

Caso práctico: Almacén de datos para el análisis de indicadores de crisis en la eurozona

PRA2 – Carga de datos

Nombre estudiante: Gloria Manresa Santamaría

Aula 1

Índice de contenidos

1. IDENTIFICACIÓN DE LOS PROCESOS DE ETL	3
1.1. IDENTIFICACIÓN Y DESCRIPCIÓN DE LOS PROCESOS DE ETL DE ORIGEN A STAGE	3
1.2. IDENTIFICACIÓN Y DESCRIPCIÓN DE LOS PROCESOS DE ETL DE STAGE A DW.....	4
2. DISEÑO Y DESARROLLO DE LOS PROCESOS DE ETL.....	6
2.1. CREACIÓN DE TABLAS	6
2.2. TRANSFORMACIONES BLOQUE IN.....	12
2.2.1. TRANSFORMACIÓN IN_COICOP	13
2.2.2. TRANSFORMACIÓN IN_COUNTRY	16
2.2.3. TRANSFORMACIÓN IN_HICP.....	19
2.2.4. TRANSFORMACIÓN IN_PGAS	23
2.2.5. TRANSFORMACIÓN IN_PELEC	27
2.2.6. TRANSFORMACIÓN IN_CONSUMPTION	28
2.3. TRANSFORMACIÓN BLOQUE TR_DIM.....	30
2.3.1. TRANSFORMACIÓN TR_DIM_COUNTRY.....	30
2.3.2. TRANSFORMACIÓN TR_DIM_DATE.....	33
2.3.3. TRANSFORMACIÓN TR_DIM_DATE_SEMESTER.....	34
2.3.4. TRANSFORMACIÓN TR_DIM_COICOP	36
2.3.5. TRANSFORMACIÓN TR_DIM_TAX.....	39
2.3.6. TRANSFORMACIÓN TR_DIM_PRODUCT.....	42
2.3.7. TRANSFORMACIÓN TR_DIM_CONSUMPTION	44
2.3.8. TRANSFORMACIÓN TR_DIM_CURRENCY	46
2.3.9. TRANSFORMACIÓN TR_DIM_UNIT	48
2.4. TRANSFORMACIÓN BLOQUE TR_FACT	50
2.4.1. TRANSFORMACIÓN FACT_EUROZONE_INDICATORS	50
2.4.2. TRANSFORMACIÓN FACT_ENERY_PRICE.....	53
3. IMPLEMENTACIÓN TRABAJOS (JOBS) DE LOS PROCESOS DE ETL	58
3.1. JOBS BLOQUE IN.....	58
3.2. JOBS BLOQUE TR_DIM.....	59
3.3. JOBS BLOQUE TR_FACT.....	60
3.4. PROCESO COMPLETO (UTILIZA LOS JOBS IN Y TR).....	60

1. IDENTIFICACIÓN DE LOS PROCESOS DE ETL

El proceso de carga en el data warehouse en este caso se ha dividido en diferentes bloques de actualización. Cada uno de ellos tendrá sus correspondientes etapas de extracción, transformación y carga. Esta división permitirá ejecutar los bloques de forma consecutiva o de forma aislada ya que cada uno de ellos corresponderá con una transformación en la herramienta Pentaho Data Integration.

Se identifican dos bloques:

- Bloque IN
- Bloque TR

A continuación, se explican los bloques en profundidad y se identifican algunos de los procesos que forman parte de cada uno de ellos.

1.1. Identificación y descripción de los procesos de ETL de ORIGEN a STAGE

El bloque IN hace referencia al proceso de carga de los datos desde el origen, es decir, desde las fuentes hasta las tablas intermedias (staging area). El prefijo de estos procesos será “IN”.

En este caso este bloque se dividirá en los siguientes procesos:

Nombre ETL	Descripción	Orígenes de datos	Tabla destino (stage)
IN_COUNTRY	Carga de los datos correspondientes a los nombres de los países y los elementos de código ISO 3166-1-alpha-2.	Countries.json	STG_COUNTRY
IN_HICP	Carga de los datos correspondientes a la información relativa al indicador HICP. Uno de los indicadores que se usa en la eurozona para ver la evolución de la economía.	prc_hicp_mv12r.tsv	STG_HICP
IN_PGAS	Carga de los datos correspondientes a la información relativa a la evolución del mercado energético europeo de gas, por países.	nrg_pc_202_tabular.tsv	STG_PGAS

IN_PELEC	Carga de los datos correspondientes a la información relativa a la evolución del mercado energético europeo de la electricidad por países.	nrg_pc_204_tabular.tsv	STG_PELEC
IN_COICOP	Carga de los datos correspondientes a la información relativa al código y descripción de la finalidad de consumo	COICOP.xml	STG_COICOP
IN_CONSUMPTION	Carga de los datos correspondientes a la información relativa al código y descripción de la franja de consumo de energía	consumption_band.csv	STG_CONSUMPTION

1.2. Identificación y descripción de los procesos de ETL de STAGE a DW

El bloque TR hace referencia al proceso de carga de los datos desde tablas intermedias (staging area) hasta nuestro almacén. Podemos encontrar dos tipos de procesos:

- Procesos de ETL para la carga de dimensiones. Estos procesos se distinguen con el prefijo "TR_DIM" en el nombre.
- Procesos de ETL para la carga de las tablas de hecho. Estos procesos se distinguen con el prefijo "TR_FACT" en el nombre.

