

Caso práctico: Almacén de datos para el análisis del impacto ambiental y consumo energético derivados de la actividad económica

Solución PRA2- Carga de datos

Índice de contenidos

1.	Creación de la matriz de dimensiones y métricas.	2
2.	Identificación de los procesos ETL	4
2.1.	Bloque IN (de las fuentes a tablas intermedias)	5
2.2.	Bloque TR (poblar las tablas de nuestro almacén)	6
3.	Diseño y desarrollo de los procesos de ETL	7
3.1	Creación de tablas intermedias (<i>staging area</i>)	7
3.2	Creación del modelo multidimensional	8
3.2.1.	Dimensiones	9
3.2.2.	Tablas de hechos	10
3.2.3.	Claves foráneas	11
3.3	Creación del proceso de extracción, transformación y carga	13
3.3.1	Variables de entorno	13
3.3.2	Conexión a la base de datos SQL Server	14
3.3.3	Bloque IN	14
3.3.3.1	Transformación «IN_INVESTMENTS»	15
3.3.3.2	Transformación «IN_COUNTRIES»	19
3.3.3.3	Transformación «IN_PROTECTED_AREA»	21
3.3.3.4	Transformación «IN_ENERGY_BALANCE»	24
3.3.3.5	Transformación «IN_URBAN_WASTES»	27
3.3.3.6	Transformación «IN_OBJECTIVES»	34
3.3.4	Bloque TR	36

3.3.5 Bloque TR_DIM	37
3.3.5.1 TR_DIM_Date	37
3.3.5.2 TR_DIM_Region	39
3.3.5.3 TR_DIM_SDG	40
3.3.5.4 TR_DIM_Measurement	42
3.3.5.5 TR_DIM_EconomicActivitySector	48
3.3.5.6 TR_DIM_TypeEquipmentInstallation	48
3.3.5.7 TR_DIM_Country	49
3.3.5.8 TR_DIM_Product	50
3.3.6 Bloque TR_FACT	50
3.3.6.1 TR_FACT_EnvironmentalMeasurements	51
3.3.6.2 TR_FACT_EnergyBalances	54
4. Implementación con trabajos (jobs) de los procesos ETL	60
4.1 JOB_IN	60
4.2 JOB_TR_DIM	61
4.3 JOB_TR_FACT	63
4.4 JOB_DW	64

1. Creación de la matriz de dimensiones y métricas.

La Matriz de Dimensiones y Métricas (Kimball, *Data Warehouse Toolkit*, 2013) es una herramienta clave en el diseño del Data Warehouse, que representa los procesos centrales de la organización y la dimensionalidad asociada. Este modelo busca proporcionar la perspectiva necesaria para garantizar que toda la empresa pueda integrar sus datos en el entorno del Data Warehouse.

Partiendo del análisis de requerimientos y diseño del Data Warehouse, esta herramienta debe servir de ayuda para la creación de procesos ETL que van a permitir la carga del Data Warehouse. Su finalidad es ayudar a organizar las ideas de forma que se pueda tener una visión completa sobre cómo se relacionan estos procesos y qué transformaciones son necesarias para llegar al modelo dimensional buscado.

PROCESO	Environmental Measurements			Energy Balance
	Inversión en protección ambiental (€)	Áreas protegidas (km², %)	Residuos urbanos (toneladas, %, índice, Kg/Hab)	Flujos o circulantes que se consideran en la medición del balance energético
Dimensión \ Métrica	ambito_medioambiental (STG_Investments)	Areaprot (STG_ProtectedAreas)	Var-Unit (STG_UrbanWastes)	Flow (STG_EnergyBalance)
	-Protección de la biodiversidad y los paisajes (€) -Protección del aire y el clima (€) -Gestión de aguas residuales (€) -Gestión de residuos (€) -Otras actividades de protección ambiental (€) -Protección y descontaminación de suelos, aguas subterráneas y superficiales (€) -Reducción del ruido y las vibraciones (€)	-Área protegida marina (MPA_KM2) -Área protegida terrestre (TPA_KM2) -Área protegida terrestre (TPA_PC) (%)	-BULKY (Tn) -COMPOST (Tn) -COMPST_SHARE (%) -DISPOSAL (Tn) -HOUSE_LIKE (Tn) -HOUSEHOLD (Tn) -INC_SHARE (%) -INC_WITH_SHARE (%) -INC_WITHOUT_SHARE (%) -INCINERATION (Tn) -INCINERATION_WITH (Tn) -INCINERATION_WITHOUT (Tn) -INDEX1990 (idx) -INDEX2000 (idx)	-LANDF_SHARE (%) -LANDFILL (Tn) -MAT_RECOV_SHARE (%) -MUNICIPAL (Tn) -MW_CAP (Kg/Hab) -OTH_DISP (Tn) -OTH_RECOV (Tn) -OTHMUN (Tn) -RECOVERY (Tn) -RECYCLING (Tn) -RECYCLING_SHARE (%) -TREATMENT (Tn) -WEEE (Tn)
DIM_Date	X	X	X	X
DIM_Region	X	X	X	N/A
DIM_EconomicActivitySector	X	N/A	N/A	N/A
DIM_TypeEquipmentInstallation	X	N/A	N/A	N/A
DIM_SDG	X	X	X	X
DIM_Measurement	X	X	X	X
DIM_Product	N/A	N/A	N/A	X
DIM_Country	N/A	N/A	N/A	X

2. Identificación de los procesos ETL

A la hora de diseñar el proceso de carga de un Data Warehouse no hay una única estrategia que sirva para todos los casos. Por un lado, es habitual estructurar los procesos de ETL sobre la base de las entidades de datos que se deben actualizar. Por otro lado, encontramos diferencias conceptuales entre la actualización de una dimensión y la de una tabla de hechos. Asimismo, la división del proceso en diferentes bloques de actualización facilitará diseñar un orden de ejecución y gestionar las dependencias. Cada uno de estos bloques de actualización se dividirá en las correspondientes etapas de extracción, transformación y carga.

Esta separación permite ejecutar los bloques de forma consecutiva, que es lo más habitual, pero también posibilita ejecutarlos de forma aislada si se requiere reprocesar alguno de los bloques, por cualquier incidencia que se haya producido en la ejecución consecutiva. Cada uno de estos bloques de actualización se corresponderá con una transformación de la herramienta *Pentaho Data Integration* (PDI).

En el diseño de estos procesos deben considerarse una serie de factores:

- Orden de carga de los bloques.
- Ventana de tiempo disponible.
- Tipo de carga: inicial o incremental.
- Uso de un área intermedia o de maniobras (*staging area*).

Dado que se trata de un caso académico iniciado de cero, vamos a realizar una carga inicial, por lo que los procesos no serán diseñados con el objetivo de repetirse periódicamente. También es cierto que se desconoce la ventana de tiempo disponible (lo que no es una condición para esta actividad), sin embargo esto no ocurre en un entorno de producción, donde éste es considerado un factor muy relevante. En lo relativo al área intermedia hay que tener en cuenta su uso, ya que permite guardar la información de origen en bruto, puede facilitar mucho el trabajo de depuración de la información o de trazabilidad del dato.

Se identificarán dos bloques y se utilizará un prefijo en el nombre para identificarlos:

- **Bloque IN:** procesos de carga de los datos desde las fuentes a tablas intermedias en el área de maniobras (*staging area*). Estos procesos se distinguen por el prefijo: «IN_» en el nombre.
- **Bloque TR:** procesos de transformación para la carga de datos desde tablas intermedias a nuestro almacén según el modelo multidimensional diseñado. Se diferencian los procesos de ETL de transformación, para la carga de dimensiones (TR_DIM_), de los procesos de transformación, para la carga de las tablas de hecho (TR_FACT_). Estos procesos se distinguen con el prefijo «TR_» en el nombre.

Veamos a continuación los procesos de los dos bloques identificados.

2.1. Bloque IN (de las fuentes a tablas intermedias)

Nombre ETL	Descripción	Orígenes de datos	Tabla destino (<i>stage</i>)
IN_INVESTMENTS	Carga de datos correspondientes a la evolución de la inversión en protección ambiental por tipo de equipo e instalación, ámbito medioambiental y sector de actividad económica.	02002.xlsx	STG_Investments
IN_COUNTRIES	Carga de datos correspondientes a los nombres de los países (nombres cortos oficiales en español, inglés y francés) en orden alfabético (ISO 3166-1) y los elementos de código ISO 3166-1-alpha-2 y alpha-3 en formato JSON	Countries.json	STG_Countries
IN_PROTECTED_AREAS	Carga de datos de la Biodiversidad, información de áreas protegidas tanto marinas como terrestres.	env_bio1.tsv	STG_ProtectedAreas
IN_ENERGY_BALANCE	Carga de datos de todos aquellos aspectos relevantes del balance energético mundial.	WorldEnergyBalancesHighlights_final.xlsx:	STG_EnergyBalance
IN_URBAN_WASTES	Carga de los datos de los residuos urbanos, su generación y tratamiento	DataGeneric.xml	STG_UrbanWastes
IN_OBJECTIVES	Carga de los datos correspondientes a los Objetivos de Desarrollo Sostenible con los ámbitos medioambientales.	ODS.xlsx (dos pestañas)	STG_Objectives STG_ObjectivesAreas

2.2. Bloque TR (poblar las tablas de nuestro almacén)

El bloque «TR_» de procesos ETL para poblar el modelo multidimensional del almacén tiene dos partes diferenciadas: los procesos de carga y transformación de las dimensiones y los de las tablas de hechos. El orden de ejecución es importante para que la carga de datos sea correcta. Las dimensiones se cargarán primero y, después, las tablas de hechos, si no ha habido errores.

Los procesos del bloque de carga y transformación de las dimensiones son los siguientes:

Nombre ETL	Descripción	Tabla origen	Tabla destino
TR_DIM_Date	Carga y transformación de la dimensión temporal.	SQL	DIM_Date
TR_DIM_Region	Carga de la dimensión con información de la región o ámbito geográfico donde se localiza la medición.	STG_Countries STG_Investments	Dim_Region
TR_DIM_EconomicActivitySector	Carga y transformación de la dimensión con datos del sector de la actividad económica al que se clasifica la medición.	STG_Investments	Dim_EconomicActivitySector
TR_DIM_TypeEquipmentInstallation	Carga de la dimensión con los tipos de equipamientos o instalación de la medición.	STG_Investments	Dim_TypeEquipmentInstallation
TR_DIM_SDG	Carga y transformación de la dimensión con datos los Objetivos de Desarrollo Sostenible.	STG_Objectives	DIM_SDG (Sustainable Development Goals)
TR_DIM_Measurement	Carga de la dimensión con las mediciones a tratar, relacionadas con los objetivos SDG.	STG_Investments STG_ProtectedAreas STG_UrbanWastes STG_EnergyBalance	Dim_Measurement
TR_DIM_Country	Carga de la dimensión de países donde se localiza la medición del balance energético	STG_Countries	Dim_Country
TR_DIM_Product	Carga de la dimensión con datos de productos que intervienen en la medición del balance energético.	STG_EnergyBalance	Dim_Product

Los procesos del bloque de carga y transformación de las tablas de hechos son:

Nombre ETL	Descripción	Tabla origen
TR_FACT_EnvironmentalMeasurements	Carga y transformación de la tabla de hechos «FACT_EnvironmentalMeasurements» con las mediciones ambientales para un desarrollo sostenible.	STG_Investments STG_ProtectedAreas STG_UrbanWastes
TR_FACT_EnergyBalances	Carga y transformación de la tabla de hechos «FACT_EnergyBalances» con datos del balance energético mundial de los países productores y consumidores de energía.	STG_EnergyBalance

Existen otras estrategias válidas que nos permitirán cargar los datos, ya sea organizando los procesos de otra forma o fusionándolos en un único proceso que lleve a cabo todas las tareas.

La opción de un único proceso de ETL se podría aplicar, pero no es recomendable en términos de mantenimiento ni en cargas más complejas y cambiantes.

3. Diseño y desarrollo de los procesos de ETL

En este apartado, y teniendo en cuenta las consideraciones anteriores, vamos a diseñar e implementar los procesos de carga mediante la herramienta de diseño proporcionada: *Pentaho Data Integration* (PDI). Y, en particular, el programa de escritorio llamado Spoon, que corresponde al entorno gráfico IDE de desarrollo de los procedimientos de ETL.

Los procesos de ETL que diseñaremos en PDI consistirán en la definición de trabajos y transformaciones. Estas contienen la operativa de bajo nivel con las acciones que hay que realizar sobre los datos, mientras que los trabajos son procesos de alto nivel compuestos por flujos de transformaciones.

3.1 Creación de tablas intermedias (*staging area*)

El primer paso para la implementación del proceso de ETL consiste en la creación de las tablas intermedias en la *staging area*. Esta se llevará a cabo una única vez, mediante *scripts* sobre la base de datos, en nuestro caso SQL Server. Las tablas intermedias se utilizarán en los procesos IN, que permitirán cargar los datos desde las fuentes de datos.

Las tablas intermedias se han creado sin restricciones ni índices para facilitar la carga de datos desde las fuentes de origen.

```
CREATE TABLE [dbo].[STG_Investments](
    [periodo] [int] NULL,
```

```

[sector_economico] [varchar](255) NULL,
[tipo_instalacion] [varchar](255) NULL,
[ambito_medioambiental] [varchar](255) NULL,
[comunidad_autonoma] [varchar](255) NULL,
[inversion] [float] NULL) ON [PRIMARY]

GO

CREATE TABLE [dbo].[STG_Countries](
[nombre] [varchar](255) NULL,
[name] [varchar](255) NULL,
[nom] [varchar](255) NULL,
[iso2] [varchar](255) NULL,
[iso3] [varchar](255) NULL,
[phone_code] [varchar](255) NULL) ON [PRIMARY]

GO

CREATE TABLE [dbo].[STG_ProtectedAreas](
[areaprot] [varchar](255) NULL,
[geotime] [varchar](255) NULL,
[year] [int] NULL,
[value] [float] NULL) ON [PRIMARY]

GO

CREATE TABLE [dbo].[STG_EnergyBalance](
[country] [varchar](255) NULL,
[product] [varchar](255) NULL,
[flow] [varchar](255) NULL,
[year] [int] NULL,
[value] [float] NULL) ON [PRIMARY]

GO

CREATE TABLE [dbo].[STG_UrbanWastes](
[GROUP_ID] INT NULL,
[COU] [varchar](255) NULL,
[VAR] [varchar](255) NULL,
[TIME_FORMAT] [varchar](255) NULL,
[UNIT] [varchar](255) NULL,
[POWERCODE] [varchar](255) NULL,
[TIME] [varchar](255) NULL,
[VALUE] [varchar](255) NULL,
[OBS_STATUS] [varchar](255) NULL) ON [PRIMARY]

GO

CREATE TABLE [dbo].[STG_Objectives](
[Objetivo] [int] NULL,
[Nombre] [varchar](100) NULL,
[Descripcion] [varchar](512) NULL) ON [PRIMARY]

GO

CREATE TABLE [dbo].[STG_ObjectivesAreas](
[Codigo][varchar](255) NULL,
[Area] [varchar](255) NULL,
[ODS_Principal] [int] NULL) ON [PRIMARY]

GO

```

3.2 Creación del modelo multidimensional

En este punto se incluyen los *scripts* de creación del modelo físico multidimensional diseñados para el almacén de datos, compuestos por las dimensiones y las tablas de hechos. En la creación, además de atributos y métricas, también se aplicarán las restricciones definidas y que son propias del modelo multidimensional, las claves primarias de las dimensiones y las foráneas de las tablas de hechos.

3.2.1. Dimensiones

```
CREATE TABLE [dbo].[DIM_Measurement](
    [pk_measurement] [int] NOT NULL,
    [measurement_code] [varchar](100) NULL,
    [measurement_name] [varchar](200) NULL,
    [unit] [varchar](25) NULL,
    [fk_sdg] [int] NULL,
    CONSTRAINT [PK_DIM_Measurement] PRIMARY KEY CLUSTERED
(
    [pk_measurement] ASC
)) ON [PRIMARY]
GO

CREATE TABLE [dbo].[DIM_Date](
    [pk_date] [int] NOT NULL,
    [date_year] [int] NOT NULL,
    [date_month] [int] NOT NULL,
    [date_day] [int] NOT NULL,
    [date_date] [datetime] NOT NULL,
    CONSTRAINT [PK_DIM_Date] PRIMARY KEY CLUSTERED
(
    [pk_date] ASC
)) ON [PRIMARY]
GO

CREATE TABLE [dbo].[DIM_EconomicActivitySector](
    [pk_activitysector] [int] NOT NULL,
    [activitysector_name] [varchar](100) NULL,
    CONSTRAINT [PK_DIM_EconomicActivitySector] PRIMARY KEY CLUSTERED
(
    [pk_activitysector] ASC
)) ON [PRIMARY]
GO

CREATE TABLE [dbo].[DIM_TypeEquipmentInstallation](
    [pk_typeequipinstall] [int] NOT NULL,
    [typeequipinstall_name] [varchar](100) NULL,
    CONSTRAINT [PK_DIM_TypeEquipmentInstallation] PRIMARY KEY CLUSTERED
(
    [pk_typeequipinstall] ASC
)) ON [PRIMARY]
GO

CREATE TABLE [dbo].[DIM_Product](
    [pk_product] [int] NOT NULL,
    [product_name] [varchar](100) NULL,
    CONSTRAINT [PK_DIM_Product] PRIMARY KEY CLUSTERED
(
```

```

        [pk_product] ASC)) ON [PRIMARY]
GO

CREATE TABLE [dbo].[DIM_Region](
    [pk_region] [int] NOT NULL,
    [region] [varchar](100) NULL,
    [country_code2] [varchar](2) NULL,
    [country_code3] [varchar](3) NULL,
    [country_name] [varchar](100) NULL,
    CONSTRAINT [PK_DIM_Region] PRIMARY KEY CLUSTERED
(
    [pk_region] ASC)) ON [PRIMARY]
GO

CREATE TABLE [dbo].[DIM_SDG](
    [pk_sdg] [int] NOT NULL,
    [sdg_name] [varchar](50) NULL,
    [sdg_description] [varchar](500) NULL,
    CONSTRAINT [PK_DIM_SDG] PRIMARY KEY CLUSTERED
(
    [pk_sdg] ASC)) ON [PRIMARY]
GO

CREATE TABLE [dbo].[DIM_Country](
    [pk_country] [int] NOT NULL,
    [country_code] [varchar](2) NULL,
    [country_code3] [varchar](3) NULL,
    [country_name_sp] [varchar](100) NULL,
    [country_name_en] [varchar](100) NULL,
    [country_name_fr] [varchar](100) NULL,
    [country_phone_code] [varchar](5) NULL,
    CONSTRAINT [PK_DIM_Country] PRIMARY KEY CLUSTERED
(
    [pk_country] ASC)) ON [PRIMARY]
GO

