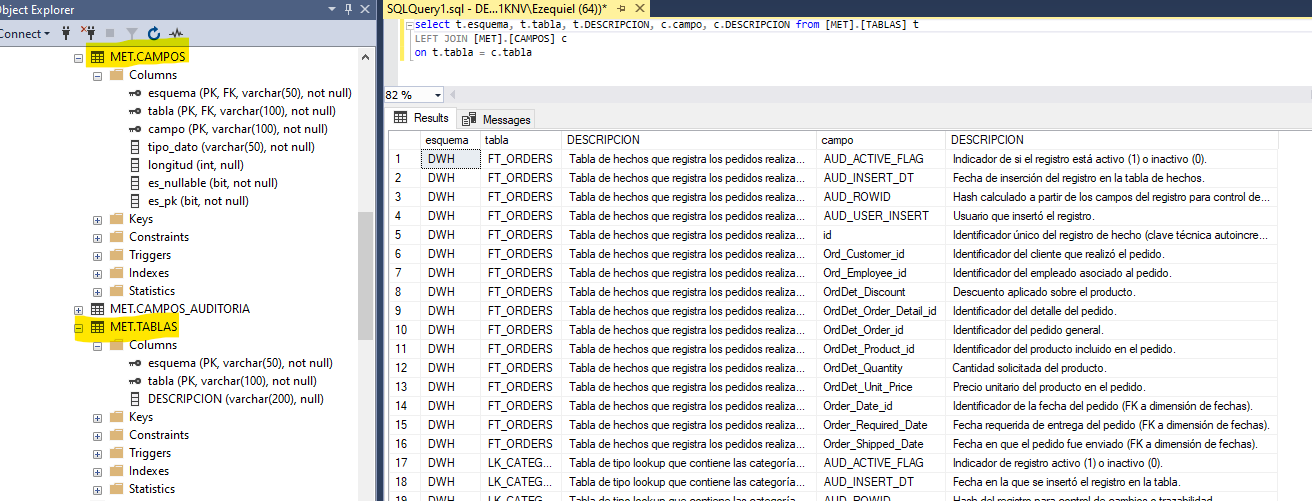
|  |  |
| --- | --- |
| JobName | DWH\_Orders.dtsx |
| Input | select \* from [STG].[ODS\_ORDERS] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_ODS\_Customers] Script Date: 18/5/2025 23:31:19 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_ODS\_Orders]  AS  BEGIN  SET NOCOUNT ON;  -- Inicializar Dimensión con valor NA  -- Verificar si el valor NA (-1) ya existe en la tabla LK\_REGIONS  IF NOT EXISTS (SELECT 1 FROM [ODS].[ORDERS] WHERE Ord\_Order\_id = -1)  BEGIN  -- Habilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [ODS].[ORDERS] ON;  -- Insertar el valor NA (-1)  INSERT INTO [ODS].[ORDERS] (  Ord\_Order\_id, -- Identity, por eso ahora se puede insertar explícitamente  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID,  Ord\_Order\_Date,  Ord\_Ship\_Country\_id,  Ord\_Ship\_Region\_Id  )  VALUES (  -1,  GETDATE(),  SYSTEM\_USER,  1,  0, -- o algún valor identificador especial si tu lógica lo requiere  TRY\_CAST('1900-01-01' AS DATE),  -1,  -1  );  -- Deshabilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [ODS].[ORDERS] OFF;  END;  -- Realizando el MERGE en la tabla de destino con la tabla fuente  MERGE INTO [ODS].[ORDERS] AS target  USING [STG].[ODS\_ORDERS] 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,  Ord\_Order\_CD,  Ord\_Customer\_cd,  Ord\_Customer\_id,  Ord\_Employee\_cd,  Ord\_Employee\_id,  Ord\_Order\_Date,  Ord\_Required\_Date,  Ord\_Shipped\_Date,  Ord\_Ship\_Via,  Ord\_Freight,  Ord\_Ship\_name,  Ord\_Ship\_Address,  Ord\_Ship\_City,  Ord\_Ship\_Region\_cd,  Ord\_Ship\_Region\_Id,  Ord\_Ship\_PostalCode,  Ord\_Ship\_Country\_cd,  Ord\_Ship\_Country\_id  )  VALUES (  GETDATE(), -- AUD\_INSERT\_DT  SYSTEM\_USER, -- AUD\_USER\_INSERT  1, -- AUD\_ACTIVE\_FLAG (TRUE)  source.AUD\_ROWID,  source.Ord\_Order\_CD,  source.Ord\_Customer\_cd,  source.Ord\_Customer\_id,  source.Ord\_Employee\_cd,  source.Ord\_Employee\_id,  source.Ord\_Order\_Date,  source.Ord\_Required\_Date,  source.Ord\_Shipped\_Date,  source.Ord\_Ship\_Via,  source.Ord\_Freight,  source.Ord\_Ship\_name,  source.Ord\_Ship\_Address,  source.Ord\_Ship\_City,  source.Ord\_Ship\_Region\_cd,  source.Ord\_Ship\_Region\_Id,  source.Ord\_Ship\_PostalCode,  source.Ord\_Ship\_Country\_cd,  source.Ord\_Ship\_Country\_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 [ODS].[ORDERS] |