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

Nombre ETL	Descripción	Tabla origen	Tabla destino
TR_DIM_COUNTRY	Carga y transformación de la dimensión que contiene los datos de los países.	STG_COUNTRY	DIM_COUNTRY
TR_DIM_DATE	Carga y transformación de la dimensión temporal para el análisis de la información de indicadores de la eurozona.	STG_HICP	DIM_DATE

TR_DIM_DATE_SEMESTER	Carga y transformación de la dimensión temporal para el análisis de la información de precios de la energía.	STG_PELEC	DIM_DATE_SEMESTER
TR_DIM_COICOP	Carga y transformación de la dimensión que contiene los datos de las diferentes finalidades de consumo en las que se clasifica el indicador HICP.	STG_COICOP	DIM_COICOP
TR_DIM_TAX	Carga y transformación de la dimensión que contiene los datos de los diferentes tipos de impuestos.	Data Grid (PDI)	DIM_TAX
TR_DIM_PRODUCT	Carga y transformación de la dimensión que contiene los datos de los diferentes tipos de energías.	Data Grid (PDI)	DIM_PRODUCT
TR_DIM_CONSUMPTION	Carga y transformación de la dimensión que contiene los datos de las distintas franjas de consumo.	STG_CONSUMPTION	DIM_CONSUMPTION
TR_DIM_CURRENCY	Carga y transformación de la dimensión que contiene los datos de las distintas monedas.	Data Grid (PDI)	DIM_CURRENCY
TR_DIM_UNIT	Carga y transformación de la dimensión que contiene los datos de las distintas unidades de medida.	Data Grid (PDI)	DIM_UNIT

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

Nombre ETL	Descripción	Tabla origen
TR_FACT_EUROZONE_INDICATORS	Carga y transformación de la tabla de hechos FACT_EUROZONE_INDICATORS	STG_HICP DIM_DATE DIM_COUNTRY DIM_COICOP

TR_FACT_ENERGY_PRICE	Carga y transformación de la tabla de hechos FACT_ENERGY_PRICE	STG_PELEC STG_PGAS DIM_DATE_SEMESTER DIM_COUNTRY DIM_CURRENCY DIM_TAX DIM_CONSUMPTION DIM_UNIT DIM_PRODUCT
----------------------	--	--

2. DISEÑO Y DESARROLLO DE LOS PROCESOS DE ETL

A continuación, vamos a diseñar e implementar los procesos de carga mediante Pentaho Data Integration (PDI) y Spoon.

2.1. Creación de tablas

En primer lugar, para la implementación del proceso de ETL debemos crear las tablas intermedias en la staging área así como las tablas del modelo dimensional (las dimensiones y las tablas de hechos).

En la siguiente imagen se puede observar que el script de creación de tablas se ha ejecutado correctamente.

The screenshot shows the Object Explorer on the left and a query editor window on the right. The query editor contains a script for creating tables in the 'SOURCE_gmanresa' database. The script includes code for creating dimension tables (e.g., STG_COUNTRY, STG_HICP) and fact tables (e.g., FACT_ENERGY_PRICE). A red box highlights the list of dimension tables in the Object Explorer. Another red box highlights the message 'Commands completed successfully.' in the Messages pane at the bottom of the query editor.

```
--IN_COUNTRY:  
--IN_COUNTRY: Tabla de staging con la información de los países  
IF OBJECT_ID(N'dbo.STG_COUNTRY', N'U') IS NOT NULL DROP TABLE dbo.STG_COUNTRY;  
CREATE TABLE dbo.STG_COUNTRY(  
    name nvarchar(100),  
    code nvarchar(10) not null  
)  
  
--IN_HICP:  
--IN_HICP: Tabla de staging con los datos relativos al indicador HICP  
IF OBJECT_ID(N'dbo.STG_HICP', N'U') IS NOT NULL DROP TABLE dbo.STG_HICP;  
CREATE TABLE dbo.STG_HICP(  
    freq nvarchar(50) not null,  
    unit nvarchar(100) not null,  
    coicop varchar(50) not null,  
    geo nvarchar(50) not null,  
    period int not null,  
    HICP numeric(12,2) null)
```

A continuación, se muestran los scripts utilizados en SQL Server para crearlas.

--IN_COUNTRY:

```
--IN_COUNTRY: Tabla de staging con la información de los países
IF OBJECT_ID(N'dbo.STG_COUNTRY', N'U') IS NOT NULL DROP TABLE dbo.STG_COUNTRY;
CREATE TABLE dbo.STG_COUNTRY(
    name nvarchar(100),
    code nvarchar(10) not null
)
```

--IN_HICP:

```
--IN_HICP: Tabla de staging con los datos relativos al indicador HICP
IF OBJECT_ID(N'dbo.STG_HICP', N'U') IS NOT NULL DROP TABLE dbo.STG_HICP;
CREATE TABLE dbo.STG_HICP(
    freq nvarchar(50) not null,
    unit nvarchar(100) not null,
    coicop varchar(50) not null,
    geo nvarchar(50) not null,
    period int not null,
    HICP numeric(12,2) null
)
```