```

3.2.2. Tablas de hechos

```

CREATE TABLE [dbo].[FACT_EnvironmentalMeasurements](
    [pk_id] [int] NOT NULL,
    [fk_date] [int] NOT NULL,
    [fk_region] [int] NOT NULL,
    [fk_activitysector] [int] NOT NULL,
    [fk_typeequipinstall] [int] NOT NULL,
    [fk_measurement] [int] NOT NULL,
    [value] [decimal](19, 4) NULL,
    CONSTRAINT [PK_FACT_EnvironmentalMeasurements] PRIMARY KEY CLUSTERED
(
    [pk_id] ASC)) ON [PRIMARY]
GO

CREATE TABLE [dbo].[FACT_EnergyBalances](
    [pk_fk_date] [int] NOT NULL,
    [pk_fk_country] [int] NOT NULL,
    [pk_fk_product] [int] NOT NULL,
    [pk_fk_measurement] [int] NOT NULL,
    [value] [decimal](19, 4) NULL,

```

```

CONSTRAINT [PK_FACT_EnergyBalances] PRIMARY KEY CLUSTERED
(
    [pk_fk_date] ASC,
    [pk_fk_country] ASC,
    [pk_fk_product] ASC,
    [pk_fk_measurement] ASC)) ON [PRIMARY]
GO

```

3.2.3. Claves foráneas

```

ALTER TABLE [dbo].[FACT_EnvironmentalMeasurements] WITH CHECK ADD CONSTRAINT
[FK_FACT_EnvironmentalMeasurements_DIM_Date] FOREIGN KEY([fk_date])
REFERENCES [dbo].[DIM_Date] ([pk_date])
GO

```

```

ALTER TABLE [dbo].[FACT_EnvironmentalMeasurements] CHECK CONSTRAINT
[FK_FACT_EnvironmentalMeasurements_DIM_Date]
GO

```

```

ALTER TABLE [dbo].[FACT_EnvironmentalMeasurements] WITH CHECK ADD CONSTRAINT
[FK_FACT_EnvironmentalMeasurements_DIM_EconomicActivitySector] FOREIGN
KEY([fk_activitysector])
REFERENCES [dbo].[DIM_EconomicActivitySector] ([pk_activitysector])
GO

```

```

ALTER TABLE [dbo].[FACT_EnvironmentalMeasurements] CHECK CONSTRAINT
[FK_FACT_EnvironmentalMeasurements_DIM_EconomicActivitySector]
GO

```

```

ALTER TABLE [dbo].[FACT_EnvironmentalMeasurements] WITH CHECK ADD CONSTRAINT
[FK_FACT_EnvironmentalMeasurements_DIM_Measurement] FOREIGN KEY([fk_measurement])
REFERENCES [dbo].[DIM_Measurement] ([pk_measurement])
GO

```

```

ALTER TABLE [dbo].[FACT_EnvironmentalMeasurements] CHECK CONSTRAINT
[FK_FACT_EnvironmentalMeasurements_DIM_Measurement]
GO

```

```

ALTER TABLE [dbo].[FACT_EnvironmentalMeasurements] WITH CHECK ADD CONSTRAINT
[FK_FACT_EnvironmentalMeasurements_DIM_Region] FOREIGN KEY([fk_region])
REFERENCES [dbo].[DIM_Region] ([pk_region])
GO

```

```

ALTER TABLE [dbo].[FACT_EnvironmentalMeasurements] CHECK CONSTRAINT
[FK_FACT_EnvironmentalMeasurements_DIM_Region]
GO

```

```

ALTER TABLE [dbo].[FACT_EnvironmentalMeasurements] WITH CHECK ADD CONSTRAINT
[FK_FACT_EnvironmentalMeasurements_DIM_TypeEquipmentInstallation] FOREIGN
KEY([fk_typeequipinstall])
REFERENCES [dbo].[DIM_TypeEquipmentInstallation] ([pk_typeequipinstall])
GO

```

```

ALTER TABLE [dbo].[FACT_EnvironmentalMeasurements] CHECK CONSTRAINT
[FK_FACT_EnvironmentalMeasurements_DIM_TypeEquipmentInstallation]
GO

```

```

ALTER TABLE [dbo].[FACT_EnergyBalances] WITH CHECK ADD CONSTRAINT
[FK_FACT_EnergyBalances_DIM_Country] FOREIGN KEY([pk_fk_country])
REFERENCES [dbo].[DIM_Country] ([pk_country])

```

GO

```
ALTER TABLE [dbo].[FACT_EnergyBalances] CHECK CONSTRAINT
[FK_FACT_EnergyBalances_DIM_Country]
```

GO

```
ALTER TABLE [dbo].[FACT_EnergyBalances] WITH CHECK ADD CONSTRAINT
[FK_FACT_EnergyBalances_DIM_Date] FOREIGN KEY([pk_fk_date])
REFERENCES [dbo].[DIM_Date] ([pk_date])
```

GO

```
ALTER TABLE [dbo].[FACT_EnergyBalances] CHECK CONSTRAINT
[FK_FACT_EnergyBalances_DIM_Date]
```

GO

```
ALTER TABLE [dbo].[FACT_EnergyBalances] WITH CHECK ADD CONSTRAINT
[FK_FACT_EnergyBalances_DIM_Measurement] FOREIGN KEY([pk_fk_measurement])
REFERENCES [dbo].[DIM_Measurement] ([pk_measurement])
```

GO

```
ALTER TABLE [dbo].[FACT_EnergyBalances] CHECK CONSTRAINT
[FK_FACT_EnergyBalances_DIM_Measurement]
```

GO

```
ALTER TABLE [dbo].[FACT_EnergyBalances] WITH CHECK ADD CONSTRAINT
[FK_FACT_EnergyBalances_DIM_Product] FOREIGN KEY([pk_fk_product])
REFERENCES [dbo].[DIM_Product] ([pk_product])
```

GO

```
ALTER TABLE [dbo].[FACT_EnergyBalances] CHECK CONSTRAINT
[FK_FACT_EnergyBalances_DIM_Product]
```

GO

```
ALTER TABLE [dbo].[DIM_Measurement] WITH CHECK ADD CONSTRAINT
[FK_DIM_Measurement_DIM_SDG] FOREIGN KEY([fk_sdg])
REFERENCES [dbo].[DIM_SDG] ([pk_sdg])
```

GO

```
ALTER TABLE [dbo].[DIM_Measurement] CHECK CONSTRAINT [FK_DIM_Measurement_DIM_SDG]
```

GO

3.3 Creación del proceso de extracción, transformación y carga

Una vez implementado el modelo físico del almacén, pasaremos a diseñar los procesos de ETL que permitirán poblar las tablas intermedias del área intermedia (*staging area*) y las tablas de dimensiones y de hechos del *data mart*.

Antes del diseño de las transformaciones, se definirán en PDI las variables de entorno que se utilizarán en la implementación de los procesos de ETL, así como la conexiones a las bases de datos utilizadas en todos ellos.

3.3.1 Variables de entorno

Es una buena práctica utilizar variables de entorno para evitar introducir errores en definiciones repetitivas durante la implementación de los procesos. PDI os permite añadir variables personalizadas y propias de vuestros desarrollos en el archivo «*kettle.properties*».

Se utilizarán tres variables. Una para almacenar la ruta de las fuentes de datos y otras dos para reunir las cadenas de conexión a la base de datos, «CN_STAGE» (área intermedia / *staging area*) y «CN_DW» (Data Warehouse). Se podría crear un esquema *stage* en el SQL Server dentro de la base de datos asignada al estudiante para cargar las tablas intermedias (IN_) y definir la variable «CN_STAGE» haciendo referencia a este esquema, pero para simplificar la solución de la práctica se cargarán todas las tablas al esquema por defecto, *dbo*.

Variable	Valor
DIR_IN	F:\fuentes
CN_STAGE	jdbc:sqlserver://UCS1R1UOCSQLXX:1433;databaseName= DB_loginuoc;integratedSecurity=false
CN_DW	jdbc:sqlserver://UCS1R1UOCSQLXX:1433;databaseName= DB_loginuoc;integratedSecurity=false

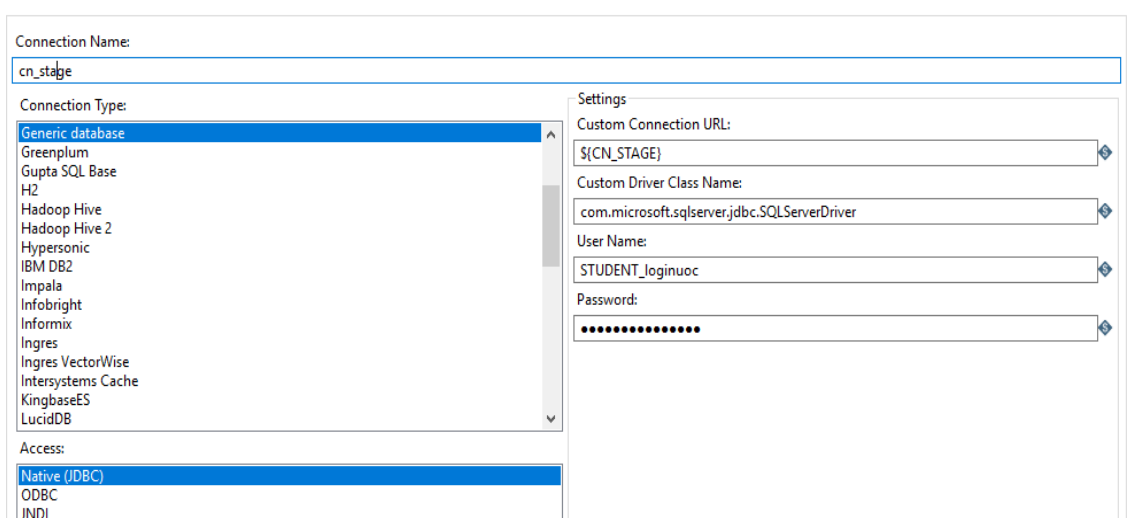
La referencia a las variables de entorno durante la implementación de los procesos se realiza mediante llaves, de esta manera: \${DIR_IN}, \${CN_STAGE}, \${CN_DW}.

3.3.2 Conexión a la base de datos SQL Server

Otro paso previo que se debe realizar es crear las conexiones a las bases de datos que se usan en todas las transformaciones y trabajos de los procesos de carga.

Se han definido dos conexiones diferentes, una para la base de datos del modelo multidimensional («BBDD») y otra para el área intermedia («STAGE»); de esta manera se diferenciará claramente su uso, aunque físicamente se refieran al mismo esquema de la base de datos.

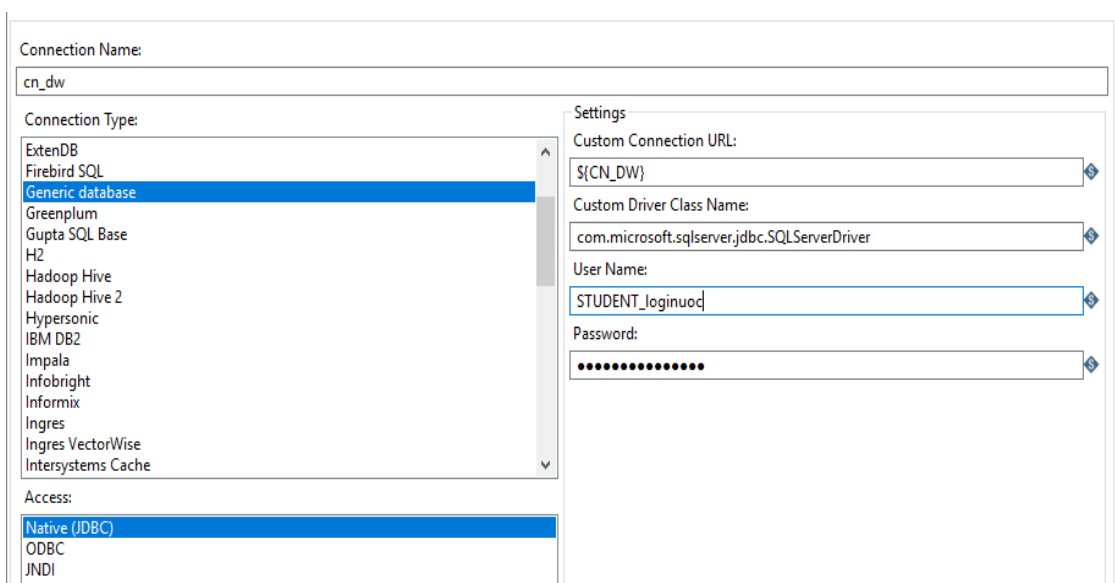
A la conexión al «STAGE» se le dará el nombre «cn_stage»:



The screenshot shows a configuration window for a new connection named 'cn_stage'. The 'Connection Type' is set to 'Generic database'. The 'Access' dropdown is set to 'Native (JDBC)'. The 'Settings' section on the right contains the following fields:

- Custom Connection URL:** \${CN_STAGE}
- Custom Driver Class Name:** com.microsoft.sqlserver.jdbc.SQLServerDriver
- User Name:** STUDENT_loginuoc
- Password:** (masked with dots)

A la conexión al DW se le dará el nombre «cn_dw»:



The screenshot shows a configuration window for a new connection named 'cn_dw'. The 'Connection Type' is set to 'Generic database'. The 'Access' dropdown is set to 'Native (JDBC)'. The 'Settings' section on the right contains the following fields:

- Custom Connection URL:** \${CN_DW}
- Custom Driver Class Name:** com.microsoft.sqlserver.jdbc.SQLServerDriver
- User Name:** STUDENT_loginuoc
- Password:** (masked with dots)

3.3.3 Bloque IN

Hay dos estrategias para plantear la carga del bloque IN:

- Minimizar las transformaciones sobre el origen de datos (campos calculados, etc.); esto se realizará en las transformaciones del bloque TR. El objetivo es disponer de una «copia rápida» de los orígenes de los datos normalizados.
- Utilizar la zona de maniobras (*staging area*) para realizar las transformaciones intermedias necesarias para facilitar y normalizar la carga de los bloques TR.

La opción a) es muy recomendable en los casos donde los orígenes de datos sean bases de datos operacionales. Esta estrategia minimizará el impacto sobre los sistemas de origen.

En esta práctica, dado que todos los orígenes de datos son ficheros externos «desconectados», no existe riesgo de penalizar a los sistemas origen y se optará por la segunda estrategia, para posteriormente, normalizar lo máximo posible la carga del modelo dimensional. Las fuentes externas se utilizarán para el descubrimiento de conocimiento realizando un análisis de los datos. No se utilizarán fuentes operacionales, en cuyo caso habría una etapa muy importante de preparación de las fuentes para dejarlas listas para su tratamiento con la herramienta ETL

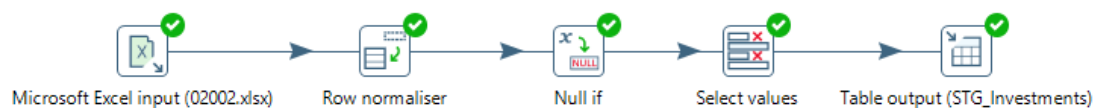
3.3.3.1 Transformación «IN_INVESTMENTS»

STG_Investments

El primer proceso que se desarrollará es la carga de la fuente de «02002.xlsx» a la tabla intermedia «STG_Investments» del *staging area*.

La transformación «IN_INVESTMENTS» contiene cinco operaciones: la lectura del fichero XLSX, la normalización de los registros, el control de los valores nulos, la selección de campos a insertar y la carga en la tabla intermedia «STG_Investments».

La transformación completa es la siguiente:



A continuación, se detallan los pasos más significativos:

<p>Microsoft Excel input (02002.xlsx)</p>	<p>Se selecciona el archivo XLSX de fuente de datos. Es necesario elegir correctamente el tipo. Además, es muy recomendable el uso de variables de entorno para parametrizar el directorio de carga.</p>
---	--

Spread sheet type (engine)	Excel 2007 XLSX (Apache POI)	
File or directory		
Regular Expression		
Exclude Regular Expression		
Password		
Selected files:	#	File/Directory
	1	\$(DIR_IN)\02002.xlsx
		Wildcard (RegExp)

Como el fichero contiene registros de cabecera, en la pestaña «Sheets», además de seleccionar la pestaña a cargar, se debe indicar la fila inicial (7).

Microsoft Excel input

Step name: Microsoft Excel input (02002.xlsx)

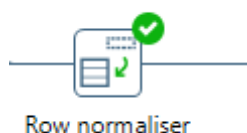
Files	Sheets	Content	Error Handling	Fields	Additional output fields
List of sheets to read					
#	Sheet name	Start row	Start column		
1	tabla-0	7	0		

Desde la pestaña «Fields» se obtienen los campos a cargar. En esta opción se puede seleccionar el tipo de datos para cada columna y eliminar espacios en blanco sobrantes. Para facilitar el proceso de carga, en el bloque IN es habitual seleccionar el tipo *string* para todos los campos. Posteriormente se dará formato numérico a los valores correspondientes.

Microsoft Excel input

Step name: Microsoft Excel input (02002.xlsx)

Files	Sheets	Content	Error Handling	Fields	Additional output fields	
#	Name	Type	Length	Precision	Trim type	Repeat
1	Periodo	String	-1	-1	both	N
2	Sector	String	-1	-1	both	N
3	Tipo	String	-1	-1	both	N
4	Ámbito	String	-1	-1	both	N
5	Andalucía	String	-1	-1	none	N
6	Aragón	String	-1	-1	none	N
7	Asturias, Principado de	String	-1	-1	none	N
8	Baleares, Illes	String	-1	-1	none	N



Los datos de las comunidades autónomas están dispuestos en columnas, es necesario normalizarlos para poder cargar el modelo dimensional.