### 5.3 Proceso Schema ODS 3: ORDERS DETAILS

|  |  |
| --- | --- |
| JobName | DWH\_Order\_Details.dtsx |
| Input | select \* from [STG].[ODS\_ORDERS] |
| Control Flow |  |
| Proceso | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_ODS\_Customers] Script Date: 18/5/2025 23:31:19 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_ODS\_Orders]  AS  BEGIN  SET NOCOUNT ON;  -- Inicializar Dimensión con valor NA  -- Verificar si el valor NA (-1) ya existe en la tabla LK\_REGIONS  IF NOT EXISTS (SELECT 1 FROM [ODS].[ORDERS] WHERE Ord\_Order\_id = -1)  BEGIN  -- Habilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [ODS].[ORDERS] ON;  -- Insertar el valor NA (-1)  INSERT INTO [ODS].[ORDERS] (  Ord\_Order\_id, -- Identity, por eso ahora se puede insertar explícitamente  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID,  Ord\_Order\_Date,  Ord\_Ship\_Country\_id,  Ord\_Ship\_Region\_Id  )  VALUES (  -1,  GETDATE(),  SYSTEM\_USER,  1,  0, -- o algún valor identificador especial si tu lógica lo requiere  TRY\_CAST('1900-01-01' AS DATE),  -1,  -1  );  -- Deshabilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [ODS].[ORDERS] OFF;  END;  -- Realizando el MERGE en la tabla de destino con la tabla fuente  MERGE INTO [ODS].[ORDERS] AS target  USING [STG].[ODS\_ORDERS] 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,  Ord\_Order\_CD,  Ord\_Customer\_cd,  Ord\_Customer\_id,  Ord\_Employee\_cd,  Ord\_Employee\_id,  Ord\_Order\_Date,  Ord\_Required\_Date,  Ord\_Shipped\_Date,  Ord\_Ship\_Via,  Ord\_Freight,  Ord\_Ship\_name,  Ord\_Ship\_Address,  Ord\_Ship\_City,  Ord\_Ship\_Region\_cd,  Ord\_Ship\_Region\_Id,  Ord\_Ship\_PostalCode,  Ord\_Ship\_Country\_cd,  Ord\_Ship\_Country\_id  )  VALUES (  GETDATE(), -- AUD\_INSERT\_DT  SYSTEM\_USER, -- AUD\_USER\_INSERT  1, -- AUD\_ACTIVE\_FLAG (TRUE)  source.AUD\_ROWID,  source.Ord\_Order\_CD,  source.Ord\_Customer\_cd,  source.Ord\_Customer\_id,  source.Ord\_Employee\_cd,  source.Ord\_Employee\_id,  source.Ord\_Order\_Date,  source.Ord\_Required\_Date,  source.Ord\_Shipped\_Date,  source.Ord\_Ship\_Via,  source.Ord\_Freight,  source.Ord\_Ship\_name,  source.Ord\_Ship\_Address,  source.Ord\_Ship\_City,  source.Ord\_Ship\_Region\_cd,  source.Ord\_Ship\_Region\_Id,  source.Ord\_Ship\_PostalCode,  source.Ord\_Ship\_Country\_cd,  source.Ord\_Ship\_Country\_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 [ODS].[ORDERS\_DETAILS] |