--IN_PGAS:

```
--IN_PGAS: Tabla de staging con los datos relativos al precio del gas
IF OBJECT_ID(N'dbo.STG_PGAS', N'U') IS NOT NULL DROP TABLE dbo.STG_PGAS;
CREATE TABLE dbo.STG_PGAS(
    freq nvarchar(10) not null,
    product int not null,
    consom int not null,
    unit nvarchar(10) not null,
    tax nvarchar(10) not null,
    currency nvarchar(10) not null,
    geo nvarchar(10) not null,
    period nvarchar(10) not null,
    pgas numeric(12,4) null
)
```

--IN_PELEC:

```
--IN_PELEC: Tabla de staging con los datos relativos al precio de la electricidad
IF OBJECT_ID(N'dbo.STG_PELEC', N'U') IS NOT NULL DROP TABLE dbo.STG_PELEC;
CREATE TABLE dbo.STG_PELEC(
    freq nvarchar(10) not null,
    product int not null,
    consom int not null,
    unit nvarchar(10) not null,
    tax nvarchar(10) not null,
    currency nvarchar(10) not null,
    geo nvarchar(10) not null,
    period nvarchar(10) not null,
    pelec numeric(12,4) null
)
```

--IN_COICOP:

```
--IN_COICOP: Tabla de staging con información de finalidad de consumo
IF OBJECT_ID(N'dbo.STG_COICOP', N'U') IS NOT NULL DROP TABLE dbo.STG_COICOP;
CREATE TABLE dbo.STG_COICOP(
    code nvarchar(25) not null,
    description nvarchar(300) null
)
```

--IN_CONSUMPTION:

```
--IN_CONSUMPTION: Tabla de staging con información de la franja de consumo de la energía
IF OBJECT_ID(N'dbo.STG_CONSUMPTION', N'U') IS NOT NULL DROP TABLE dbo.STG_CONSUMPTION;
CREATE TABLE dbo.STG_CONSUMPTION(
    code nvarchar(25) not null,
    description nvarchar(250) null
)
```

--DIM_DATE

```
CREATE TABLE [dbo].[DIM_DATE](
    [pk_date] Int NOT NULL,
    [Year] Int,
    [Mo] Int,
    [Quarter] Int,
    CONSTRAINT [PK_DIM_DATE] PRIMARY KEY CLUSTERED
    (
        [pk_date] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

--DIM_DATE_SEMESTER

```
CREATE TABLE [dbo].[DIM_DATE_SEMESTER](
    [pk_date_semester] nvarchar(10) NOT NULL,
    [Year] Int,
    [Semester] nvarchar(5),
    CONSTRAINT [PK_DIM_DATE_SEMESTER] PRIMARY KEY CLUSTERED
    (
        [pk_date_semester] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

--DIM_COUNTRY

```
CREATE TABLE [dbo].[DIM_COUNTRY] (
    [pk_country] int NOT NULL,
    [code] nvarchar(10),
    [country_name] nvarchar(100),
    CONSTRAINT [PK_DIM_COUNTRY] PRIMARY KEY CLUSTERED
    (
        [pk_country] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

--DIM_COICOP

```
CREATE TABLE [dbo].[DIM_COICOP](
    [pk_coicop] int NOT NULL,
    [code] nvarchar(25),
    [coicop_name] nvarchar(300),
    CONSTRAINT [PK_DIM_COICOP] PRIMARY KEY CLUSTERED
    (
        [pk_coicop] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

--DIM_CURRENCY

```
CREATE TABLE [dbo].[DIM_CURRENCY](
    [pk_currency] int NOT NULL,
    [currency_name] nvarchar(10),
    CONSTRAINT [PK_DIM_CURRENCY] PRIMARY KEY CLUSTERED
    (
        [pk_currency] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

--DIM_TAX

```
CREATE TABLE [dbo].[DIM_TAX](
    [pk_tax] int NOT NULL,
    [tax_name] nvarchar(50),
    CONSTRAINT [PK_DIM_TAX] PRIMARY KEY CLUSTERED
    (
        [pk_tax] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

--DIM_UNIT

```
CREATE TABLE [dbo].[DIM_UNIT](
    [pk_unit] int NOT NULL,
    [unit_name] nvarchar(25),
    CONSTRAINT [PK_DIM_UNIT] PRIMARY KEY CLUSTERED
    (
        [pk_unit] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

--DIM_PRODUCT

```
CREATE TABLE [dbo].[DIM_PRODUCT](
    [pk_product] int NOT NULL,
    [product_code] int,
    CONSTRAINT [PK_DIM_PRODUCT] PRIMARY KEY CLUSTERED
    (
        [pk_product] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

--DIM_CONSUMPTION

```
CREATE TABLE [dbo].[DIM_CONSUMPTION](
    [pk_consumption] int NOT NULL,
    [code] nvarchar(25),
    [consumption_name] nvarchar(250),
    CONSTRAINT [PK_DIM_CONSUMPTION] PRIMARY KEY CLUSTERED
    (
        [pk_consumption] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

--FACT_EUROZONE_INDICATORS

```
CREATE TABLE [dbo].[FACT_EUROZONE_INDICATORS] (
    [pk_id] int NOT NULL,
    [fk_date] int,
    [fk_country] int,
    [fk_coicop] int,
    [HICP] numeric(12,2),
    CONSTRAINT [PK_FACT_EUROZONE_INDICATORS] PRIMARY KEY CLUSTERED
    (
        [pk_id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