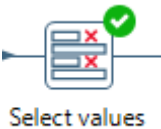


	 Row normaliser <div> Step name: <input type="text" value="Row normaliser"/> Type field: <input type="text" value="Comunidad_Autonomas"/> </div> <table border="1"> <thead> <tr> <th>#</th> <th>Fieldname</th> <th>Type</th> <th>new field</th> </tr> </thead> <tbody> <tr><td>1</td><td>Andalucía</td><td>Andalucía</td><td>Inversión</td></tr> <tr><td>2</td><td>Aragón</td><td>Aragón</td><td>Inversión</td></tr> <tr><td>3</td><td>Asturias, Principado de</td><td>Asturias, Principado de</td><td>Inversión</td></tr> <tr><td>4</td><td>Baleares, Illes</td><td>Baleares, Illes</td><td>Inversión</td></tr> <tr><td>5</td><td>Canarias</td><td>Canarias</td><td>Inversión</td></tr> <tr><td>6</td><td>Cantabria</td><td>Cantabria</td><td>Inversión</td></tr> <tr><td>7</td><td>Castilla y León</td><td>Castilla y León</td><td>Inversión</td></tr> <tr><td>8</td><td>Castilla-La Mancha</td><td>Castilla-La Mancha</td><td>Inversión</td></tr> <tr><td>9</td><td>Cataluña</td><td>Cataluña</td><td>Inversión</td></tr> </tbody> </table>	#	Fieldname	Type	new field	1	Andalucía	Andalucía	Inversión	2	Aragón	Aragón	Inversión	3	Asturias, Principado de	Asturias, Principado de	Inversión	4	Baleares, Illes	Baleares, Illes	Inversión	5	Canarias	Canarias	Inversión	6	Cantabria	Cantabria	Inversión	7	Castilla y León	Castilla y León	Inversión	8	Castilla-La Mancha	Castilla-La Mancha	Inversión	9	Cataluña	Cataluña	Inversión									
#	Fieldname	Type	new field																																															
1	Andalucía	Andalucía	Inversión																																															
2	Aragón	Aragón	Inversión																																															
3	Asturias, Principado de	Asturias, Principado de	Inversión																																															
4	Baleares, Illes	Baleares, Illes	Inversión																																															
5	Canarias	Canarias	Inversión																																															
6	Cantabria	Cantabria	Inversión																																															
7	Castilla y León	Castilla y León	Inversión																																															
8	Castilla-La Mancha	Castilla-La Mancha	Inversión																																															
9	Cataluña	Cataluña	Inversión																																															
 <p>Null if</p>	<p>En el campo «Inversión» («métrica») se observan algunos valores no numéricos que debemos corregir.</p>  Null if <div> Step name: <input type="text" value="Null if"/> </div> <table border="1"> <thead> <tr> <th>#</th> <th>Name</th> <th>Value to turn to NULL</th> </tr> </thead> <tbody> <tr><td>1</td><td>Inversión</td><td>..</td></tr> </tbody> </table> <div> <input type="button" value="Help"/> <input type="button" value="OK"/> <input type="button" value="Cancel"/> <input type="button" value="Get Fields"/> </div>	#	Name	Value to turn to NULL	1	Inversión	..																																											
#	Name	Value to turn to NULL																																																
1	Inversión	..																																																
 <p>Select values</p>	<p>Antes de realizar la carga es necesario seleccionar los campos del flujo de datos que se quieren insertar. Esta es una operación común en muchos ETLs y no será detallada en todos los procesos posteriores. Se puede observar que además es posible especificar algunos metadatos, como por ejemplo el tipo de dato.</p>  Select values <div> Step name: <input type="text" value="Select values"/> </div> <table border="1"> <thead> <tr> <th>#</th> <th>Fieldname</th> <th>Rename to</th> <th>Type</th> <th>Length</th> <th>Precision</th> <th>Binary to Normal?</th> </tr> </thead> <tbody> <tr><td>1</td><td>Periodo</td><td></td><td>Integer</td><td></td><td></td><td>N</td></tr> <tr><td>2</td><td>Sector</td><td></td><td>None</td><td></td><td></td><td>N</td></tr> <tr><td>3</td><td>Tipo</td><td></td><td>None</td><td></td><td></td><td>N</td></tr> <tr><td>4</td><td>Ámbito</td><td></td><td>None</td><td></td><td></td><td>N</td></tr> <tr><td>5</td><td>Comunidad_Autonomas</td><td></td><td>None</td><td></td><td></td><td>N</td></tr> <tr><td>6</td><td>Inversión</td><td></td><td>Number</td><td></td><td></td><td>N</td></tr> </tbody> </table>	#	Fieldname	Rename to	Type	Length	Precision	Binary to Normal?	1	Periodo		Integer			N	2	Sector		None			N	3	Tipo		None			N	4	Ámbito		None			N	5	Comunidad_Autonomas		None			N	6	Inversión		Number			N
#	Fieldname	Rename to	Type	Length	Precision	Binary to Normal?																																												
1	Periodo		Integer			N																																												
2	Sector		None			N																																												
3	Tipo		None			N																																												
4	Ámbito		None			N																																												
5	Comunidad_Autonomas		None			N																																												
6	Inversión		Number			N																																												
 <p>Table output (STG_Investments)</p>	<p>Finalmente, se selecciona la conexión «cn_stage» y la tabla de destino. En el bloque IN es posible seleccionar la opción «Truncate table», dado que el modelo físico se creó sin claves ni restricciones.</p>																																																	

Table output

Step name

Table output (STG_Investments)

Connection

cn_stage

Edit...

New...

Wizard...

Target schema

dbo

Browse...

Target table

STG_Investments

Browse...

Commit size

1000

Truncate table

☒

Ignore insert errors

☐

Specify database fields

☒

El último paso consiste en establecer la correspondencia (mapeo) entre los campos del flujo de datos y los campos de la tabla del área de maniobras.

Main options

Database fields

Fields to insert:

#	Table field	Stream field
1	periodo	Periodo
2	sector_economico	Sector
3	tipo_instalacion	Tipo
4	ambito_medioambiental	Ámbito
5	comunidad_autonoma	Comunidad_Autonoma
6	inversion	Inversión

Get fields

Enter field mapping

El resultado de la ejecución es el siguiente:

Execution Results

Logging Execution History Step Metrics Performance Graph Metrics Preview data

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	Microsoft Excel input (02002.xlsx)	0	0	154	154	0	0	0	0	Finished	2.3s	67
2	Row normaliser	0	154	2618	0	0	0	0	0	Finished	2.6s	994
3	Null if	0	2618	2618	0	0	0	0	0	Finished	2.7s	975
4	Select values	0	2618	2618	0	0	0	0	0	Finished	2.7s	958
5	Table output (STG_Investments)	0	2618	2618	0	2618	0	0	0	Finished	3.3s	802

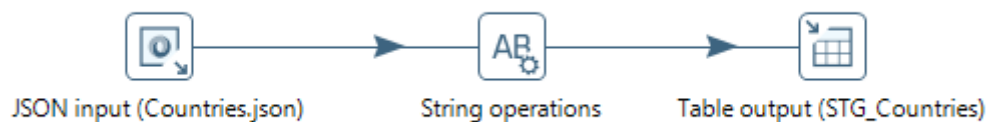
Como se observa en las métricas, se cargan los 2618 registros del fichero de entrada, una vez normalizados los datos.

3.3.3.2 Transformación «IN_COUNTRIES»


STG_Countries

Esta transformación es muy sencilla, únicamente consiste en obtener los valores del fichero JSON, realizar algunas normalizaciones de los campos de texto y cargar la tabla de destino.

La transformación completa es la siguiente:



A continuación, se detallan los pasos más significativos:



JSON input (Countries.json)

En primer lugar, se debe seleccionar el archivo JSON de origen de datos

JSON input

Step name:

File Content Fields Additional output fields

Source from field

Source is from a previous step: ☐

Select field:

Use field as file names: ☐

Read source as URL: ☐

Do not pass field downstream: ☐

File or directory:


Regular Expression:

Exclude Regular Expression:

Selected files:

#	File/Directory	Wildcard (RegExp)
1	\${DIR_IN}\Countries.json	


En la pestaña «Fields» se configurará el *path* de acceso a cada campo del JSON



JSON input

Step name

#	Name	Path	Type	Format
1	nombre	\$..countries[*].nombre	String	
2	name	\$..countries[*].name	String	
3	nom	\$..countries[*].nom	String	
4	iso2	\$..countries[*].iso2	String	
5	iso3	\$..countries[*].iso3	String	
6	phone_code	\$..countries[*].phone_code	String	



String operations

Step name

#	In stream field	Out stream field	Trim type	Lower/Upper	Padding	Pad char	Pad Length	InitCap
1	nombre		both	none	none			N
2	name		both	none	none			N
3	nom		both	none	none			N
4	iso2		both	none	none			N
5	iso3		both	none	none			N
6	phone_code		both	none	none			N




Table output (STG_Countries)

Step name

Connection

Target schema

Target table

Commit size

Truncate table ☒

Ignore insert errors ☐

Specify database fields ☒

#	Table field	Stream field
1	nombre	nombre
2	name	name
3	nom	nom
4	iso2	iso2
5	iso3	iso3
6	phone_code	phone_code

En este paso se normalizan los valores de tipo *string*. Por ejemplo, eliminando espacios en blanco sobrantes o pasando los valores a mayúsculas/minúsculas. Este es un paso muy habitual y no será detallada en transformaciones posteriores.

Finalmente se selecciona la conexión y tabla correspondiente del «área de *staging*» y se establece el mapeo de campos.

El resultado de la ejecución es el siguiente (246 registros)

Execution Results

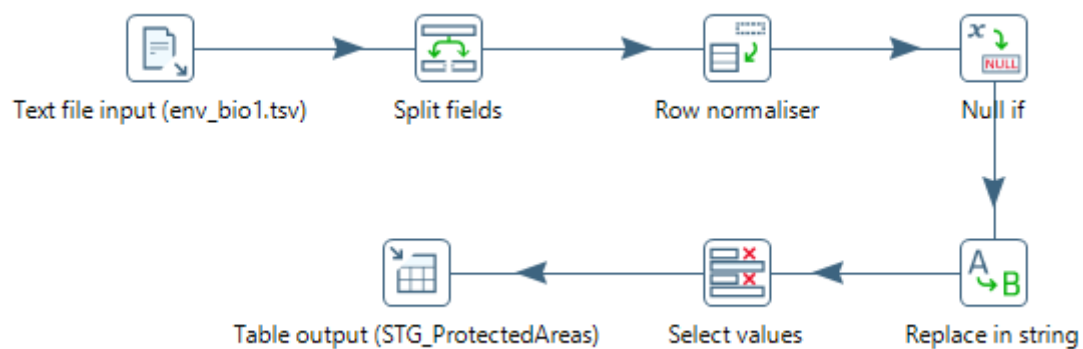
Execution Results												
Logging Execution History Step Metrics Performance Graph Metrics Preview data												
#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	JSON input (Countries.json)	0	0	246	246	0	0	0	0	Finished	0.1s	1,662
2	String operations	0	246	246	0	0	0	0	0	Finished	0.2s	1,000
3	Table output (STG_Countries)	0	246	246	0	246	0	0	0	Finished	0.3s	888

3.3.3.3 Transformación «IN_PROTECTED_AREA»


STG ProtectedAreas

Esta transformación se realiza en 7 pasos desde la extracción hasta la carga.

La transformación completa es la siguiente:



A continuación, se detallan los pasos más significativos:



Text file input (env_bio1.tsv)

Como es habitual, en el primer paso de la transformación se debe seleccionar el archivo de origen de datos y los campos correspondientes. En este caso se trata de un archivo de texto separado por tabuladores.

Text file input

Step name: Text file input (env_bio1.tsv)

File or directory:

Regular Expression:

Exclude Regular Expression:

Selected files:

#	File/Directory	Wildcard (RegExp)	Exclude wildcard
1			
2	\$(DIR_IN)\env_bio1.tsv		
3			

Step name: Text file input (env_bio1.tsv)

File | Content | Error Handling | Filters | Fields | Additional output fields

Filetype: CSV

Separator: Insert TAB

Enclosure: "

Escape: \

Text file input

Step name: Text file input (env_bio1.tsv)

File | Content | Error Handling | Filters | Fields | Additional output fields

#	Name	Type	Format	Position	Length	Precision	Currency	Decimal	Group	Null if	Default	Trim type
1	areaprot,geo,time	String			18		\$	-	-	-		none
2	2019	String			15	0	\$	-	-	-		both
3	2018	String			15	0	\$	-	-	-		both
4	2017	String			8		\$	-	-	-		both
5	2016	String			8		\$	-	-	-		both
6	2015	String			8		\$	-	-	-		both
7	2014	String			8		\$	-	-	-		both
8	2013	String			8		\$	-	-	-		both
9	2012	String			7		\$	-	-	-		both
10	2011	String			7		\$	-	-	-		both

Se observa que en el origen de datos hay dos campos concatenados que debemos separar: [areaprot] y [geto-time]. Para realizar esta operación se utilizará una transformación «Split fields». El separador de campo utilizado será «,».

Split fields

Step name: Split fields

Field to split: areaprot,geo,time

Delimiter: ,

Enclosure: "

Fields

#	New field	ID	Remove ID?	Type	Length	Precision	Format	Group	Decimal
1	areaprot	N		String					
2	geo-time	N		String					

Help OK Cancel

Es necesario normalizar las columnas (campo «Year») para adaptarlas al modelo dimensional.

Row normaliser

Step name: Row normaliser

Type field: Year

Fields

#	Fieldname	Type	new field
1	2019	2019	Value
2	2018	2018	Value
3	2017	2017	Value
4	2016	2016	Value
5	2015	2015	Value
6	2014	2014	Value


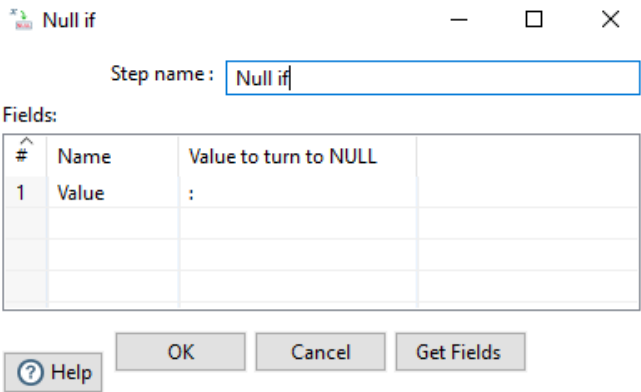

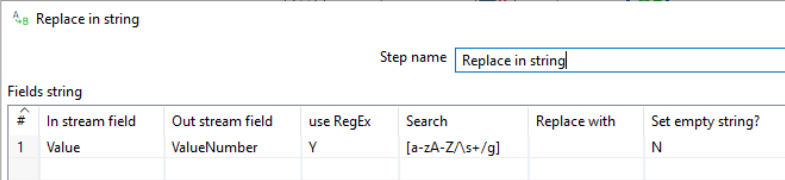

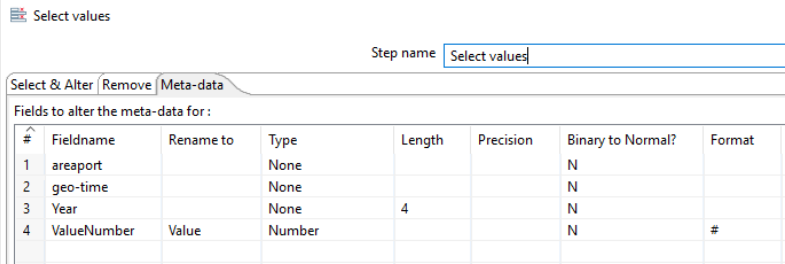

 <p>Null if</p>	<p>Se identifican y corrigen los valores no numéricos del campo «Value».</p> 
 <p>Replace in string</p>	<p>En esta transformación se utilizará una expresión regular para eliminar todos los caracteres no numéricos: [a-zA-Z/\s+/g]</p> 
 <p>Select values</p>	<p>Como se ha comentado anteriormente, «Select values» es una operación habitual. En este caso se observa la configuración de tipos, longitudes, formatos y nombres.</p> 
 <p>Table output (STG_ProtectedAreas)</p>	<p>La carga final se realiza una vez seleccionada la conexión, la tabla STG y el mapeo de campos.</p>

Table output

Step name: Table output (STG_ProtectedAreas)

Connection: cn_stage

Target schema: dbo

Target table: STG_ProtectedAreas

Commit size: 1000

Truncate table: ☒

Ignore insert errors: ☐

Specify database fields: ☒

Main options | Database fields

Fields to insert:

#	Table field	Stream field
1	areaprot	areaprot
2	geotime	geo-time
3	year	Year
4	value	Value

Get fields

Enter field mapping

El resultado de la ejecución es el siguiente (1080 registros):

Execution Results

Logging | Execution History | Step Metrics | Performance Graph | Metrics | Preview data

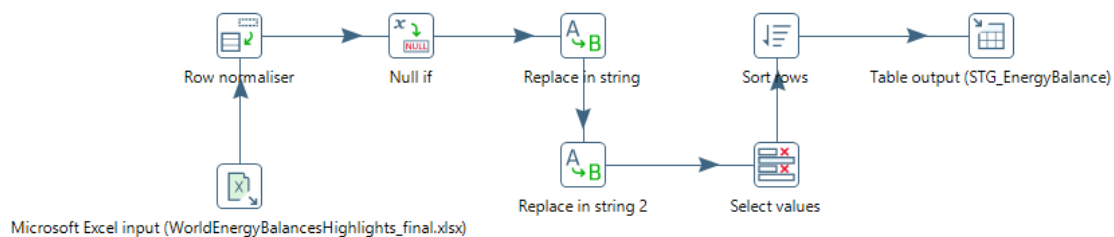
#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	Text file input (env_bio1.tsv)	0	0	120	121	0	1	0	0	Finished	0.1s	917
2	Split fields	0	120	120	0	0	0	0	0	Finished	0.2s	628
3	Row normaliser	0	120	1080	0	0	0	0	0	Finished	0.2s	4,696
4	Null if	0	1080	1080	0	0	0	0	0	Finished	0.3s	3,737
5	Replace in string	0	1080	1080	0	0	0	0	0	Finished	0.3s	3,612
6	Select values	0	1080	1080	0	0	0	0	0	Finished	0.3s	3,167
7	Table output (STG_ProtectedAreas)	0	1080	1080	0	1080	0	0	0	Finished	0.4s	2,903

3.3.3.4 Transformación «IN_ENERGY_BALANCE»

STG_EnergyBalance

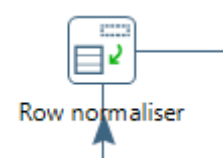
Para realizar esta transformación se utilizarán pasos habituales, algunos de ellos ya comentados en cargas anteriores.

La transformación completa es la siguiente:



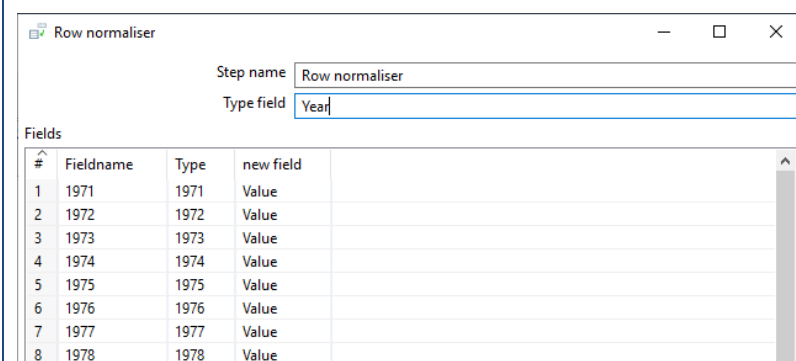
A continuación, se detallan los pasos más significativos:

En el primer paso de la transformación se selecciona el archivo XLSX de origen, la pestaña y los campos a cargar.



Row normaliser

Se observa que es necesario normalizar los datos de la serie 1971-2019. Para ello se creará el campo «Year».




Row normaliser

Step name: Row normaliser

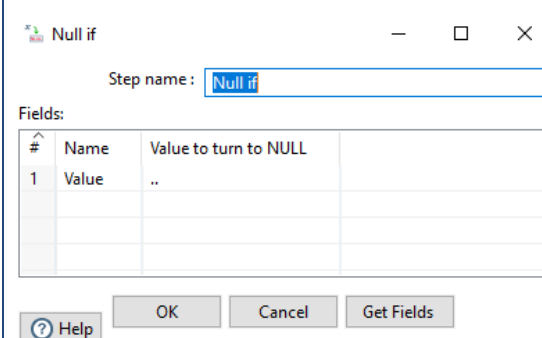
Type field: Year

#	Fieldname	Type	new field
1	1971	1971	Value
2	1972	1972	Value
3	1973	1973	Value
4	1974	1974	Value
5	1975	1975	Value
6	1976	1976	Value
7	1977	1977	Value
8	1978	1978	Value



Null if

Se corrigen valores incorrectos o con problemas de calidad del dato.