## Gestión de METADATA & Memoria institucional

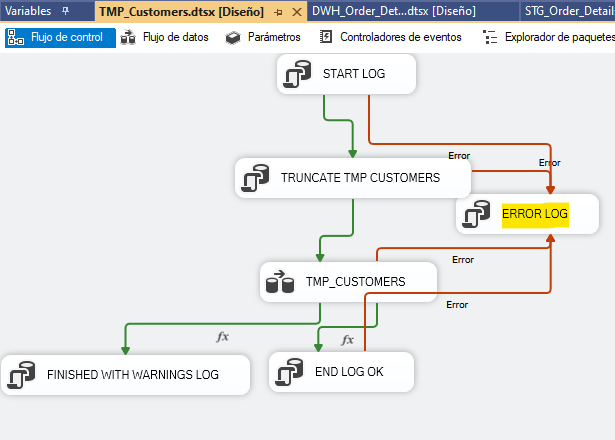
La gestión de metadata consiste en la administración estructurada de la información que describe y contextualiza los datos dentro de un sistema, facilitando su comprensión, uso y gobernanza. Esta metadata incluye detalles como la estructura de las tablas, los tipos de datos, las relaciones, las reglas de negocio y las descripciones de cada elemento, lo que permite un manejo eficiente y controlado del entorno de datos. En nuestro caso, hemos creado un repositorio de metadata que documenta las tablas y columnas del Data Warehouse, específicamente la tabla de dimensión temporal [DWH].[LK\_TIEMPO]. Esto incluye registrar la definición de la tabla en la tabla de metadata de tablas [MET].[TABLAS] y describir cada campo en la tabla de metadata de campos [MET].[CAMPOS], con detalles sobre el tipo de dato, longitud, obligatoriedad y propósito de cada columna. Adicionalmente, en el documento asociado se detalla exhaustivamente cada uno de los procesos, los orquestadores utilizados y la gestión del cambio aplicada, contribuyendo así a la memoria institucional. Esta memoria institucional garantiza que el conocimiento acumulado y las prácticas adoptadas queden formalmente registradas, facilitando la continuidad operativa, el aprendizaje organizacional y la trazabilidad para futuros equipos. Este proceso integral es fundamental para asegurar la calidad, trazabilidad y entendimiento común de los datos, apoyando el análisis y la toma de decisiones dentro de la organización.