--FACT_ENERGY_PRICE

```

CREATE TABLE [dbo].[FACT_ENERGY_PRICE] (
    [pk_id] int NOT NULL,
    [fk_date_semester] nvarchar(10),
    [fk_country] int,
    [fk_currency] int,
    [fk_tax] int,
    [fk_consumption] int,
    [fk_unit] int,
    [fk_product] int,
    [price] numeric(12,4)
    CONSTRAINT [PK_FACT_ENERGY_PRICE] PRIMARY KEY CLUSTERED
    (
        [pk_id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

```

--Claves foráneas

```

ALTER TABLE FACT_EUROZONE_INDICATORS ADD FOREIGN KEY (fk_date) REFERENCES DIM_DATE(pk_date);
ALTER TABLE FACT_EUROZONE_INDICATORS ADD FOREIGN KEY (fk_country) REFERENCES DIM_COUNTRY(pk_country);
ALTER TABLE FACT_EUROZONE_INDICATORS ADD FOREIGN KEY (fk_coicop) REFERENCES DIM_COICOP(pk_coicop);
ALTER TABLE FACT_ENERGY_PRICE ADD FOREIGN KEY (fk_date_semester) REFERENCES DIM_DATE_SEMESTER(pk_date_semester);
ALTER TABLE FACT_ENERGY_PRICE ADD FOREIGN KEY (fk_country) REFERENCES DIM_COUNTRY(pk_country);
ALTER TABLE FACT_ENERGY_PRICE ADD FOREIGN KEY (fk_currency) REFERENCES DIM_CURRENCY(pk_currency);
ALTER TABLE FACT_ENERGY_PRICE ADD FOREIGN KEY (fk_tax) REFERENCES DIM_TAX(pk_tax);
ALTER TABLE FACT_ENERGY_PRICE ADD FOREIGN KEY (fk_consumption) REFERENCES DIM_CONSUMPTION(pk_consumption);
ALTER TABLE FACT_ENERGY_PRICE ADD FOREIGN KEY (fk_unit) REFERENCES DIM_UNIT(pk_unit);
ALTER TABLE FACT_ENERGY_PRICE ADD FOREIGN KEY (fk_product) REFERENCES DIM_PRODUCT(pk_product);

```

2.2. Transformaciones Bloque IN

Una vez implementado el modelo físico del almacén, se realizarán las transformaciones correspondientes del Bloque IN. El objetivo de estas transformaciones es cargar la información del origen a la tabla correspondiente del área intermedia (staging área).

Para realizar las transformaciones se utilizarán las variables de entorno creadas en la práctica anterior:

#	Variable name	Value
1	CN_DW	jdbc:sqlserver://UCS1R1UOCSQL01:1433;databaseName=SOURCE_gmanresas; integratedSecurity=false
2	CN_STAGE	jdbc:sqlserver://UCS1R1UOCSQL01:1433;databaseName=SOURCE_gmanresas; integratedSecurity=false
3	DIR_IN	FAPRA 2

2.2.1. Transformación IN_COICOP

La transformación de «IN_COICOP» contiene las siguientes etapas:

- lectura del fichero .xml,
- filtración de valores nulos,
- carga a la tabla intermedia «STG_COICOP».

A continuación, se muestran las capturas de pantalla correspondientes a estos pasos:

Lectura del fichero .xml

Utilizamos la transformación «Get data from XML» para extraer los datos del origen. En este paso se indica el fichero desde donde se extraen los datos utilizando la variable de entorno «DIR_IN».



Get data from XML

Get data from XML

Step name: Get data from XML

File Content Fields Additional output fields

XML source from field

XML source is defined in a field? XML source is a filename? Read source as URL get XML source from a field

File or directory: \${DIR_IN}/COICOP.xml Add Browse

Regular Expression:

Exclude Regular Expression:

#	File/Directory	Wildcard (RegExp)	Exclude wildcard	Required	Include subfolders
1	\${DIR_IN}/COICOP.xml			N	N

Delete Edit

En la pestaña “Content” indicamos el «Loop XPath» e ignoramos los posibles comentarios que haya dentro del fichero:

Get data from XML

Step name: Get data from XML

File Content Fields Additional output fields

Settings

Loop XPath: /root/row Get XPath nodes

Encoding: UTF-8

Namespace aware? Ignore comments? Validate XML? Use token Ignore empty file Do not raise an error if no files Limit: 0 Prune path to handle large files

Additional fields

Include filename in output? Filename fieldname:
Rownum in output? Rownum fieldname:

En la pestaña “Fields” seleccionamos qué campos queremos traernos a Spoon mediante el botón de «Get fields»:

#	Name	XPath	Element	Result type	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type	Repeat
1	Code	Code	Node	Value of	String							none	N
2	Description	Description	Node	Value of	String							none	N

Get fields

Podemos realizar una visualización previa de los datos que se cargarán con el botón «Preview rows»:

Rows of step: Get data from XML (962 rows)		
#	Code	Description
1	TOTAL	Total
2	TOTAL_1	Total excluding rents
3	CP01_TO_12	Total individual consumption expenditure by households
4	CP00	All-items HICP
5	00FLASH	Flash estimate - All items HICP
6	CP01	Food and non-alcoholic beverages
7	CP011	Food
8	CP0111	Bread and cereals
9	CP01111	Rice
10	CP01112	Flours and other cereals

Filtración de valores nulos

Para eliminar los valores nulos (ya que la tabla que hemos creado en nuestra base de datos SQL («STG_COICOP») no los admite) deberemos filtrarlos mediante el paso «Filter rows»:



Filter rows

The condition:
Code IS NOT NULL

Carga a la tabla intermedia «STG_COICOP»

Finalmente, ya se puede realizar la carga en la tabla intermedia del stage, utilizando el componente «Table Output».