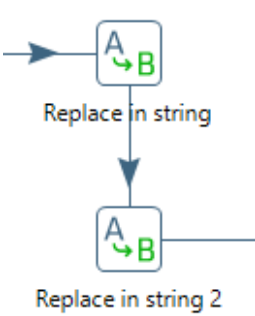


Null if

Step name: Null if

#	Name	Value to turn to NULL
1	Value	..

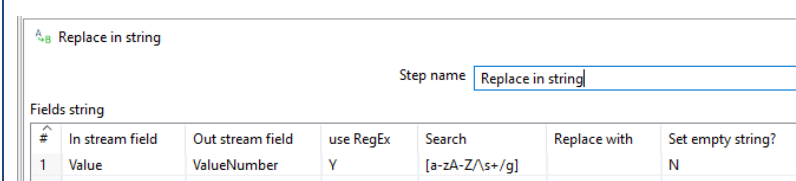
Help OK Cancel Get Fields



Replace in string

Replace in string 2

A continuación, se enlazarán dos transformaciones de texto para eliminar caracteres incorrectos. El primero haciendo uso de una expresión regular: [a-zA-Z/\s+/g]

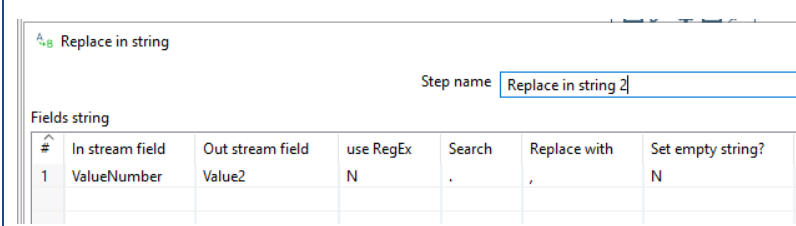


Replace in string

Step name: Replace in string

#	In stream field	Out stream field	use RegEx	Search	Replace with	Set empty string?
1	Value	ValueNumber	Y	[a-zA-Z/\s+/g]		N

Y en el segundo substituyendo «puntos» por «comas». Esto permitirá tratar el campo como un valor numérico agregable.



Replace in string

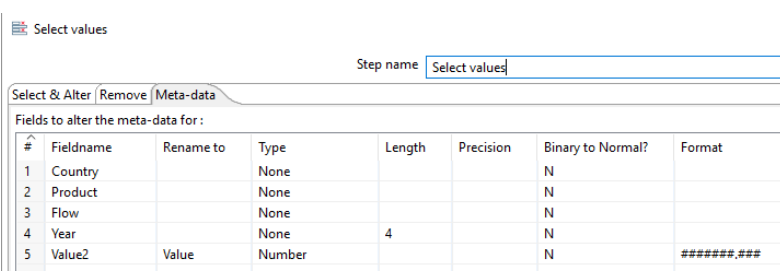
Step name: Replace in string 2

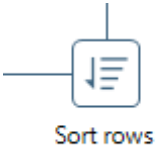
#	In stream field	Out stream field	use RegEx	Search	Replace with	Set empty string?
1	ValueNumber	Value2	N	.	,	N



Select values

Se seleccionan los campos a cargar aplicando los meta-datos correspondientes.





Sort rows

En esta transformación se considera necesario ordenar los datos directamente en el flujo de datos.




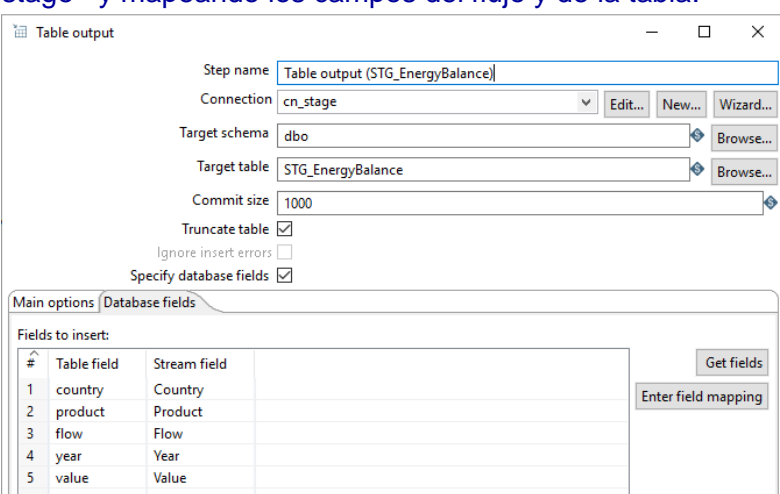


Table output (STG_EnergyBalance)

En el paso final de carga de la tabla, se establece la configuración habitual, seleccionando la conexión «cn stage» y mapeando los campos del flujo y de la tabla.



El resultado de la ejecución es el siguiente (296.352 registros):

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	Microsoft Excel input (WorldEnergyBalancesHighlights_final.xlsx)	0	0	6048	6048	0	0	0	0	Finished	18.3s	331
2	Row normaliser	0	6048	296352	0	0	0	0	0	Finished	18.3s	16,202
3	Null if	0	296352	296352	0	0	0	0	0	Finished	18.3s	16,191
4	Replace in string	0	296352	296352	0	0	0	0	0	Finished	18.3s	16,181
5	Replace in string 2	0	296352	296352	0	0	0	0	0	Finished	18.3s	16,176
6	Select values	0	296352	296352	0	0	0	0	0	Finished	18.3s	16,173
7	Sort rows	0	296352	296352	0	0	0	0	0	Finished	25.5s	11,621
8	Table output (STG_EnergyBalance)	0	296352	296352	0	296352	0	0	0	Finished	25.8s	11,502

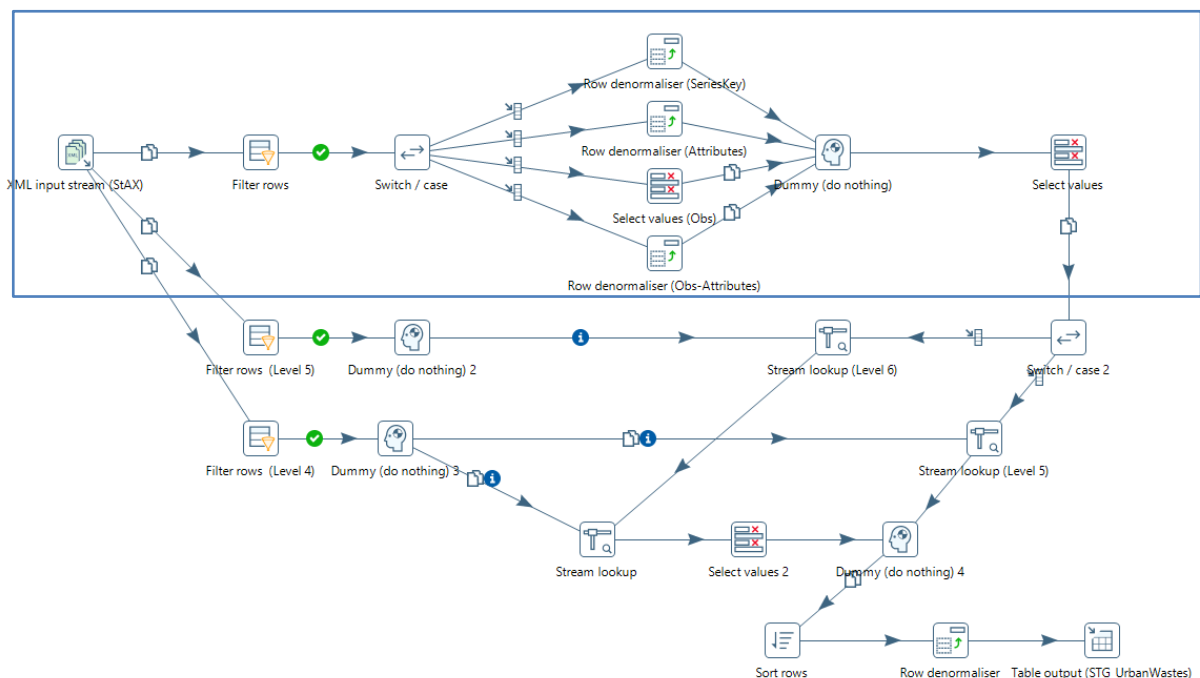
3.3.3.5 Transformación «IN_URBAN_WASTES»

STG_UrbanWastes

Probablemente esta sea la transformación más compleja del proceso. En el ejemplo se transforma el XML de origen completamente para cargarlo en una tabla STG. Sin embargo, sería posible simplificar el proceso cargando únicamente los nodos necesarios para la explotación de datos requerida. El hecho de tener que desnormalizar los datos para ser agrupados requiere identificar el nodo padre de todos los elementos, para posteriormente agruparlos y normalizarlos nuevamente.

La primera parte de la transformación (cuadro en color azul) corresponde al tratamiento del XML. La segunda parte únicamente realiza los *lookups* necesarios para obtener el nodo por el que se agruparan los datos (campo *group_id*).

La transformación completa es la siguiente:



A continuación, se detallan los pasos más significativos:

Este componente de entrada permite analizar un fichero XML y detectar su estructura interna.

XML input stream (StAX)

Step name: XML input stream (StAX)

Filename: \${DIR_IN}\DataGeneric.xml Browse...

Source is from a previous step: ☐

Source field name:

Add filename to result?: ☐

Skip (Elements/Attributes):

Limit (Elements/Attributes):

Default String Length:

Encoding:

Add Namespace information?: ☐

Trim strings?: ☒

Include filename in output?: ☐ Fieldname: xml_filename

Row number in output?: ☐ Fieldname: xml_row_number

XML data type (numeric) in output?: ☐ Fieldname: xml_data_type_numeric

XML data type (description) in output?: ☒ Fieldname: xml_data_type_description

XML location line in output?: ☐ Fieldname: xml_location_line

XML location column in output?: ☐ Fieldname: xml_location_column

XML element ID in output?: ☒ Fieldname: xml_element_id

XML parent element ID in output?: ☒ Fieldname: xml_parent_element_id

XML element level in output?: ☒ Fieldname: xml_element_level

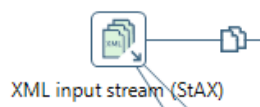
XML path in output?: ☒ Fieldname: xml_path

XML parent path in output?: ☒ Fieldname: xml_parent_path

XML data name in output?: ☒ Fieldname: xml_data_name

XML data value in output?: ☒ Fieldname: xml_data_value

Help OK Preview Cancel



XML input stream (StAX)

Tras realizar un «Preview» se pueden observar los nodos, atributos, valores, *path*, etc. del XML cargado

Examine preview data

Rows of step XML input stream (StAX) (1000 rows)

#	xml_data_type_description	xml_element_id	xml_parent_element_id	xml_element_level	xml_path	xml_parent_path	xml_data_name	xml_data_value
1	START_DOCUMENT	0	<null>	0	<null>	<null>	<null>	<null>
2	START_ELEMENT	1	0	1	/MessageGroup	<null>	MessageGroup	<null>
3	ATTRIBUTE	1	0	1	/MessageGroup	<null>	schemalocation	http://www.SDMG.org
4	START_ELEMENT	2	1	2	/MessageGroup/Header	/MessageGroup	Header	<null>
5	START_ELEMENT	3	2	3	/MessageGroup/Header/ID	/MessageGroup/Header	ID	<null>
6	CHARACTERS	3	2	3	/MessageGroup/Header/ID	/MessageGroup/Header	ID	none
7	END_ELEMENT	3	2	3	/MessageGroup/Header/ID	/MessageGroup/Header	ID	<null>
8	START_ELEMENT	4	2	3	/MessageGroup/Header/Test	/MessageGroup/Header	Test	<null>
9	CHARACTERS	4	2	3	/MessageGroup/Header/Test	/MessageGroup/Header	Test	false
10	END_ELEMENT	4	2	3	/MessageGroup/Header/Test	/MessageGroup/Header	Test	<null>
11	START_ELEMENT	5	2	3	/MessageGroup/Header/Truncated	/MessageGroup/Header	Truncated	<null>
12	CHARACTERS	5	2	3	/MessageGroup/Header/Truncated	/MessageGroup/Header	Truncated	false
13	END_ELEMENT	5	2	3	/MessageGroup/Header/Truncated	/MessageGroup/Header	Truncated	<null>
14	START_ELEMENT	6	2	3	/MessageGroup/Header/Prepared	/MessageGroup/Header	Prepared	<null>
15	CHARACTERS	6	2	3	/MessageGroup/Header/Prepared	/MessageGroup/Header	Prepared	2021-03-15T22:21:16
16	END_ELEMENT	6	2	3	/MessageGroup/Header/Prepared	/MessageGroup/Header	Prepared	<null>
17	START_ELEMENT	7	2	3	/MessageGroup/Header/Sender	/MessageGroup/Header	Sender	<null>
18	ATTRIBUTE	7	2	3	/MessageGroup/Header/Sender	/MessageGroup/Header	id	OECD
19	START_ELEMENT	8	7	4	/MessageGroup/Header/Sender/Name	/MessageGroup/Header/Sender	Name	<null>
20	ATTRIBUTE	8	7	4	/MessageGroup/Header/Sender/Name	/MessageGroup/Header/Sender	lang	en
21	CHARACTERS	8	7	4	/MessageGroup/Header/Sender/Name	/MessageGroup/Header/Sender	Name	Organisation for Econ



Filter rows

Para reducir la cantidad de nodos a tratar, se filtrarán los nodos en función del *path*, valor o nivel dentro del XML.

Switch / case

Una vez filtrados los datos, se creará un flujo de transformación independiente para tratar cada tipo de nodo.

Se puede observar que tres de los flujos requieren la desnormalización de los datos, mientras que en uno únicamente es necesario seleccionar los campos deseados.

Filter rows

Step name: Filter rows

Send 'true' data to step: Switch / case

Send 'false' data to step:

The condition:

```
(
  AND
  (
    xml_parent_path STARTS WITH [/MessageGroup/DataSet/Series]
    AND
    xml_data_value IS NOT NULL
  )
  OR
  (
    AND
    (
      xml_parent_path STARTS WITH [/MessageGroup/DataSet/Series]
      AND
      xml_element_level = [4]
    )
  )
)
```

Switch / case

Step name: Switch / case

Field name to switch: xml_parent_path

Use string contains comparison: ☐

Case value data type: None

Case value conversion mask:

Case value decimal symbol:

Case value grouping symbol:

#	Value	Target step
1	/MessageGroup/DataSet/Series/SeriesKey	Row denormaliser (SeriesKey)
2	/MessageGroup/DataSet/Series/Attributes	Row denormaliser (Attributes)
3	/MessageGroup/DataSet/Series/Obs	Select values (Obs)
4	/MessageGroup/DataSet/Series/Obs/Attributes	Row denormaliser (Obs-Attributes)

Row denormaliser (SeriesKey)

Row denormaliser (Attributes)

Row denormaliser (Obs-Attributes)

A modo de ejemplo se mostrará la configuración de desnormalización de los nodos «SeriesKey», entendiendo que para «Attributes» y «Obs-Attributes», el proceso es equivalente.

Row denormaliser

Step name: Row denormaliser (SeriesKey)


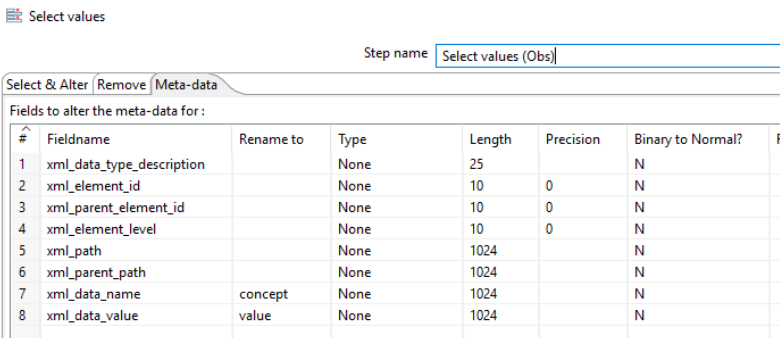

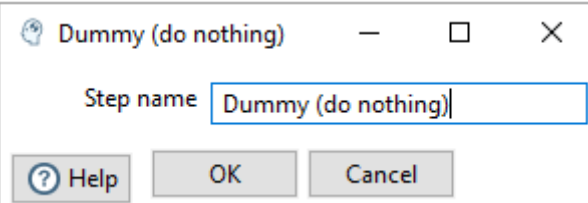
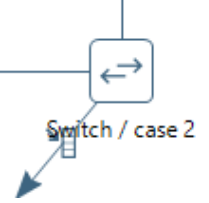
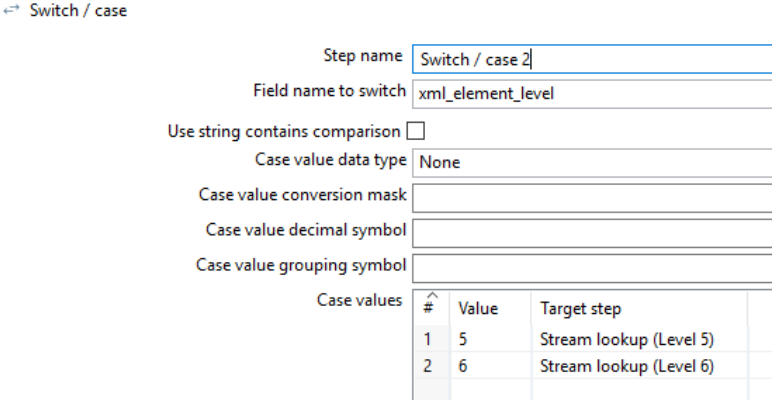
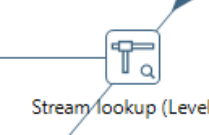
The key field: xml_data_name

The fields that make up the grouping:

#	Group field
1	xml_element_id

Target fields:

#	Target fieldname	Value fieldname	Key value	Type	Format	Length	Precision	Currency
1	concept	xml_data_value	concept	String				
2	value	xml_data_value	value	String				