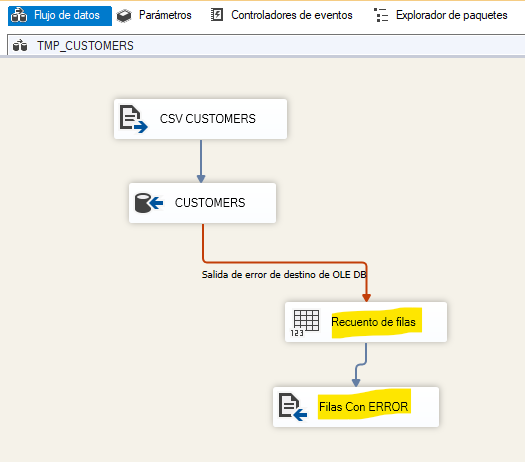


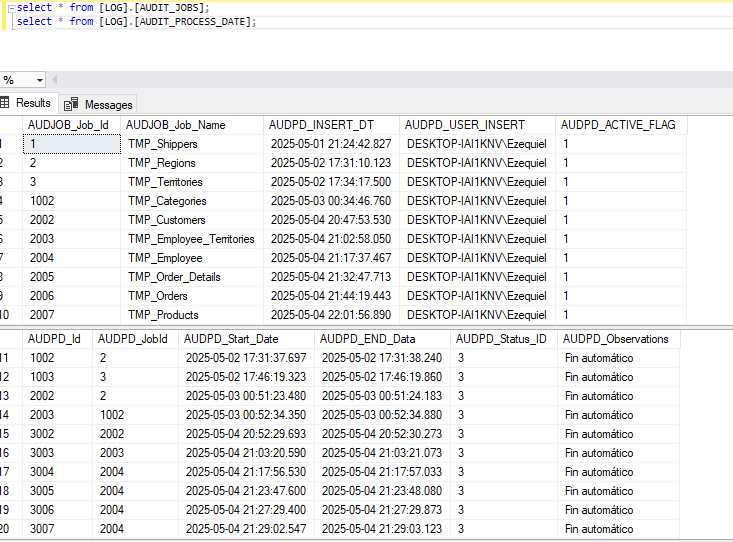
## DQM Gestión de Logs

La gestión de logs en el sistema es un componente esencial del Data Quality Management (DQM), ya que permite registrar de manera automática y estructurada el inicio y fin de cada proceso ejecutado dentro del Data Warehouse. A través de procedimientos almacenados específicos, se captura información crítica como el nombre del job, las marcas temporales de inicio y finalización, y el estado del proceso al concluir. Este control exhaustivo no solo facilita el monitoreo y la trazabilidad, sino que también se integra como un mecanismo clave para evaluar la calidad operativa de los procesos ETL/ELT. La persistencia de estos registros permite analizar la performance, detectar y corregir errores a tiempo, y garantizar la confiabilidad y consistencia del flujo de datos. En consecuencia, la gestión de logs dentro del marco del DQM contribuye a mantener la integridad, transparencia y mejora continua en la administración y calidad de los procesos del sistema.

En cada uno de los procesos, se hace una gestión de errores, y almacenamiento de métricas de la corrida;

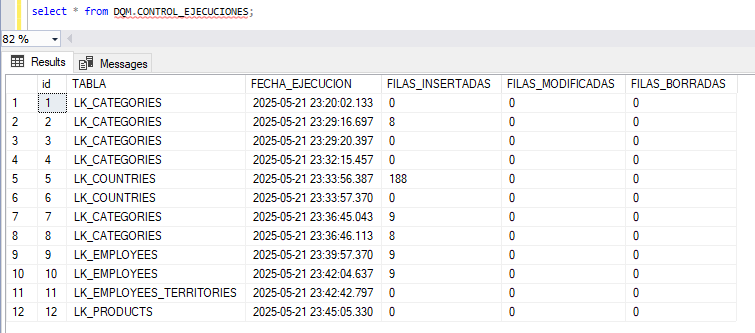






|  |  |
| --- | --- |
| START LOG | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [LOG].[DWH\_Start\_Log] Script Date: 20/5/2025 23:16:04 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE PROCEDURE [LOG].[DWH\_Start\_Log]  @JobName VARCHAR(50)  AS  BEGIN  SET NOCOUNT ON;  DECLARE @JobId INT;  DECLARE @StatusId INT;  -- Obtener ID del Job activo  SELECT @JobId = AUDJOB\_Job\_Id  FROM [LOG].[AUDIT\_JOBS]  WHERE AUDJOB\_Job\_Name = @JobName  AND AUDPD\_ACTIVE\_FLAG = 1;  IF @JobId IS NULL  BEGIN  RAISERROR('No se encontró un Job activo con el nombre proporcionado.', 16, 1);  RETURN;  END  -- Obtener ID del estado 'RUNNING' activo  SELECT @StatusId = AUDSP\_Status\_Id  FROM [LOG].[AUDIT\_STATUS\_PROCESS]  WHERE AUDPD\_Status = 'RUNNING'  AND AUDPD\_ACTIVE\_FLAG = 1;  IF @StatusId IS NULL  BEGIN  RAISERROR('No se encontró un estado activo con nombre RUNNING.', 16, 1);  RETURN;  END  -- Insertar registro en tabla de auditoría de ejecución  INSERT INTO [LOG].[AUDIT\_PROCESS\_DATE] (  AUDPD\_JobId,  AUDPD\_Start\_Date,  AUDPD\_END\_Data,  AUDPD\_Status\_ID,  AUDPD\_Observations  )  VALUES (  @JobId,  GETDATE(),  NULL,  @StatusId,  'Inicio automático'  );  END;  GO |
| END LOG | USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [LOG].[DWH\_Start\_Log] Script Date: 20/5/2025 23:13:10 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [LOG].[DWH\_Start\_Log]  @JobName VARCHAR(50)  AS  BEGIN  SET NOCOUNT ON;  DECLARE @JobId INT;  DECLARE @StatusId INT;  -- Obtener ID del Job activo  SELECT @JobId = AUDJOB\_Job\_Id  FROM [LOG].[AUDIT\_JOBS]  WHERE AUDJOB\_Job\_Name = @JobName  AND AUDPD\_ACTIVE\_FLAG = 1;  IF @JobId IS NULL  BEGIN  RAISERROR('No se encontró un Job activo con el nombre proporcionado.', 16, 1);  RETURN;  END  -- Obtener ID del estado 'RUNNING' activo  SELECT @StatusId = AUDSP\_Status\_Id  FROM [LOG].[AUDIT\_STATUS\_PROCESS]  WHERE AUDPD\_Status = 'RUNNING'  AND AUDPD\_ACTIVE\_FLAG = 1;  IF @StatusId IS NULL  BEGIN  RAISERROR('No se encontró un estado activo con nombre RUNNING.', 16, 1);  RETURN;  END  -- Insertar registro en tabla de auditoría de ejecución  INSERT INTO [LOG].[AUDIT\_PROCESS\_DATE] (  AUDPD\_JobId,  AUDPD\_Start\_Date,  AUDPD\_END\_Data,  AUDPD\_Status\_ID,  AUDPD\_Observations  )  VALUES (  @JobId,  GETDATE(),  NULL,  @StatusId,  'Inicio automático'  );  END; |