Table output

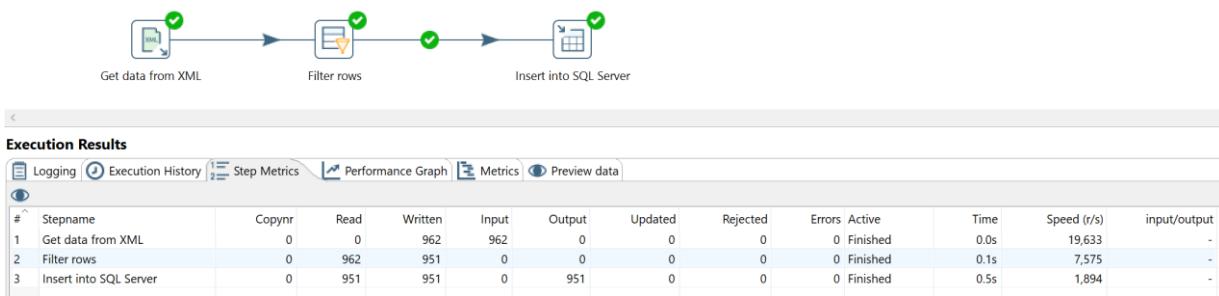
Para ello debemos utilizar la conexión creada, CN_STG, con la variable de entorno. También se debe hacer el mapeo de los campos e indicar “Truncate table” para no duplicar los datos:

Main options Database fields

#	Table field	Stream field
1	Code	Code
2	Description	Description

Get fields Enter field mapping

El proceso de la transformación completa es el siguiente:



Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:

The screenshot shows a SQL query in the SSMS interface. The query selects top 1000 rows from the table [SOURCE_gmanresas].[dbo].[STG_COICOP]. The results are displayed in a grid, showing columns 'code' and 'description'. The results include various categories like 'TOTAL', 'CP01_TO_12', and specific food items like 'Food and non-alcoholic beverages'.

code	description
TOTAL	Total
TOTAL_1	Total excluding rents
CP01_TO_12	Total individual consumption expenditure by hours
CP00	All-items HICP
00FLASH	Flash estimate - All items HICP
CP01	Food and non-alcoholic beverages
CP011	Food
CP0111	Bread and cereals
CP01111	Rice
CP01112	Flours and other cereals

2.2.2. Transformación IN_COUNTRY

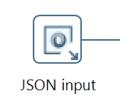
La transformación de « IN_COUNTRY » contiene las siguientes etapas:

- lectura del fichero .json,
- “string operations”,
- carga a la tabla intermedia «STG_COUNTRY ».

A continuación, se muestran las capturas de pantalla correspondientes a estos pasos:

Lectura del fichero .json

Utilizamos la transformación «JSON input» para extraer los datos del origen. En este paso se indica el fichero desde donde se extraen los datos utilizando la variable de entorno «DIR_IN».



JSON input

Step name: JSON input

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)	Exclude wildcard	Required	Include subfolders
1 \${DIR_IN}\countries.json				N	N

Por un error en el fichero json (faltan las comillas) es necesario en la pestaña “Fields” añadir los campos de la siguiente manera:

JSON input

Step name: JSON input

File Content Fields Additional output fields

#	Name	Path	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type	Repeat
1	name	\$.name	String							none	N
2	code	\$.code	String							none	N

“String operations”

En este paso eliminamos los posibles espacios que puedan haber antes y después del campo correspondiente.



String operations

String operations

Step name: String operations

The fields to process:

#	In stream field	Out stream field	Trim type	Lower/Upper	Padding	Pad char	Pad Length	InitCap	Escape	Digits	Remove Special character
1	name		both	none	none			N	None	none	none
2	code		both	none	none			N	None	none	none

Carga a la tabla intermedia «STG_COUNTRY»

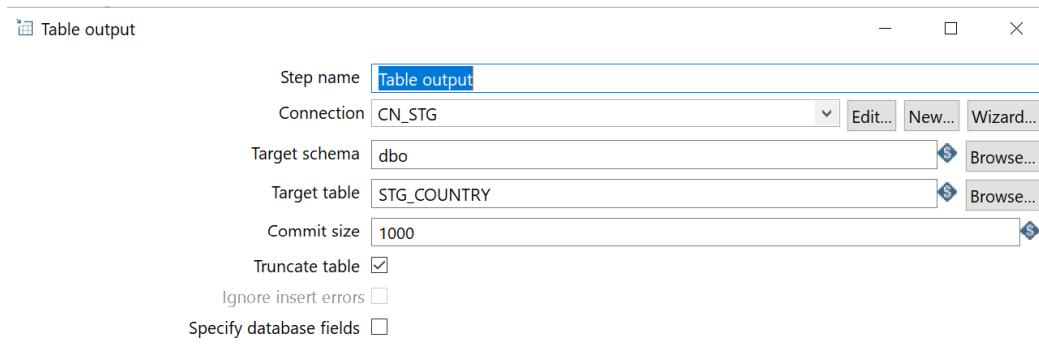
Finalmente, ya se puede realizar la carga en la tabla intermedia del stage, utilizando el componente «Table Output».