 <p>Select values (Obs)</p>	<p>Para los nodos «Obs», simplemente es necesario seleccionar los campos a tratar del XML.</p>  <table><tr><th>#</th><th>Fieldname</th><th>Rename to</th><th>Type</th><th>Length</th><th>Precision</th><th>Binary to Normal?</th><th>F</th></tr><tr><td>1</td><td>xml_data_type_description</td><td></td><td>None</td><td>25</td><td></td><td>N</td><td></td></tr><tr><td>2</td><td>xml_element_id</td><td></td><td>None</td><td>10</td><td>0</td><td>N</td><td></td></tr><tr><td>3</td><td>xml_parent_element_id</td><td></td><td>None</td><td>10</td><td>0</td><td>N</td><td></td></tr><tr><td>4</td><td>xml_element_level</td><td></td><td>None</td><td>10</td><td>0</td><td>N</td><td></td></tr><tr><td>5</td><td>xml_path</td><td></td><td>None</td><td>1024</td><td></td><td>N</td><td></td></tr><tr><td>6</td><td>xml_parent_path</td><td></td><td>None</td><td>1024</td><td></td><td>N</td><td></td></tr><tr><td>7</td><td>xml_data_name</td><td>concept</td><td>None</td><td>1024</td><td></td><td>N</td><td></td></tr><tr><td>8</td><td>xml_data_value</td><td>value</td><td>None</td><td>1024</td><td></td><td>N</td><td></td></tr></table>	#	Fieldname	Rename to	Type	Length	Precision	Binary to Normal?	F	1	xml_data_type_description		None	25		N		2	xml_element_id		None	10	0	N		3	xml_parent_element_id		None	10	0	N		4	xml_element_level		None	10	0	N		5	xml_path		None	1024		N		6	xml_parent_path		None	1024		N		7	xml_data_name	concept	None	1024		N		8	xml_data_value	value	None	1024		N	
#	Fieldname	Rename to	Type	Length	Precision	Binary to Normal?	F																																																																		
1	xml_data_type_description		None	25		N																																																																			
2	xml_element_id		None	10	0	N																																																																			
3	xml_parent_element_id		None	10	0	N																																																																			
4	xml_element_level		None	10	0	N																																																																			
5	xml_path		None	1024		N																																																																			
6	xml_parent_path		None	1024		N																																																																			
7	xml_data_name	concept	None	1024		N																																																																			
8	xml_data_value	value	None	1024		N																																																																			
 <p>Dummy (do nothing)</p>	<p>Una transformación «Dummy» en realidad no hace nada, pero es muy útil para unir flujos de datos (UNION)</p> 																																																																								
 <p>Switch / case 2</p>	<p>El siguiente paso consiste en dividir nuevamente el flujo de datos para identificar y obtener los nodos padre de cada nivel del XML.</p>  <table><tr><th>#</th><th>Value</th><th>Target step</th></tr><tr><td>1</td><td>5</td><td>Stream lookup (Level 5)</td></tr><tr><td>2</td><td>6</td><td>Stream lookup (Level 6)</td></tr></table>	#	Value	Target step	1	5	Stream lookup (Level 5)	2	6	Stream lookup (Level 6)																																																															
#	Value	Target step																																																																							
1	5	Stream lookup (Level 5)																																																																							
2	6	Stream lookup (Level 6)																																																																							
 <p>Stream lookup (Level 5)</p>	<p>Se mostrará el proceso para los nodos de nivel 5. En el caso de los nodos de nivel 6 el procedimiento es equivalente con la diferencia de que es necesario realizar dos <i>lookup</i> para obtener el <i>id</i> del agrupador.</p>																																																																								

Stream lookup (Level 6)

Stream lookup

Step name

Lookup step

The key(s) to look up the value(s):

#	Field	LookupField
1	xml_parent_element_id	xml_element_id

Specify the fields to retrieve :

#	Field	New name	Default	Type
1	xml_parent_element_id	group_id		Integer

Preserve memory (costs CPU) ☒

Key and value are exactly one integer field ☐

Use sorted list (i.s.o. hashtable) ☐

Filter rows (Level 5)

Filter rows (Level 4)

Para realizar el *lookup* anterior es necesario obtener los nodos de nivel superior. Para ello se filtrará el flujo de datos original (Salida de StAX).

Filter rows

Step name

Send 'true' data to step:

Send 'false' data to step:

The condition:

☐

```
(  
    AND  
    xml_parent_path STARTS WITH [/MessageGroup/DataSet/Series]  
    xml_element_level = [4]  
)
```

Sort rows

Tras realizar todos los *lookups* y unir los flujos con un paso «Dummy», se ordenarán los resultados para la correcta desnormalización posterior.

A diagram showing a data flow into a 'Row denormaliser' block. The block is represented by a square icon with a green arrow pointing upwards and a blue arrow pointing downwards. The text 'Row denormaliser' is written below the icon.

Sort rows

Step name: Sort rows

Sort directory: %%java.io.tmpdir%%

TMP-file prefix: out

Sort size (rows in memory): 1000000

Free memory threshold (in %):

Compress TMP Files? ☐

Only pass unique rows? (verifies keys only) ☐

Fields:

#	Fieldname	Ascending	Case sensitive compare?	Sort based on current locale?
1	group_id	Y	N	N
2	xml_parent_element_id	Y	N	N

Se identifican cada uno de los campos a tratar. Se puede observar que la agrupación se realiza por *xml_parent_element_id* y por *group_id*, que es el campo calculado que hemos obtenido en la transformación.

A diagram showing a data flow into a 'Row denormaliser' block. The block is represented by a square icon with a green arrow pointing upwards and a blue arrow pointing downwards. The text 'Row denormaliser' is written below the icon.

Row denormaliser

Step name: Row denormaliser

The key field: concept

The fields that make up the grouping:

#	Group field
1	group_id
2	xml_parent_element_id

Target fields:

#	Target fieldname	Value fieldname	Key value	Type	Format	Length	Precision
1	COU	value	COU	String			
2	VAR	value	VAR	String			
3	TIME_FORMAT	value	TIME_FORMAT	String			
4	UNIT	value	UNIT	String			
5	POWERCODE	value	POWERCODE	String			
6	TIME	value	Time	String			
7	VALUE	value	value	String			
8	OBS_STATUS	value	OBS_STATUS	String			

A diagram showing a data flow into a 'Table output (STG_UrbanWastes)' block. The block is represented by a square icon with a green arrow pointing upwards and a blue arrow pointing downwards. The text 'Table output (STG_UrbanWastes)' is written below the icon.

Para finalizar el ETL, se carga el resultado obtenido en la tabla STG_UrbanWastes. En este caso los resultados se insertan desnormalizados. El proceso de normalización se realizará en el bloque TR correspondiente.

Table output

Step name

Table output (STG_UrbanWastes)

Connection

cn_stage

Edit...

New...

Wizard...

Target schema

dbo

Browse...

Target table

STG_UrbanWastes

Browse...

Commit size

1000

Truncate table

☒

Ignore insert errors

☐

Specify database fields

☒

Main options

Database fields

Fields to insert:

#	Table field	Stream field
1	group_id	group_id
2	COU	COU
3	VAR	VAR
4	TIME_FORMAT	TIME_FORMAT
5	UNIT	UNIT
6	POWERCODE	POWERCODE
7	TIME	TIME
8	VALUE	VALUE
9	OBS_STATUS	OBS_STATUS

Get fields

Enter field mapping

El resultado de la ejecución es el siguiente:

Execution Results

<div> <div>Logging</div> <div>Execution History</div> <div>Step Metrics</div> <div>Performance Graph</div> <div>Metrics</div> <div>Preview data</div> </div>													
#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)	input/output
1	XML input stream (StAX)	0	0	640569	175699	0	0	0	0	Finished	3.6s	179,481	-
2	Filter rows	0	213523	104096	0	0	0	0	0	Finished	3.6s	59,643	-
3	Filter rows (Level 4)	0	213523	45498	0	0	0	0	0	Finished	3.6s	59,627	-
4	Switch / case	0	104096	58598	0	0	0	0	0	Finished	4.3s	24,180	-
5	Row denormaliser (Obs-Attributes)	0	7226	3613	0	0	0	0	0	Finished	4.3s	1,676	-
6	Row denormaliser (SeriesKey)	0	3940	1970	0	0	0	0	0	Finished	4.3s	914	-
7	Filter rows (Level 5)	0	213523	3613	0	0	0	0	0	Finished	3.6s	59,627	-
8	Select values (Obs)	0	41558	41558	0	0	0	0	0	Finished	4.3s	9,640	-
9	Row denormaliser (Attributes)	0	5874	2937	0	0	0	0	0	Finished	4.3s	1,364	-
10	Dummy (do nothing)	0	50078	50078	0	0	0	0	0	Finished	4.4s	11,371	-
11	Select values	0	50078	50078	0	0	0	0	0	Finished	5.0s	10,078	-
12	Dummy (do nothing) 2	0	3613	3613	0	0	0	0	0	Finished	3.6s	1,007	-
13	Switch / case 2	0	50078	50078	0	0	0	0	0	Finished	5.0s	9,980	-
14	Dummy (do nothing) 3	0	45498	90996	0	0	0	0	0	Finished	3.6s	25,375	-
15	Stream lookup (Level 5)	0	91963	46465	0	0	0	0	0	Finished	5.0s	18,214	-
16	Stream lookup (Level 6)	0	7226	3613	0	0	0	0	0	Finished	5.0s	1,439	-
17	Stream lookup	0	49111	3613	0	0	0	0	0	Finished	5.0s	9,731	-
18	Select values 2	0	3613	3613	0	0	0	0	0	Finished	5.1s	715	-
19	Dummy (do nothing) 4	0	50078	50078	0	0	0	0	0	Finished	5.1s	9,893	-
20	Sort rows	0	50078	50078	0	0	0	0	0	Finished	5.4s	9,332	-
21	Row denormaliser	0	50078	22749	0	0	0	0	0	Finished	5.5s	9,138	-
22	Table output (STG_UrbanWastes)	0	22749	22749	0	22749	0	0	0	Finished	5.8s	3,931	-

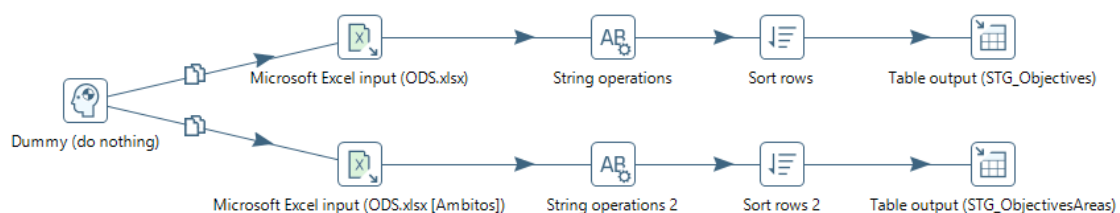
3.3.3.6 Transformación «IN_OBJECTIVES»

STG_Objectives

STG_ObjectivesAreas

Este bloque es simple de implementar, pero tiene la particularidad que, a partir de un único origen de datos, se cargarán dos tablas STG. El procedimiento es el mismo en los dos casos, únicamente varía la pestaña de datos seleccionada en el XLSX de origen.

La transformación completa es la siguiente:



A continuación, se detallan los pasos más significativos:

Microsoft Excel input (ODS.xlsx)

Microsoft Excel input (ODS.xlsx [Ambitos])

Se selecciona el archivo de origen (ODS.xlsx), la pestaña a cargar «Sheets» y la configuración de campos.

Step name: Microsoft Excel input (ODS.xlsx)

Files Sheets Content Error Handling Fields Additional output fields

Spread sheet type (engine): Excel 2007 XLSX (Apache POI)

File or directory:

Regular Expression:

Exclude Regular Expression:

Password:

Selected files:

#	File/Directory	Wildcard (RegExp)
1	\$(DIR_IN)\ODS.xlsx	

Step name: Microsoft Excel input (ODS.xlsx)


Files Sheets Content Error Handling Fields Additional output fields

#	Name	Type	Length	Precision	Trim type	Repeat	Format	Currency
1	Objetivo	Number	-1	-1	none	N	#	
2	Nombre	String	-1	-1	none	N		
3	Descripción	String	-1	-1	none	N		


Step name: Microsoft Excel input (ODS.xlsx [Ambitos])

Files Sheets Content Error Handling Fields Additional output fields

#	Name	Type	Length	Precision	Trim type	Repeat	Format	Currency
1	Codigo	String	-1	-1	none	N		
2	Ambito/VAR/Flow	String	-1	-1	none	N		
3	ODS principal	Number	-1	-1	none	N		



String operations



String operations 2

Se aplican las transformaciones típicas en el tratamiento de textos.

String operations

Step name:

The fields to process:


#	In stream field	Out stream field	Trim type	Lower/Upper	Padding	Pad char	Pad Length	InitCap
1	Nombre		both	upper	none			N
2	Descripción		both	none	none			N

String operations


Step name:

The fields to process:

#	In stream field	Out stream field	Trim type	Lower/Upper	Padding	Pad char	Pad Length	InitCap
1	Código		both	upper	none			N
2	Ambito/VAR/Flow		both		none			N



Sort rows



Sort rows 2

Se ordenan los resultados:

Sort rows

Step name:

Sort directory:

TMP-file prefix:

Sort size (rows in memory):

Free memory threshold (in %):

Compress TMP Files? ☐

Only pass unique rows? (verifies keys only) ☐

Fields:

#	Fieldname	Ascending	Case sensitive compare?	Sort based on current locale?
1	Objetivo	Y	N	N

Sort rows

Step name:

Sort directory:

TMP-file prefix:

Sort size (rows in memory):

Free memory threshold (in %):


Compress TMP Files? ☐

Only pass unique rows? (verifies keys only) ☐

Fields:

#	Fieldname	Ascending	Case sensitive compare?	Sort based on current locale?
1	ODS principal	Y	N	N

Se cargan las tablas destino del STG:


 Table output

Step name
 Connection
 Target schema
 Target table
 Commit size
 Truncate table ☒
 Ignore insert errors ☐
 Specify database fields ☒

Main options Database fields

Fields to insert:

#	Table field	Stream field
1	Objetivo	Objetivo
2	Nombre	Nombre
3	Descripcion	Descripción

 Table output

Step name
 Connection
 Target schema
 Target table
 Commit size
 Truncate table ☒
 Ignore insert errors ☐
 Specify database fields ☒

Main options Database fields

Fields to insert:

#	Table field	Stream field
1	Codigo	Codigo
2	Area	Ambito/VAR/Flow
3	ODS_Principal	ODS principal

El resultado de la ejecución es el siguiente:

Execution Results

Logging Execution History Step Metrics Performance Graph Metrics Preview data

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	Dummy (do nothing)	0	0	0	0	0	0	0	0	Finished	0.0s	0
2	Microsoft Excel input (ODS.xlsx)	0	0	17	17	0	0	0	0	Finished	0.0s	378
3	Microsoft Excel input (ODS.xlsx [Ambitos])	0	0	50	50	0	0	0	0	Finished	0.1s	685
4	String operations	0	17	17	0	0	0	0	0	Finished	0.1s	181
5	String operations 2	0	50	50	0	0	0	0	0	Finished	0.1s	532
6	Sort rows	0	17	17	0	0	0	0	0	Finished	0.1s	173
7	Table output (STG_Objectives)	0	17	17	0	17	0	0	0	Finished	0.1s	139
8	Sort rows 2	0	50	50	0	0	0	0	0	Finished	0.1s	505
9	Table output (STG_ObjectivesAreas)	0	50	50	0	50	0	0	0	Finished	0.1s	450

3.3.4 Bloque TR

El bloque TR contiene los procesos de ETL que se encargan de la carga inicial de datos desde las tablas intermedias pobladas con los procesos del bloque IN, al modelo multidimensional del almacén diseñado, compuesto por dimensiones y tablas de hechos.

Este bloque se divide, a su vez, en dos subbloques: por un lado, los procesos para la carga de dimensiones y, por el otro, los procesos para la carga de tablas de hechos. Esta división permite el retroceso de dichos procesos en caso de error y un mejor entendimiento de la implementación de los procesos.

Debido a que en Spoon antes de realizar cualquier tipo de unión o intersección de dos flujos de datos es necesario ordenarlos por el mismo campo clave, en esta solución se obviarán las transformaciones básicas de ordenación.

3.3.5 Bloque TR_DIM

Este bloque contiene las transformaciones para la carga inicial de las dimensiones al almacén desde las tablas intermedias «IN_ del staging area».

Se tendrá en cuenta que en una carga inicial pueden ejecutarse las transformaciones de carga de dimensiones las veces que sean necesarias.

3.3.5.1 TR_DIM_Date

En esta transformación se cargará la dimensión «Date». La carga de estas dimensiones es algo diferente a la carga de las dimensiones de datos. Hay diferentes opciones para cargar estas tablas de tiempo; en esta solución, dado que se utilizará una dimensión temporal sencilla, se creará un *script* SQL para generar todos los registros necesarios.

Es necesario indicar manualmente una fecha de inicio, la fecha de fin será el día que se ejecute el *script* SQL. La fecha de inicio dependerá de los datos disponibles, en el caso que nos ocupa cargaremos desde el 01/01/1970



Execute SQL script (Date)

```

DECLARE @FechaInicio datetime
DECLARE @FechaFin datetime
DECLARE @pk int
SET @FechaInicio = '01/01/1970'
SET @FechaFin = GETDATE()
SET @pk=0

-- Declaración y establecimiento de fecha ciclo
DECLARE @FechaCiclo datetime
SET @FechaCiclo = @FechaInicio

-- Bucle hasta fecha fin
WHILE @FechaCiclo <= @FechaFin
BEGIN
    SET @pk=@pk+1

    -- Insertar un registro en la dimensión Fecha
    INSERT INTO DIM_Date VALUES (
        --@PK,
        cast(cast( Year(@FechaCiclo) as varchar(4))+
            right('0'+cast(Month(@FechaCiclo) as varchar(2)),2)+
            right('0'+cast(Day(@FechaCiclo) as varchar(2)),2) as int),
        Year(@FechaCiclo),
        Month(@FechaCiclo),
        Day(@FechaCiclo),
        @FechaCiclo
    )

    -- Incrementar la FechaCiclo en un día
    SET @FechaCiclo = DateAdd(d, 1, @FechaCiclo)
END

```

El resultado de la ejecución es el siguiente:

Execution Results

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)	input/output
1	Execute SQL script (Date)	0	0	0	0	0	0	0	0	Idle	0.0s	-	-

Se puede observar que la salida del ETL es de cero registros. Esto se debe a que el propio *script* realiza un «INSERT» en la dimensión DIM_Dates y la transformación no tiene flujo de salida. Si consultamos directamente la tabla, se observa que se han cargado 18.758 registros correctamente (el número de registros dependerá del día de ejecución del *script*).

<pre>select * from DIM_Date</pre>					
100 %					
Results Messages					
	pk_date	date_year	date_month	date_day	date_date
1	19700101	1970	1	1	1970-01-01 00:00:00.000
2	19700102	1970	1	2	1970-01-02 00:00:00.000
3	19700103	1970	1	3	1970-01-03 00:00:00.000
4	19700104	1970	1	4	1970-01-04 00:00:00.000

<pre>select COUNT(*) as N from DIM_Date</pre>	
100 %	
Results Messages	
	N
1	18758



3.3.5.2 TR_DIM_Region

En general, el proceso de carga de las dimensiones suele ser una transformación sencilla. A pesar de esto hay que tener en cuenta los controles de calidad del dato habituales, la creación de las claves primarias y la inserción de registros «ficticios» de tipo NA (*not available*) para poder relacionar los registros de las tablas de hechos sin una correspondencia con alguna de las dimensiones.