También se hizo un registro de la cantidad de filas actualizadas en cada una de las corridas, para poder tener un control de la variabilidad de los datos en cada una de las ejecuciones.



El control, se aplica en cada uno de los SPs de Actualización, de la siguiente manera

|  |
| --- |
| USE [TP\_DWH\_MCD\_2025]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SP\_DWH\_Categories] Script Date: 21/5/2025 23:50:28 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[SP\_DWH\_Categories]  AS  BEGIN  SET NOCOUNT ON;  -- Variables para contar operaciones  DECLARE @Insertadas INT = 0;  DECLARE @Modificadas INT = 0;  DECLARE @Borradas INT = 0;  -- Tabla para capturar resultados del MERGE  DECLARE @ResultadosMerge TABLE (Accion VARCHAR(10));  -- Inicializar Dimensión con valor NA  -- Verificar si el valor NA (-1) ya existe en la tabla LK\_REGIONS  IF NOT EXISTS (SELECT 1 FROM [DWH].[LK\_CATEGORIES] WHERE Category\_Id = -1)  BEGIN  -- Habilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_CATEGORIES] ON;  -- Insertar el valor NA (-1)  INSERT INTO [DWH].[LK\_CATEGORIES] (  Category\_Id, -- Identity, por eso ahora se puede insertar explícitamente  AUD\_INSERT\_DT,  AUD\_USER\_INSERT,  AUD\_ACTIVE\_FLAG,  AUD\_ROWID,  Category\_CD,  Category\_Name,  Category\_Description  )  VALUES (  -1,  GETDATE(),  SYSTEM\_USER,  1,  0, -- o algún valor identificador especial si tu lógica lo requiere  '-1',  'Not Applicable',  'Not Applicable'  );  -- Deshabilitar IDENTITY\_INSERT  SET IDENTITY\_INSERT [DWH].[LK\_CATEGORIES] OFF;  END;  -- 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  OUTPUT  $action INTO @ResultadosMerge(Accion);  -- Contar operaciones  SELECT  @Insertadas = COUNT(\*) FROM @ResultadosMerge WHERE Accion = 'INSERT';  SELECT  @Borradas = COUNT(\*) FROM @ResultadosMerge WHERE Accion = 'UPDATE';  -- Registrar ejecución en tabla de control  INSERT INTO DQM.CONTROL\_EJECUCIONES (  TABLA,  FECHA\_EJECUCION,  FILAS\_INSERTADAS,  FILAS\_MODIFICADAS,  FILAS\_BORRADAS  )  VALUES (  'LK\_CATEGORIES',  GETDATE(),  @Insertadas,  0, -- No se manejan updates como modificaciones en este proceso  @Borradas  );  END; |