Para ello debemos utilizar la conexión creada, CN_STG, con la variable de entorno. También se debe indicar “Truncate table” para no duplicar los datos. En este caso no

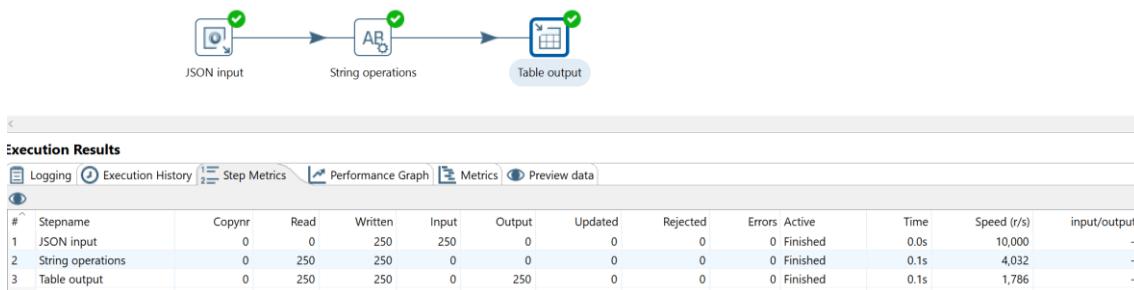
es necesario hacer el mapeo de los campos ya que las columnas tienen el mismo nombre.



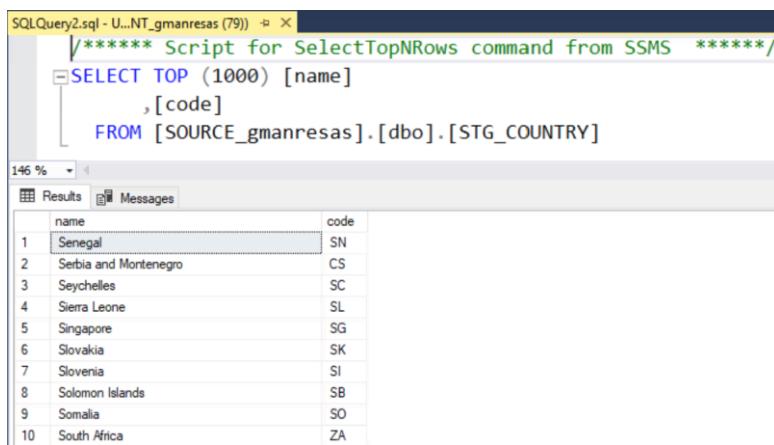
Table output



El proceso de la transformación completa es el siguiente:



Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:



```
SQLQuery2.sql - U...NT_gmanresas (79)  ▶ X
***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [name]
,[code]
FROM [SOURCE_gmanresas].[dbo].[STG_COUNTRY]
```

	name	code
1	Senegal	SN
2	Serbia and Montenegro	CS
3	Seychelles	SC
4	Sierra Leone	SL
5	Singapore	SG
6	Slovakia	SK
7	Slovenia	SI
8	Solomon Islands	SB
9	Somalia	SO
10	South Africa	ZA

2.2.3. Transformación IN_HICP

La transformación de « IN_HICP » contiene las siguientes etapas:

- lectura del fichero .csv,
- “Split fields”,
- “Row normaliser”,
- “Replace in string”,
- “String operations”,
- “Select values”,
- carga a la tabla intermedia «STG_ HICP ».

A continuación, se muestran las capturas de pantalla correspondientes a estos pasos:

Lectura del fichero .csv

Utilizamos la transformación «CSV file input» para extraer los datos del origen. En este paso se indica el fichero desde donde se extraen los datos utilizando la variable de entorno «DIR_IN».



CSV file input

Se debe especificar que el delimitador es un tabulador y se obtienen 37 campos:

The screenshot shows the configuration window for the 'CSV file input' step. The 'Step name' is set to 'CSV file input'. The 'Filename' field contains the environment variable '\$(DIR_IN)\prc_hicp_mv\2r_tabular.tsv'. Other settings include 'Lazy conversion?' checked, 'Header row present?' checked, and 'Add filename to result' unchecked. Below the configuration, a preview table shows 10 rows of data with columns for '#', 'Name', 'Type', 'Format', 'Length', 'Precision', 'Currency', 'Decimal', 'Group', and 'Trim type'. The 'Name' column lists fields like 'freq,unit,coicop,geo\TIME_PERIOD' and dates from '2020-01' to '2020-09'. The 'Type' column shows all as 'String'. The 'Format' column is 'mixed'. The 'Length' column varies between 6 and 25. The 'Precision' column is 0. The 'Currency' and 'Decimal' columns are empty. The 'Group' and 'Trim type' columns are 'none'.