La transformación completa es la siguiente:



A continuación, se detallan los pasos más significativos:

 Table input (Regions)	<p>En la carga de estos datos, se debe configurar la conexión al <i>staging area</i>.</p> <div> <div>Table input</div> <div> <div>Step name</div> <div>Table input (Regions)</div> </div> <div> <div>Connection</div> <div>cn_stage</div> </div> </div> <pre> SQL select ISNULL(A.comunidad_autonoma, 'NA') AS comunidad_autonoma, B.iso2, B.iso3, B.name from (select distinct 'ES' AS iso2, comunidad_autonoma from [dbo].[STG_Investments] union select 'ES', 'NA') A full outer join (select iso2, iso3, name from [STG_Countries] union select null, null, 'NOT AVAILABLE') B ON A.iso2 = B.iso2 order by B.iso2 </pre> <p>En el código SQL anterior se observa que para esta carga es necesario acceder a dos tablas del STG, ya que contienen los datos en diferente granularidad. Además, se crean manualmente los registros ficticios de tipo NA.</p>
 Add sequence	<p>Todas las tablas de dimensiones deben tener un campo clave primaria. Este campo es el referenciado desde la clave foránea de la tabla de hechos.</p> <p>Lo más habitual es crear una secuencia en el flujo del ETL o configurar un <i>identity</i> en el diseño físico de la tabla.</p>




Table output (DIM_Region)

Add sequence

Step name: Add sequence

Name of value: pk

Use a database to generate the sequence

Use DB to get sequence? ☐

Connection: [dropdown] Edit... New... Wizard...

Schema name: [dropdown] Schemas...

Sequence name: SEQ_ Sequences...

Use a transformation counter to generate the sequence

Use counter to calculate sequence? ☒

Counter name (optional): [text]

Start at value: 1

Increment by: 1

Maximum value: 999999999

Help OK Cancel

Tras seleccionar la conexión al DW y la tabla DIM_Region, es necesario establecer el mapeo de campos.




Table output (DIM_Region)

Table output

Step name: Table output (DIM_Region)

Connection: cn_dw Edit

Target schema: dbo

Target table: DIM_Region

Commit size: 1000

Truncate table: ☐

Ignore insert errors: ☐

Specify database fields: ☒

Main options Database fields

Fields to insert:

#	Table field	Stream field
1	region	comunidad_autono...
2	country_code2	iso2
3	country_code3	iso3
4	country_name	name
5	pk_region	pk

El resultado de la ejecución es el siguiente (264 registros):

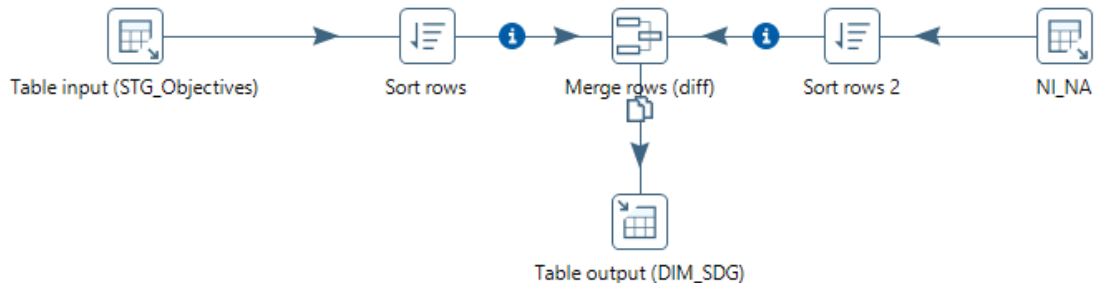
Execution Results												
Logging Execution History Step Metrics Performance Graph Metrics Preview data												
#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	Table input (Regions)	0	0	264	264	0	0	0	0	Finished	0.0s	7,765
2	Add sequence	0	264	264	0	0	0	0	0	Finished	0.1s	4,328
3	Table output (DIM_Region)	0	264	264	0	264	0	0	0	Finished	0.1s	2,422

3.3.5.3 TR_DIM_SDG




En esta carga se mostrará otro método para insertar los registros NA. No se utilizará

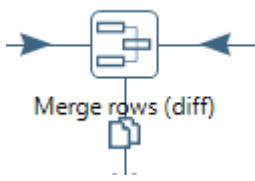
código SQL, sino que el proceso se realizará directamente en el flujo de datos del ETL.

La transformación completa es la siguiente:



A continuación, se detallan los pasos más significativos:

 Table input (STG_Objectives)	<p>En este caso el código SQL del <i>table input</i> se reduce a una sentencia básica de selección sobre la tabla STG.</p> <p>SQL</p> <pre>SELECT Objetivo , Nombre , Descripcion FROM dbo.STG_Objectives</pre>										
 NI_NA	<p>En paralelo a la operación anterior se configura un nuevo <i>input</i> con los valores NA.</p> <p>SQL</p> <pre>SELECT 999 as Objetivo, 'NA' as Nombre, 'NOT AVAILABLE' as Descripcion</pre>										
 Sort rows	<p>Para poder unir los dos flujos de entrada, es necesario que estén ordenados por el mismo campo clave (objetivo).</p> <p>Sort rows</p> <div><div>Step name</div><div>Sort directory</div><div>TMP-file prefix</div><div>Sort size (rows in memory)</div><div>Free memory threshold (in %)</div><div>Compress TMP Files?</div><div>Only pass unique rows? (verifies keys only)</div></div> <p>Fields :</p> <table><tr><th>#</th><th>Fieldname</th><th>Ascending</th><th>Case sensitive compare?</th><th>Sort based on c</th></tr><tr><td>1</td><td>Objetivo</td><td>Y</td><td>N</td><td>N</td></tr></table>	#	Fieldname	Ascending	Case sensitive compare?	Sort based on c	1	Objetivo	Y	N	N
#	Fieldname	Ascending	Case sensitive compare?	Sort based on c							
1	Objetivo	Y	N	N							



Merge rows (diff)

Mediante la transformación «Merge» se fusionan los dos *input* del ETL, utilizando el campo «Objetivo».

Merge rows (diff)

Step name: Merge rows (diff)

Reference rows origin: Sort rows

Compare rows origin: Sort rows 2

Flag fieldname: flagfield

Keys to match :		Values to compare :	
#	Key field	#	Value field
1	Objetivo	1	Objetivo

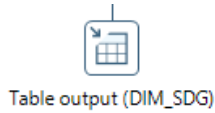


Table output (DIM_SDG)

Table output

Step name: Table output (DIM_SDG)

Connection: cn_dw

Target schema: dbo

Target table: DIM_SDG

Commit size: 1000

Truncate table: ☐

Ignore insert errors: ☐

Specify database fields: ☒

Main options | Database fields

#	Table field	Stream field
1	pk_sdg	Objetivo
2	sdg_name	Nombre
3	sdg_description	Descripcion

El resultado de la ejecución es el siguiente (18 registros):

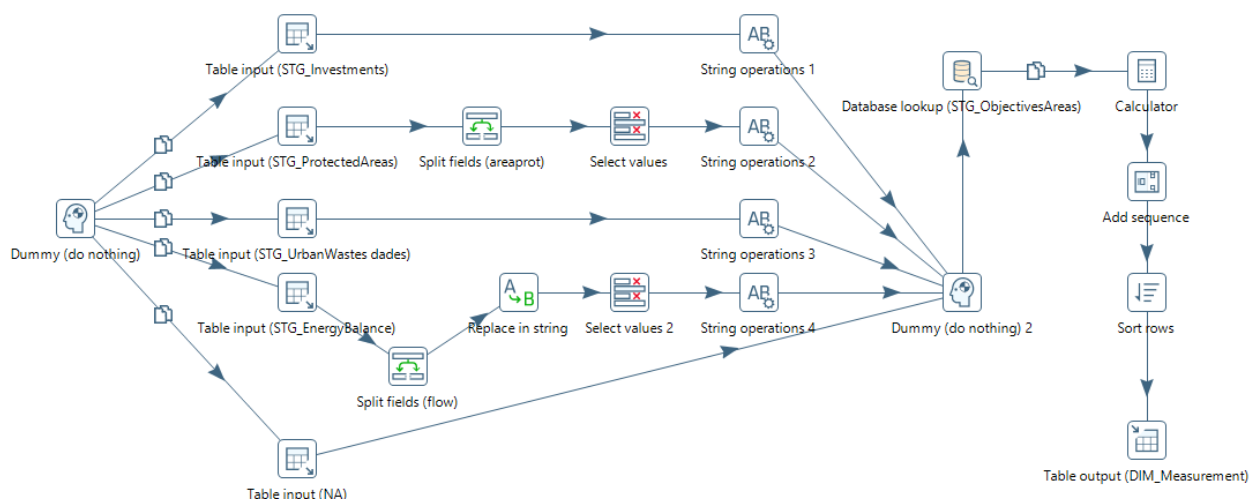
Execution Results

Execution Results												
Logging Execution History Step Metrics Performance Graph Metrics Preview data												
#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	Table input (STG_Objectives)	0	0	17	17	0	0	0	0	Finished	0.0s	1,133
2	NI_NA	0	0	1	1	0	0	0	0	Finished	0.0s	77
3	Sort rows 2	0	1	1	0	0	0	0	0	Finished	0.0s	21
4	Sort rows	0	17	17	0	0	0	0	0	Finished	0.0s	362
5	Merge rows (diff)	0	18	18	0	0	0	0	0	Finished	0.4s	45
6	Table output (DIM_SDG)	0	18	18	0	18	0	0	0	Finished	0.4s	42



3.3.5.4 TR_DIM_Measurement

La transformación para la carga de esta dimensión es más compleja desde el punto de vista de que se debe acceder a 4 tablas STG y unir sus flujos de datos. Además, debido al diseño en «copo de nieve», es necesario realizar un *lookup* para identificar el ODS principal asociado a cada métrica.

La transformación completa es la siguiente:



A continuación, se detallan los pasos más significativos:

 <p>Table input (STG_Investments)</p>	<p>Para cada uno de los orígenes (tablas STG) se debe construir la sentencia SQL que permita seleccionar los datos deseados. En los cuatro casos se obtienen los campos «measurement» y «unit». Además, se recupera un tercer campo necesario para asignar la PK del valor «NA» a los registros de la tabla de hechos, sin una correspondencia con el ODS.</p> <p>SQL</p> <pre>select distinct ambito_medioambiental as measurement, '€' as unit, (select pk_sdg from DIM_SDG where sdg_name = 'NA') as pk_ODS_Principal_NA from dbo.STG_Investments</pre>
 <p>Table input (STG_ProtectedAreas)</p>	<p>SQL</p> <pre>SELECT distinct areaprot as measurement, areaprot as unit, (select pk_sdg from DIM_SDG where sdg_name = 'NA') as pk_ODS_Principal_NA FROM dbo.STG_ProtectedAreas WHERE areaprot <> 'AREA_KM2' -- AREA_KM2 = MPA_KM2 + TPA_KM2</pre>





 Table input (STG_UrbanWastes dades)	<p>SQL</p> <pre>select distinct ISNULL(VAR,(SELECT max(VAR) FROM STG_UrbanWastes WHERE group_id = dades.group_id)) as measurement, ISNULL(UNIT,(SELECT max(UNIT) FROM STG_UrbanWastes WHERE group_id = dades.group_id)) as unit, (SELECT pk_sdg FROM DIM_SDG WHERE sdg_name = 'NA') as pk_ODS_Principal_NA from dbo.STG_UrbanWastes dades</pre>																																																	
 Table input (STG_EnergyBalance)	<p>SQL</p> <pre>SELECT distinct flow as measurement, flow as unit, (select pk_sdg from DIM_SDG where sdg_name = 'NA') as pk_ODS_Principal_NA FROM dbo.STG_EnergyBalance</pre>																																																	
 Table input (NA)	<p>En este INPUT TABLE simulamos el registro ficticio correspondiente a NA</p> <p>SQL</p> <pre>select 'NA' as measurement, 'NA' as unit, (select pk_sdg from DIM_SDG where sdg_name = 'NA') as pk_ODS_Principal_NA</pre>																																																	
 Split fields (areaprot)	<p>En el caso de «Protected areas» la unidad viene definida en el propio nombre de la medida y es necesario extraerla en un campo independiente. Se utiliza un <i>split fields</i> con el carácter delimitador de campo «_».</p> <p> Split fields</p> <table><tr><td>Step name</td><td colspan="6">Split fields (areaprot) </td></tr><tr><td>Field to split</td><td colspan="6">unit</td></tr><tr><td>Delimiter</td><td colspan="6">_</td></tr><tr><td>Enclosure</td><td colspan="6"></td></tr></table> <p>Fields</p> <table><tr><th>#</th><th>New field</th><th>ID</th><th>Remove ID?</th><th>Type</th><th>Length</th><th>Precision</th></tr><tr><td>1</td><td>dummy</td><td></td><td>N</td><td>String</td><td></td><td></td></tr><tr><td>2</td><td>unit</td><td></td><td>N</td><td>String</td><td></td><td></td></tr></table>	Step name	Split fields (areaprot)						Field to split	unit						Delimiter	_						Enclosure							#	New field	ID	Remove ID?	Type	Length	Precision	1	dummy		N	String			2	unit		N	String		
Step name	Split fields (areaprot)																																																	
Field to split	unit																																																	
Delimiter	_																																																	
Enclosure																																																		
#	New field	ID	Remove ID?	Type	Length	Precision																																												
1	dummy		N	String																																														
2	unit		N	String																																														
 Split fields (flow)	<p>Lo mismo ocurre para «Energy balance»: se realiza la misma transformación «Split fields», utilizando en este caso el «(» como carácter delimitador de campo.</p>																																																	

Table output

Step name

Table output (DIM_Measurement)

Connection

cn_dw

Target schema

dbo

Target table

DIM_Measurement

Commit size

1000

Truncate table

☐

Ignore insert errors

☐

Specify database fields

☒

Main options

Database fields

Fields to insert:

#	Table field	Stream field
1	measurement_code	measurement
2	unit	unit
3	fk_sdg	pk_ODS
4	measurement_name	name
5	pk_measurement	pk

El resultado de la ejecución es el siguiente:

Execution Results

<div> <div>Logging</div> <div>Execution History</div> <div>Step Metrics</div> <div>Performance Graph</div> <div>Metrics</div> <div>Preview data</div> </div>												
#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	Dummy (do nothing)	0	0	0	0	0	0	0	0	Finished	0.0s	0
2	Table input (STG_ProtectedAreas)	0	0	3	3	0	0	0	0	Finished	0.1s	22
3	Table input (STG_EnergyBalance)	0	0	13	13	0	0	0	0	Finished	0.7s	18
4	Table input (STG_UrbanWastes dades)	0	0	29	29	0	0	0	0	Finished	0.3s	112
5	Split fields (flow)	0	13	13	0	0	0	0	0	Finished	0.7s	18
6	Table input (STG_Investments)	0	0	7	7	0	0	0	0	Finished	0.1s	51
7	Split fields (areaprot)	0	3	3	0	0	0	0	0	Finished	0.2s	15
8	Table input (NA)	0	0	1	1	0	0	0	0	Finished	0.1s	7
9	Select values	0	3	3	0	0	0	0	0	Finished	0.2s	14
10	String operations 1	0	7	7	0	0	0	0	0	Finished	0.2s	35
11	String operations 3	0	29	29	0	0	0	0	0	Finished	0.3s	109
12	String operations 2	0	3	3	0	0	0	0	0	Finished	0.2s	12
13	Replace in string	0	13	13	0	0	0	0	0	Finished	0.7s	18
14	Select values 2	0	13	13	0	0	0	0	0	Finished	0.7s	18
15	String operations 4	0	13	13	0	0	0	0	0	Finished	0.7s	18
16	Dummy (do nothing) 2	0	53	53	0	0	0	0	0	Finished	0.7s	72
17	Database lookup (STG_ObjectivesAreas)	0	53	53	50	0	0	0	0	Finished	0.7s	71
18	Calculator	0	53	53	0	0	0	0	0	Finished	0.7s	71
19	Add sequence	0	53	53	0	0	0	0	0	Finished	0.8s	69
20	Sort rows	0	53	53	0	0	0	0	0	Finished	0.8s	66
21	Table output (DIM_Measurement)	0	53	53	0	53	0	0	0	Finished	0.8s	63

3.3.5.5 TR_DIM_EconomicActivitySector

La transformación para alimentar DIM_EconomicActivitySector es básica y se obviará el detalle de algunos pasos. Está compuesta por tres pasos: extracción de datos del STG, creación de secuencia y carga de datos en el DW.

La transformación completa es la siguiente:



A continuación, se detallan los pasos más significativos:




Table input (STG_Investments)

SQL

```
SELECT distinct sector_economico
FROM dbo.STG_Investments
union
select 'NA'--, 'NOT AVAILABLE'
```

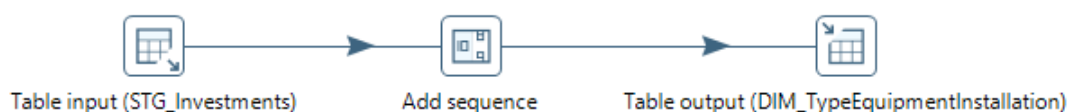
El resultado de la ejecución es el siguiente (2 registros):

Execution Results												
<div> Logging Execution History Step Metrics Performance Graph Metrics Preview data </div>												
#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	Table input (STG_Investments)	0	0	2	2	0	0	0	0	Finished	0.0s	54
2	Add sequence	0	2	2	0	0	0	0	0	Finished	0.0s	44
3	Table output	0	2	2	0	2	0	0	0	Finished	0.1s	22



3.3.5.6 TR_DIM_TypeEquipmentInstallation

El proceso para cargar esta dimensión también consiste en un flujo básico de tres pasos equivalentes a DIM_EconomicActivitySector.

La transformación completa es la siguiente:



A continuación, se detallan los pasos más significativos:

 Table input (STG_Investments)	<pre>SELECT distinct tipo_instalacion FROM dbo.STG_Investments union select 'NA'--, 'NOT AVAILABLE'</pre>									
 (DIM_TypeEquipmentInstallation)	<div> <div>Table output</div> <div> <div>Step name</div> <div>Table output (DIM_TypeEquipmentInstallation)</div> </div> <div> <div>Connection</div> <div>cn_dw Edit...</div> </div> <div> <div>Target schema</div> <div>dbo</div> </div> <div> <div>Target table</div> <div>DIM_TypeEquipmentInstallation</div> </div> <div> <div>Commit size</div> <div>1000</div> </div> <div> <div>Truncate table</div> <div><input type="checkbox"/></div> </div> <div> <div>Ignore insert errors</div> <div><input type="checkbox"/></div> </div> <div> <div>Specify database fields</div> <div><input checked="" type="checkbox"/></div> </div> <div> <div>Main options</div> <div>Database fields</div> </div> <div> <div>Fields to insert:</div> <table border="1"> <thead> <tr> <th>#</th> <th>Table field</th> <th>Stream field</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>typeequipinstall_name</td> <td>tipo_instalacion</td> </tr> <tr> <td>2</td> <td>pk_typeequipinstall</td> <td>pk</td> </tr> </tbody> </table> </div> </div>	#	Table field	Stream field	1	typeequipinstall_name	tipo_instalacion	2	pk_typeequipinstall	pk
#	Table field	Stream field								
1	typeequipinstall_name	tipo_instalacion								
2	pk_typeequipinstall	pk								

El resultado de la ejecución es el siguiente (3 registros):

Execution Results												
Logging Execution History Step Metrics Performance Graph Metrics Preview data												
#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	Table input (STG_Investments)	0	0	3	3	0	0	0	0	Finished	0.0s	250
2	Add sequence	0	3	3	0	0	0	0	0	Finished	0.0s	107
3	Table output (DIM_TypeEquipmentInstallation)	0	3	3	0	3	0	0	0	Finished	0.1s	50


3.3.5.7 TR_DIM_Country

El proceso de carga por DIM_Country es una transformación básica.