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type
1	freq,unit,coicop,geo\TIME_PERIOD	String		25	0				none
2	2020-01	String		6	0				none
3	2020-02	String		6	0				none
4	2020-03	String		6	0				none
5	2020-04	String		6	0				none
6	2020-05	String		6	0				none
7	2020-06	String		6	0				none
8	2020-07	String		6	0				none
9	2020-08	String		6	0				none
10	2020-09	String		6	0				none

“Split fields”

Observamos en la imagen anterior que el primer campo es “freq,unit,coicop,geo\TIME_PERIOD” este campo lo vamos a dividir en varios con la transformación “split fields”.



Split fields

Se especifica que el delimitador es una coma “,” y se especifican los nuevos campos.

#	New field	ID	Remove ID?	Type	Length	Precision	Format	Group	Decimal	Currency	Nullif	Default	Trim type
1	freq	N		String									none
2	unit	N		String									none
3	coicop	N		String									none
4	geo	N		String									none

El resultado de esta transformación es:

Execution Results														
Logging Execution History Step Metrics Performance Graph Metrics Preview data														
#	freq	unit	coicop	geo	2020-01	2020-02	2020-03	2020-04	2020-05	2020-06	2020-07	2020-08	2020-09	2020-10
1	M	RCH_MV12MAVR	AP	AT	2.2	2.2	2.2	2.2	2.2	2.2	2.2	2.2	2.2	2.2
2	M	RCH_MV12MAVR	AP	BE	1.7	1.7	1.7	1.7	1.6	1.6	1.5	1.5	1.5	1.4
3	M	RCH_MV12MAVR	AP	BG	2.6	2.6	2.6	2.5	2.4	2.3	2.3	2.2	2.0	1.9
4	M	RCH_MV12MAVR	AP	CH	:	:	:	:	:	:	:	:	:	:
5	M	RCH_MV12MAVR	AP	CY	2.1	1.8	1.7	1.5	1.3	0.8	0.2	-0.1	-0.5	-0.8
6	M	RCH_MV12MAVR	AP	CZ	3.9	4.1	4.2	4.2	4.2	4.2	4.3	4.3	4.2	4.1
7	M	RCH_MV12MAVR	AP	DE	1.6	1.6	1.6	1.7	1.7	1.7	1.7	1.7	1.7	1.6
8	M	RCH_MV12MAVR	AP	DK	1.4	1.4	1.4	1.5	1.5	1.6	1.6	1.6	1.7	1.7

“Row normaliser”



Con el componente “row normaliser” pasamos los campos correspondientes a un periodo de tiempo a un nuevo campo llamado “period”. El contenido de los campos anteriores lo añadimos a un nuevo campo llamado “HICP”.

#	Fieldname	Type	new field
1	2020-01	202001	HICP
2	2020-02	202002	HICP
3	2020-03	202003	HICP
4	2020-04	202004	HICP
5	2020-05	202005	HICP
6	2020-06	202006	HICP
7	2020-07	202007	HICP
8	2020-08	202008	HICP
9	2020-09	202009	HICP
10	2020-10	202010	HICP
11	2020-11	202011	HICP
12	2020-12	202012	HICP
13	2021-01	202101	HICP

De manera que el resultado es:

Execution Results						
	Logging	Execution History	Step Metrics	Performance Graph	Metrics	Preview data
#	freq	unit	coicop	geo	period	HICP
1	M	RCH_MV12MAVR	AP	AT	202001	2.2
2	M	RCH_MV12MAVR	AP	AT	202002	2.2
3	M	RCH_MV12MAVR	AP	AT	202003	2.2
4	M	RCH_MV12MAVR	AP	AT	202004	2.2
5	M	RCH_MV12MAVR	AP	AT	202005	2.2
6	M	RCH_MV12MAVR	AP	AT	202006	2.2
7	M	RCH_MV12MAVR	AP	AT	202007	2.2
8	M	RCH_MV12MAVR	AP	AT	202008	2.2
9	M	RCH_MV12MAVR	AP	AT	202009	2.2
10	M	RCH_MV12MAVR	AP	AT	202010	2.2
11	M	RCH_MV12MAVR	AP	AT	202011	2.2

“Replace in string”



El campo HICP que debe ser numérico aparecen letras en algunos registros. Con el componente “replace in string” eliminamos cualquier letra de este campo. También se eliminan los dos puntos “.” para obtener valores “NULL” en su lugar.

“String operations”

En este paso eliminamos los posibles espacios que pueda haber antes y después del campo correspondiente especificando “Trim type = both”.