La transformación completa es la siguiente:



A continuación, se detallan los pasos más significativos:

 Table input (STG_Countries)	<pre>SELECT nombre , name , nom , iso2 , iso3 , phone_code FROM dbo.STG_Countries UNION SELECT 'NA' as nombre, 'NA' as name, 'NA' as nom, NULL as iso2, NULL as iso3, NULL as phone_code ORDER BY iso2</pre>
--	--

El resultado de la ejecución es el siguiente (247 registros):

Execution Results												
Logging Execution History Step Metrics Performance Graph Metrics Preview data												
#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	Table input (STG_Countries)	0	0	247	247	0	0	0	0	Finished	0.1s	2,398
2	Add sequence	0	247	247	0	0	0	0	0	Finished	0.2s	1,165
3	Table output (DIM_Country)	0	247	247	0	247	0	0	0	Finished	0.3s	767

3.3.5.8 TR_DIM_Product

El proceso de carga par DIM_Product es una transformación básica.

La transformación completa es la siguiente:



El resultado de la ejecución es el siguiente (12 registros):

Execution Results												
Logging Execution History Step Metrics Performance Graph Metrics Preview data												
#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	Table input (STG_EnergyBalance)	0	0	12	12	0	0	0	0	Finished	0.5s	
2	Add sequence	0	12	12	0	0	0	0	0	Finished	0.6s	
3	Table output (DIM_Product)	0	12	12	0	12	0	0	0	Finished	0.6s	

3.3.6 Bloque TR_FACT

Este bloque contiene las transformaciones para la carga inicial de las tablas de hecho «FACT_» al almacén desde las tablas intermedias «STG_» del *staging area*.

Con la implementación y ejecución de los procesos de carga de dimensiones se obtiene una gran cantidad de datos en el modelo dimensional y permite pasar a añadir los datos al modelo de hechos, haciendo referencia a las dimensiones disponibles mediante sus claves foráneas.

La parte principal en la carga de las tablas de hechos es la búsqueda de los valores de las claves foráneas en las tablas de dimensiones cargadas anteriormente. En estas transformaciones se integrarán las tablas de datos en una única tabla de hechos del Data Warehouse.

De la misma manera que en la carga de las dimensiones, siempre que sea necesario unir dos o más flujos de datos o realizar una unión los datos deben estar ordenados por la misma clave.

Existen varias soluciones para tratar los valores nulos en claves foráneas: eliminar los registros «incompletos» (asumiendo una pérdida de datos), asignar valores constantes,

buscar registros NI, NA dinámicamente, etc.

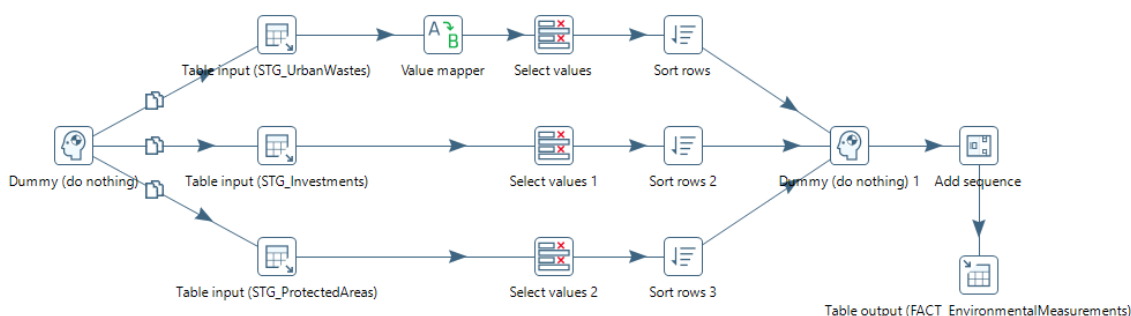
3.3.6.1 TR_FACT_EnvironmentalMeasurements

FACT_EnvironmentalMeasurements

Para poblar esta tabla de hechos es necesario obtener datos de tres tablas STG y unir los flujos de datos. Al haber normalizado previamente los datos en el bloque IN, el proceso es relativamente sencillo.

Existen varias opciones para realizar la carga de las tablas FACT. Para FACT_EnvironmentalMeasurements se mostrará como integrar código SQL en el flujo de datos para obtener el valor de las claves foráneas (FK), correspondientes a las claves primarias (PK) de las dimensiones. En el siguiente ejemplo se mostrará cómo realizar esta misma operación utilizando transformaciones de tipo *lookup*.

La transformación completa es la siguiente:



A continuación, se detallan los pasos más significativos:



Mediante el uso de una entrada de tipo tabla, es posible establecer el código SQL necesario para obtener los datos del *staging area*. En el siguiente ejemplo se puede observar la normalización de los datos provenientes de STG_UrbanWastes (subconsulta), así como la obtención de las FKs necesarias para cargar correctamente la tabla de hechos. Es importante observar que, en caso de no obtener un valor para las FK, el procedimiento le asigna el valor correspondiente para el registro «ficticio» de tipo NA (*not available*).

Table input (STG_UrbanWastes)

Step name Table input (STG_UrbanWastes)

Connection cn_stage

SQL

```
select
  (select pk_date from DIM_Date where date_year = UV.TIME and date_month=1 and date_day=1) as fk_date,
  ISNULL( (select pk_region from DIM_Region where country_code3 = UV.COU AND region = 'NA'),
  (select pk_region from DIM_Region where country_name = 'NOT AVAILABLE')) as fk_region,
  (select pk_activitysector from DIM_EconomicActivitySector where activitysector_name = 'NA') as fk_activitysector,
  (select pk_typeequipinstall from DIM_TypeEquipmentInstallation where typeequipinstall_name = 'NA') as fk_typeequipinstall,
  ISNULL( (select pk_measurement from DIM_Measurement where measurement_code = UV.VAR),
  (select pk_measurement from DIM_Measurement where measurement_code = 'NA')) as fk_measurement,
  ISNULL(value, 0) as value
from
  (
    select group_id,
    ISNULL(COU, (select max(COU) FROM STG_UrbanWastes WHERE group_id = dates.group_id)) as COU,
    ISNULL(VAR, (select max(VAR) FROM STG_UrbanWastes WHERE group_id = dates.group_id)) as VAR,
    ISNULL(TIME_FORMAT, (select max(TIME_FORMAT) FROM STG_UrbanWastes WHERE group_id = dates.group_id)) as TIME_FORMAT,
    ISNULL(UNIT, (select max(UNIT) FROM STG_UrbanWastes WHERE group_id = dates.group_id)) as UNIT,
    ISNULL(POWERCODE, (select max(POWERCODE) FROM STG_UrbanWastes WHERE group_id = dates.group_id)) as POWERCODE, |
    TIME,
    VALUE,
    OBS STATUS
  from STG_UrbanWastes dates
  where TIME is NOT NULL
) UV
```

Table input (STG_Investments)

Para STG_Investments se sigue la misma estrategia con la diferencia que no es necesario normalizar los datos, esto ya fue tratado en el bloque IN correspondiente.

Step name Table input (STG_Investments)

Connection cn_stage

SQL

```
select
  (select pk_date from DIM_Date where date_year = INV.Periodo and date_month=1 and date_day=1) as fk_date,
  ISNULL( (select pk_region from DIM_Region where region = INV.comunidad_autonoma),
  (select pk_region from DIM_Region where region = 'NA' and country_code2 = 'ES')) as fk_region,
  ISNULL( (select pk_activitysector from DIM_EconomicActivitySector where activitysector_name = INV.sector_economico),
  (select pk_activitysector from DIM_EconomicActivitySector where activitysector_name = 'NA')) as fk_activitysector,
  ISNULL( (select pk_typeequipinstall from DIM_TypeEquipmentInstallation where typeequipinstall_name = INV.tipo_instalacion),
  (select pk_typeequipinstall from DIM_TypeEquipmentInstallation where typeequipinstall_name = 'NA')) as fk_typeequipinstall,
  ISNULL( (select pk_measurement from DIM_Measurement where measurement_code = INV.ambito_medioambiental),
  (select pk_measurement from DIM_Measurement where measurement_code = 'NA')) as fk_measurement,
  ISNULL(inversion, 0) as value
from [dbo].[STG_Investments] INV
```

Table input (STG_ProtectedAreas)

De la misma manera se obtienen los datos de STG_ProtectedAreas.

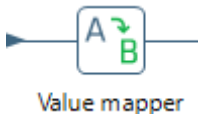
Step name Table input (STG_ProtectedAreas)

Connection cn_stage

SQL

```
select
  (select pk_date from DIM_Date where date_year = PA.year and date_month=1 and date_day=1) as fk_date,
  isnull( (select pk_region from DIM_Region where country_code2 = PA.geotime and region = 'NA'),
  (select pk_region from DIM_Region where country_code2 = 'NA')) as fk_region,
  (select pk_activitysector from DIM_EconomicActivitySector where activitysector_name = 'NA') as fk_activitysector,
  (select pk_typeequipinstall from DIM_TypeEquipmentInstallation where typeequipinstall_name = 'NA') as fk_typeequipinstall,
  isnull( (select pk_measurement from DIM_Measurement where measurement_code = PA.areaprot),
  (select pk_measurement from DIM_Measurement where measurement_code = 'NA')) as fk_measurement,
  ISNULL(value, 0) as value
from [dbo].[STG_ProtectedAreas] PA
```

Para el caso de STG_UrbanWastes es necesario realizar un mapeo de valores:



Mediante un *Value mapper* se corrigen algunos datos incorrectos (NaN).

Value mapper

Step name:

Fieldname to use:


Target field name (empty=overwrite):

Default upon non-matching:

Field values:

#	Source value	Target value
1	NaN	0

Para cada uno de los tres flujos de datos, se aplican las siguientes transformaciones:



Se seleccionan los campos y se configuran los metadatos.

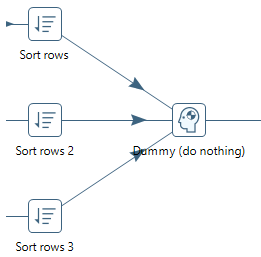
Select values

Step name:

Select & Alter | Remove | Meta-data

Fields to alter the meta-data for:

#	Fieldname	Rename to	Type	Length	Precision	Binary to Normal?	Format	Date Format Lenient?
1	fk_date		Integer	9	0	N		N
2	fk_region		Integer	9	0	N		N
3	fk_activitysector		Integer	9	0	N		N
4	fk_typeequipinstall		Integer	9	0	N		N
5	fk_measurement		Integer	9	0	N		N
6	value		Number			N	#,##0.###	N



Se ordenan los registros y se unen en un único flujo de datos, que es el que se cargará en la tabla de hechos.

Sort rows

Step name:

Sort directory:

TMP-file prefix:

Sort size (rows in memory):

Free memory threshold (in %):

Compress TMP Files? ☐

Only pass unique rows? (verifies keys only) ☐

Fields:

#	Fieldname	Ascending	Case sensitive compare?	Sort based on current locale?
1	fk_date	Y	N	N
2	fk_region	Y	N	N
3	fk_activitysector	Y	N	N
4	fk_typeequipinstall	Y	N	N
5	fk_measurement	Y	N	N

Una vez unificado el flujo, se crea una secuencia que se utilizará como PK de la tabla de hechos y se realiza la carga de los datos

Table output

Step name: Table output (FACT_EnvironmentalMeasurements)

Connection: cn_dw

Target schema: dbo

Target table: FACT_EnvironmentalMeasurements

Commit size: 1000

Truncate table: ☐

Ignore insert errors: ☐

Specify database fields: ☒

Main options | Database fields

Fields to insert:

#	Table field	Stream field
1	fk_date	fk_date
2	fk_region	fk_region
3	fk_activitysector	fk_activitysector
4	fk_typeequipinstall	fk_typeequipinstall
5	fk_measurement	fk_measurement
6	value	value
7	pk_id	pk

El resultado de la ejecución es el siguiente (24.477 registros):

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	Table input (STG_UrbanWastes)	0	0	20779	20779	0	0	0	0	Finished	0.8s	24,649
2	Table input (STG_ProtectedAreas)	0	0	1080	1080	0	0	0	0	Finished	0.4s	2,975
3	Table input (STG_Investments)	0	0	2618	2618	0	0	0	0	Finished	0.4s	6,370
4	Value mapper	0	20779	20779	0	0	0	0	0	Finished	0.8s	24,620
5	Replace in string	0	20779	20779	0	0	0	0	0	Finished	0.8s	24,475
6	Sort rows 2	0	2618	2618	0	0	0	0	0	Finished	0.4s	6,117
7	Sort rows 3	0	1080	1080	0	0	0	0	0	Finished	0.4s	2,983
8	Select values	0	20779	20779	0	0	0	0	0	Finished	0.9s	24,303
9	Sort rows	0	20779	20779	0	0	0	0	0	Finished	0.9s	22,684
10	Dummy (do nothing)	0	24477	24477	0	0	0	0	0	Finished	0.9s	25,929
11	Add sequence	0	24477	24477	0	0	0	0	0	Finished	1.4s	17,237
12	Table output	0	24477	24477	0	24477	0	0	0	Finished	2.0s	12,178

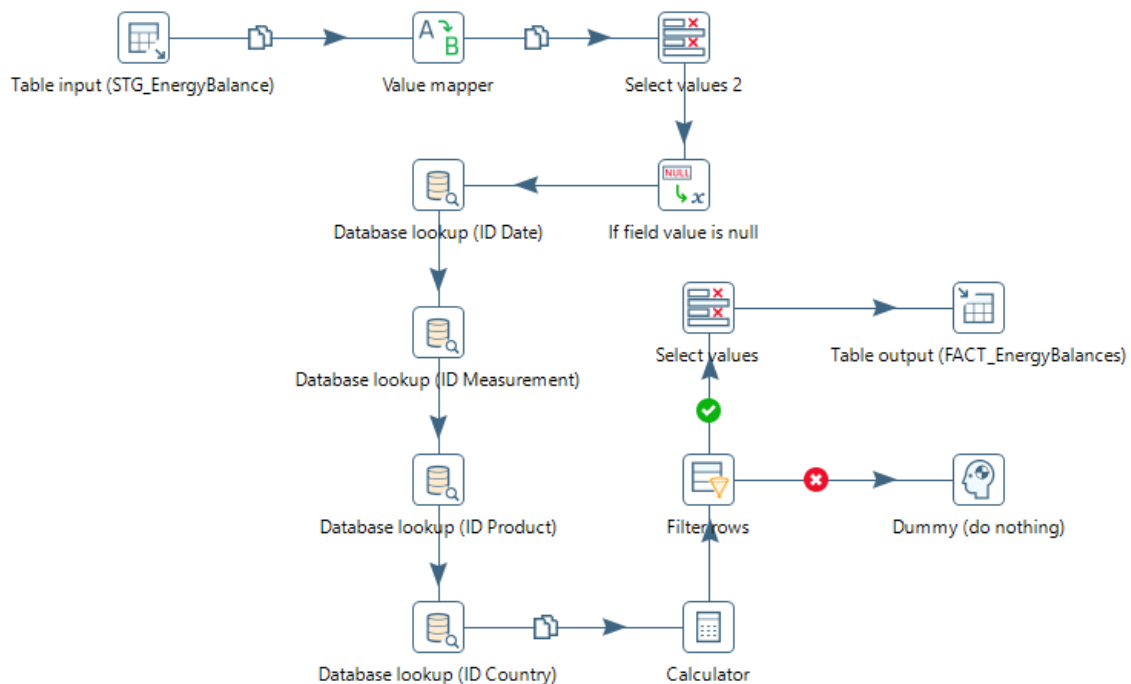
3.3.6.2 TR_FACT_EnergyBalances

FACT_EnergyBalances

La carga de la tabla de hechos «FACT_EnergyBalances» es más sencilla, ya que únicamente es necesario obtener los datos de una tabla intermedia.

A pesar de ello, también es necesario obtener los valores de las claves foráneas. En este caso se mostrará cómo realizar esta operación mediante el uso de transformaciones «Database lookup».

La transformación completa es la siguiente:



A continuación, se detallan los pasos más significativos:



 : (STG_EnergyBalance)	Se realiza una consulta SQL para obtener las claves de negocio necesarias para los <i>lookups</i> posteriores y los IDs de los registros de tipo NA, de cada una de las dimensiones.
--	--

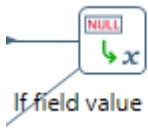
Table input

Step name Table input (STG_EnergyBalance)
Connection cn_stage

SQL

```
select
    year,
    flow as measurement,
    (select pk_measurement from DIM_Measurement where measurement_code = 'NA') as ID_measurement_NA,
    product,
    (select pk_product from DIM_Product where product_name = 'NA') as ID_product_NA,
    country,
    (select pk_country from DIM_Country where country_name_en = 'NA') as ID_country_NA,
    value
from [STG_EnergyBalance] EB
```

 Value mapper	Para mejorar la calidad del dato, se observa la necesidad de mapear manualmente algunos valores y corregir campos nulos.
---	--



If field value is null

Value mapper

Step name:


Fieldname to use:

Target field name (empty=overwrite):

Default upon non-matching:

Field values:

#	Source value	Target value
1	United States	United States of America
2	Slovak Republic	Slovakia
3	People's Republic of China	China



If field value is null

Step name:

Replace Null for all fields

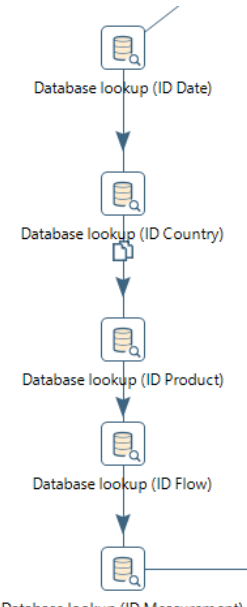
Replace by value:

Set empty string? ☐

Mask (Date):


Fields

#	Field	Replace by value	Conversion mask (Date)
1	value	0	



Un *lookup* consiste en ir a buscar a la tabla de dimensión el valor de la clave primaria (PK) correspondiente a la clave de negocio disponible en la tabla de hechos. El resultado se utilizará para cargar el campo de la clave foránea (FK) correspondiente.

A modo de ejemplo se muestra la configuración de «Database lookup» para «product».

 Database lookup

Step name

Connection

Lookup schema

Lookup table

Enable cache? ☒

Cache size in rows (0=cache)


Load all data from table ☐

The key(s) to look up the value(s):


#	Table field	Comparator	Field1	Field2
1	product_name	=	product	

Values to return from the lookup table :

#	Field	New name	Default	Type
1	pk_product	pk_fk_product		Integer



Calculator

 Calculator

Step name

☒ Throw an error on non existing files

Fields:

#	New field	Calculation	Field A	Field B	Field C	Value type
1	country2	NVL(A, B)	pk_fk_country	ID_country_NA		Integer
2	product2	NVL(A, B)	pk_fk_product	ID_product_NA		Integer
3	measurement2	NVL(A, B)	pk_fk_measurement	ID_measurement_NA		Integer




Select values

Filter rows

Dummy (do nothing)

En esta carga se observan problemas de calidad del dato (datos incompletos) que implicarán tomar alguna decisión o estrategia de carga. En este caso se considera aceptable la pérdida de cierto grado de información y, por lo tanto, se eliminarán del flujo de datos aquellos registros que no obtengan un valor válido para pk_fk_country.



Filter rows

Step name

Send 'true' data to step:

Send 'false' data to step:

The condition:


Table output

Step name

Connection

Target schema

Target table

Commit size

Truncate table ☐

Ignore insert errors ☐

Specify database fields ☒

Main options Database fields

Fields to insert:

#	Table field	Stream field
1	pk_fk_date	pk_fk_date
2	pk_fk_product	pk_fk_product
3	pk_fk_country	pk_fk_country
4	pk_fk_measurement	pk_fk_measurement
5	value	value



(FACT_EnergyBalances)

El resultado de la ejecución es el siguiente:

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	Table input (STG_EnergyBalance)	0	0	296352	296352	0	0	0	0	Finished	14mn 13s	347
2	Value mapper	0	296352	296352	0	0	0	0	0	Finished	15mn 14s	324
3	If field value is null	0	296352	296352	0	0	0	0	0	Finished	16mn 4s	307
4	Database lookup (ID Date)	0	296352	296352	296352	0	0	0	0	Finished	17mn 4s	289
5	Database lookup (ID Country)	0	296352	296352	235984	0	0	0	0	Finished	17mn 5s	289
6	Database lookup (ID Product)	0	296352	296352	296352	0	0	0	0	Finished	17mn 5s	289
7	Database lookup (ID Flow)	0	296352	296352	296352	0	0	0	0	Finished	17mn 5s	289
8	Database lookup (ID Measurement)	0	296352	296352	0	0	0	0	0	Finished	17mn 5s	289
9	Calculator	0	296352	296352	0	0	0	0	0	Finished	17mn 5s	289
10	Filter rows	0	296352	296352	0	0	0	0	0	Finished	17mn 5s	289
11	Select values	0	235984	235984	0	0	0	0	0	Finished	17mn 5s	230
12	Dummy (do nothing)	0	60368	60368	0	0	0	0	0	Finished	17mn 5s	59
13	Table output (FACT_EnergyBalances)	0	235984	235984	0	235984	0	0	0	Finished	17mn 5s	230

En este tipo de cargas con uso intenso de *lookups* se recomienda marcar la opción «Enable cache», esto permite mejorar considerablemente los tiempos de carga obtenidos. En este caso se ha pasado de tiempos de carga del orden de **17 minutos a 12 segundos**.

Resultado con la opción «Enable cache» habilitada:

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	Table input (STG_EnergyBalance)	0	0	296352	296352	0	0	0	0	Finished	10.3s	28,769
2	Value mapper	0	296352	296352	0	0	0	0	0	Finished	10.7s	27,792
3	If field value is null	0	296352	296352	0	0	0	0	0	Finished	11.0s	26,834
4	Database lookup (ID Date)	0	296352	296352	49	0	0	0	0	Finished	11.1s	26,763
5	Database lookup (ID Country)	0	296352	296352	43	0	0	0	0	Finished	11.2s	26,552
6	Database lookup (ID Product)	0	296352	296352	11	0	0	0	0	Finished	11.2s	26,361
7	Database lookup (ID Flow)	0	296352	296352	13	0	0	0	0	Finished	11.6s	25,466
8	Database lookup (ID Measurement)	0	296352	296352	0	0	0	0	0	Finished	12.0s	24,630
9	Calculator	0	296352	296352	0	0	0	0	0	Finished	12.4s	23,828
10	Filter rows	0	296352	296352	0	0	0	0	0	Finished	12.9s	22,909
11	Select values	0	235984	235984	0	0	0	0	0	Finished	13.4s	17,561
12	Dummy (do nothing)	0	60368	60368	0	0	0	0	0	Finished	12.9s	4,666
13	Table output (FACT_EnergyBalances)	0	235984	235984	0	235984	0	0	0	Finished	13.9s	16,950

4. Implementación con trabajos (*jobs*) de los procesos ETL

Para planificar correctamente los *jobs* se deben tener en cuenta los siguientes bloques de procesos implementados:

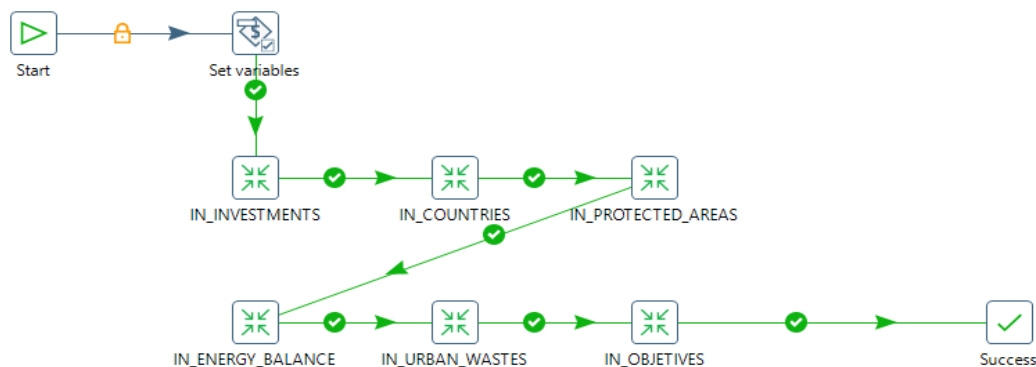
- Bloque «IN_»: procesos de ETL de transformación y carga al área intermedia.
- Bloque «TR_DIM»: procesos de ETL de transformación y carga de dimensiones.
- Bloque «TR_FACT»: procesos de ETL de transformación y carga de hechos.

El diseño de trabajos (*jobs*) mediante PDI va a permitir la ejecución secuencial de todos los procesos del ETL. Cada paso del trabajo contiene una de las transformaciones implementadas en el apartado anterior de diseño de ETL.

4.1 JOB_IN

El trabajo (*job*) «JOB_IN» procesa todas las transformaciones del bloque «IN_» para la carga de datos desde las fuentes de datos proporcionadas al área intermedia (*staging area*).

El diseño completo del trabajo (*job*) «JOB_IN» es el siguiente:



Los pasos incluidos en el trabajo «JOB_IN» son:

- Inicio del *job*.
- Configuración de las variables de entorno (DIR_IN, CN_DW, CN_STAGE)
- Ejecución de las transformaciones «IN_» de carga del *staging area*.
- Finalización del *job*.

El resultado de la ejecución de la transformación completa es el siguiente:

Execution Results					
Job / Job Entry	Comment	Result	Reason	Filename	Nr
▼ JOB_IN					
Job: JOB_IN	Start of job execution		start		
Start	Start of job execution		start		
Start	Job execution finished	Success			0
Set variables	Start of job execution		Followed unconditional link		
Set variables	Job execution finished	Success			0
IN_INVESTMENTS	Start of job execution		Followed link after success		
IN_INVESTMENTS	Job execution finished	Success			2
IN_COUNTRIES	Start of job execution		Followed link after success		
IN_COUNTRIES	Job execution finished	Success			3
IN_PROTECTED_AREAS	Start of job execution		Followed link after success		
IN_PROTECTED_AREAS	Job execution finished	Success			4
IN_ENERGY_BALANCE	Start of job execution		Followed link after success		
IN_ENERGY_BALANCE	Job execution finished	Success			5
IN_URBAN_WASTES	Start of job execution		Followed link after success		
IN_URBAN_WASTES	Job execution finished	Success			6
IN_OBJETIVES	Start of job execution		Followed link after success		
IN_OBJETIVES	Job execution finished	Success			7
Success	Start of job execution		Followed link after success		
Success	Job execution finished	Success			7
Job: JOB_IN	Job execution finished	Success	finished		7

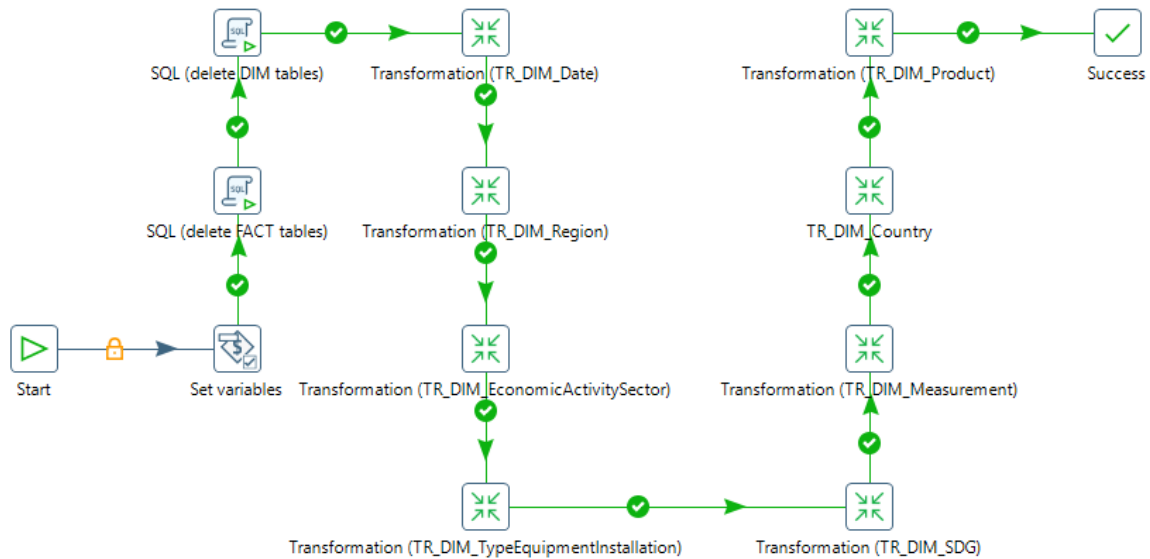
Se observa el procesamiento con éxito de todos los pasos del «JOB_IN» correspondientes a la ejecución de las transformaciones que están incluidas en el trabajo.

En este *job* no es necesario eliminar previamente el contenido de las tablas debido a que se diseñaron sin restricciones ni claves. Esto permite ejecutar un «Truncate table» directamente desde el ETL.

4.2 JOB_TR_DIM

El trabajo (*job*) «JOB_TR_DIM» procesa todas las transformaciones del bloque «TR_DIM» para la carga de datos, desde las tablas intermedias hasta las tablas de dimensiones del almacén.

El diseño completo del trabajo (*job*) «JOB_TR_DIM» es el siguiente:



Los pasos incluidos en el trabajo «JOB_TR_DIM» son:

- Inicio del *job*.
- Carga de variables de entorno (*path* de orígenes de datos y conexiones).
- Borrado de todas las tablas. Esto permite la recarga inicial en caso de ser necesario. Es importante respetar el orden de borrado, según las relaciones definidas entre tablas.
- Ejecución secuencial de todas las transformaciones «TR_DIM» (extracción, transformación y carga de dimensiones).
- Finalización del *job*.

El resultado de la ejecución de la transformación completa es el siguiente:

Execution Results

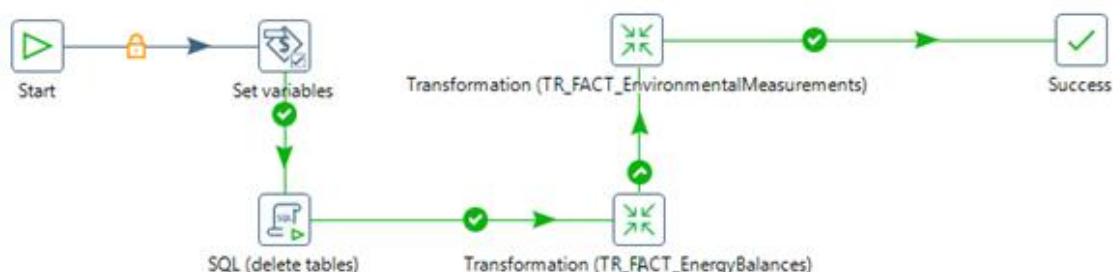
Job / Job Entry	Comment	Result	Reason	Filename	Nr
▼ JOB_TR_DIM					
Job: JOB_TR_DIM	Start of job execution		start		
Start	Start of job execution		start		
Start	Job execution finished	Success			0
Set variables	Start of job execution		Followed unconditional link		
Set variables	Job execution finished	Success			0
SQL (delete FACT tables)	Start of job execution		Followed link after success		
SQL (delete FACT tables)	Job execution finished	Success			0
SQL (delete DIM tables)	Start of job execution		Followed link after success		
SQL (delete DIM tables)	Job execution finished	Success			0
Transformation (TR_DIM_Date	Start of job execution		Followed link after success		
Transformation (TR_DIM_Date	Job execution finished	Success			4
Transformation (TR_DIM_Regi	Start of job execution		Followed link after success		
Transformation (TR_DIM_Regi	Job execution finished	Success			5
Transformation (TR_DIM_Ecor	Start of job execution		Followed link after success		
Transformation (TR_DIM_Ecor	Job execution finished	Success			6
Transformation (TR_DIM_Type	Start of job execution		Followed link after success		
Transformation (TR_DIM_Type	Job execution finished	Success			7
Transformation (TR_DIM_SDG	Start of job execution		Followed link after success		
Transformation (TR_DIM_SDG	Job execution finished	Success			8
Transformation (TR_DIM_Mea	Start of job execution		Followed link after success		
Transformation (TR_DIM_Mea	Job execution finished	Success			9
TR_DIM_Country	Start of job execution		Followed link after success		
TR_DIM_Country	Job execution finished	Success			10
Transformation (TR_DIM_Proc	Start of job execution		Followed link after success		
Transformation (TR_DIM_Proc	Job execution finished	Success			11
Success	Start of job execution		Followed link after success		
Success	Job execution finished	Success			11
Job: JOB_TR_DIM	Job execution finished	Success	finished		11

Se observa el procesamiento con éxito de todos los pasos del «JOB_TR_DIM», correspondientes a la ejecución de las transformaciones que están incluidas en el trabajo.

4.3 JOB_TR_FACT

El trabajo (*job*) «JOB_TR_FACT» procesa todas las transformaciones del bloque «TR_FACT» para la carga de datos desde las tablas intermedias a las tablas de hechos del almacén.

El diseño completo del trabajo (*job*) «JOB_TR_FACT» es el siguiente:



Los pasos incluidos en el trabajo «JOB_TR_FACT» son:

- Inicio del *job*.
- Carga de variables de entorno.
- Borrado de tablas.
- Ejecución de las transformaciones «TR_FACT».
- Finalización del *job*.

El resultado de la ejecución de la transformación completa es el siguiente:

Execution Results

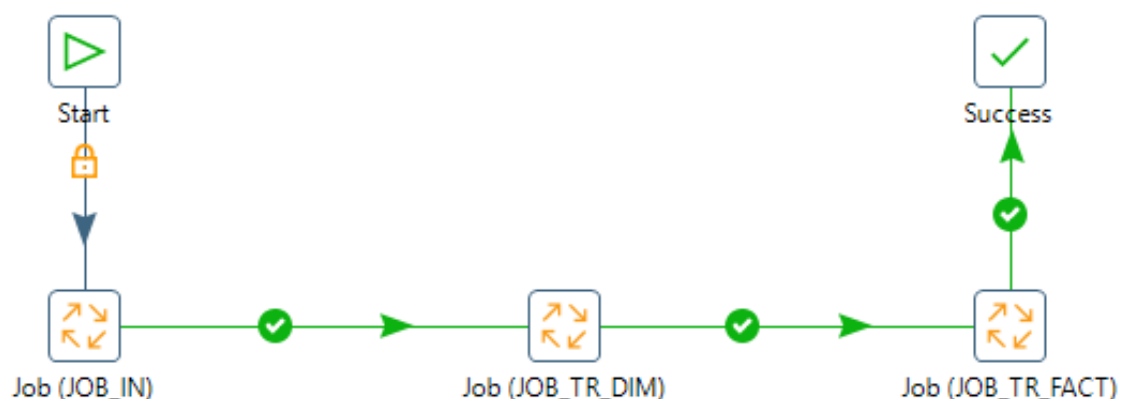
Job / Job Entry	Comment	Result	Reason	Filename	Nr
▼ JOB_TR_FACT					
Job: JOB_TR_FACT	Start of job execution		start		
Start	Start of job execution		start		
Start	Job execution finished	Success			0
Set variables	Start of job execution		Followed unconditional link		
Set variables	Job execution finished	Success			0
SQL (delete tables)	Start of job execution		Followed link after success		
SQL (delete tables)	Job execution finished	Success			0
Transformation (TR_FACT_Ene	Start of job execution		Followed link after success		

Se observa el procesamiento con éxito de los pasos del «JOB_TR_FACT», correspondientes a la ejecución de todas las transformaciones que están incluidas en el trabajo.

4.4 JOB_DW

Finalmente, el trabajo (*job*) «JOB_DW» orquesta todos los trabajos anteriores en un único proceso.

El diseño completo del trabajo (*job*) «JOB_DW» es el siguiente:



Los pasos incluidos en el trabajo «JOB_DW» son:

- Inicio del *job*.
- Ejecución orquestada de los *jobs* de carga de todas las transformaciones («JOB_IN», «JOB_TR_DIM», «JOB_TR_FACT»).
- Finalización del *job*.

El resultado de la ejecución de la transformación completa es el siguiente:

Execution Results

Logging History Job metrics Metrics

Job / Job Entry	Comment	Result	Reason	Filename	Nr
▼ JOB_DW					
Job: JOB_DW	Start of job execution		start		
Start	Start of job execution		start		
Start	Job execution finished	Success			0
Job (JOB_IN)	Start of job execution		Followed unconditional link		
> Job: JOB_IN					
Job (JOB_IN)	Job execution finished	Success			1
Job (JOB_TR_DIM)	Start of job execution		Followed link after success		
> Job: JOB_TR_DIM					
Job (JOB_TR_DIM)	Job execution finished	Success			2
Job (JOB_TR_FACT)	Start of job execution		Followed link after success		
> Job: JOB_TR_FACT					
Job (JOB_TR_FACT)	Job execution finished	Success			3
Success	Start of job execution		Followed link after success		
Success	Job execution finished	Success			3
Job: JOB_DW	Job execution finished	Success	finished		3

Se observa el procesamiento con éxito de todos los pasos del «JOB_DW», correspondientes a la ejecución de las transformaciones que están incluidas en el trabajo.

El tiempo resultante puede variar significativamente en función de las transformaciones realizadas. Es importante recordar que existen múltiples soluciones posibles y válidas. Esta propuesta de solución pretende ser la mejor a nivel pedagógico, aunque podría no ser óptima.