“Select values”



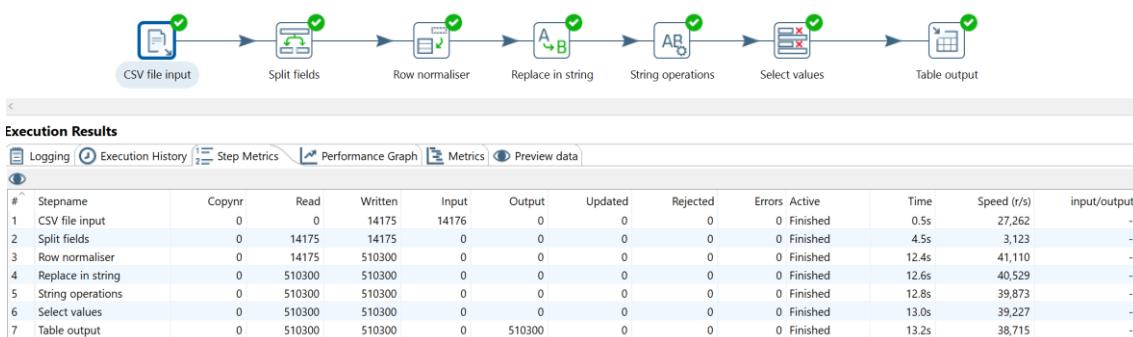
Con el componente “select values” pasamos el campo HICP a numeric, especificando el punto “.” como separador decimal y la coma “,” se especifica en “grouping” para especificar el separador de miles.

Carga a la tabla intermedia «STG_HICP»

Finalmente, ya se puede realizar la carga en la tabla intermedia del stage, utilizando el componente «Table Output».

Para ello debemos utilizar la conexión creada, CN_STG, con la variable de entorno. También se debe indicar “Truncate table” para no duplicar los datos. En este caso no es necesario hacer el mapeo de los campos ya que las columnas tienen el mismo nombre.

El proceso de la transformación completa es el siguiente:



Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:

```

SQLQuery3.sql - U...NT_gmanresas (86)  x
***** Script for SelectTopNRows command from SSMS *****/
--SELECT TOP (1000) [freq]
--,[unit]
--,[colcop]
--,[geo]
--,[period]
--,[HICP]
FROM [SOURCE_gmanresas].[dbo].[STG_HICP]

146 %  ▾
Results  Messages
freq unit colcop geo period HICP
1 M RCH_MV12MAVR AP CH 202102 -0.60
2 M RCH_MV12MAVR AP ES 202006 -0.90
3 M RCH_MV12MAVR AP IT 202210 18.00
4 M RCH_MV12MAVR AP NL 202202 6.80
5 M RCH_MV12MAVR AP SI 202106 1.40
6 M RCH_MV12MAVR APF CY 202007 -1.20
7 M RCH_MV12MAVR APF ES 202207 4.10
8 M RCH_MV12MAVR APF LT 202107 4.40
9 M RCH_MV12MAVR APF NO 202007 4.10

```

2.2.4. Transformación IN_P GAS

La transformación de « IN_P GAS » contiene las siguientes etapas:

- lectura del fichero .csv,
- “Split fields”,
- “Row normaliser”,
- “Replace in string”,
- “String operations”,
- “Select values”,
- carga a la tabla intermedia «STG_ PGAS ».

A continuación, se muestran las capturas de pantalla correspondientes a estos pasos:

Lectura del fichero .csv

Utilizamos la transformación «CSV file input» para extraer los datos del origen. En este paso se indica el fichero desde donde se extraen los datos utilizando la variable de entorno «DIR_IN».

Se debe especificar que el delimitador es un tabulador y se obtienen 32 campos:

Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type
freq	String		41	\$,	.		none
product	String		9	\$,	.		none
consum	String		10	\$,	.		none
unit	String		10	\$,	.		none
tax	String		10	\$,	.		none
currency	String		10	\$,	.		none
geo	String		10	\$,	.		none
TIME_PERIOD	String		10	\$,	.		none
2007-51	String		9	\$,	.		none
2007-52	String		10	\$,	.		none
2008-51	String		10	\$,	.		none
2008-52	String		10	\$,	.		none
2009-51	String		10	\$,	.		none
2009-52	String		10	\$,	.		none
2010-51	String		10	\$,	.		none
2010-52	String		10	\$,	.		none

“Split fields”

Observamos en la imagen anterior que el primer campo es “freq,product,consum,unit,tax,currency,geo\TIME_PERIOD” este campo lo vamos a dividir en varios con la transformación “split fields”.

Se especifica que el delimitador es una coma “,” y se especifican los nuevos campos.

#	New field	ID	Remove ID?	Type	Length	Precision	Format	Group	Decimal	Currency	Nullif	Default	Trim type
1	freq		N	String									none
2	product		N	String									none
3	consum		N	String									none
4	unit		N	String									none
5	tax		N	String									none
6	currency		N	String									none
7	geo		N	String									none

El resultado de esta transformación es:

Execution Results															
#	freq	product	consum	unit	tax	currency	geo	2007-S1	2007-S2	2008-S1	2008-S2	2009-S1	2009-S2	2010-S1	2010-S2
1	S	4100	4141901	GJ_GCV	I_TAX	EUR	AT	:	20.9300	20.2600	20.9000	21.8200	21.0500	20.9900	20.4500
2	S	4100	4141901	GJ_GCV	I_TAX	EUR	BA	:	:	:	:	:	:	:	:
3	S	4100	4141901	GJ_GCV	I_TAX	EUR	BE	:	21.4800	23.8800	28.1300	24.4400	21.9200	22.1700	23.7200
4	S	4100	4141901	GJ_GCV	I_TAX	EUR	BG	:	8.5131	8.9938	11.5912	13.2375	9.3875	10.0931	11.7305
5	S	4100	4141901	GJ_GCV	I_TAX	EUR	CZ	:	15.1113	18.2522	21.0187	19.9823	19.6196	20.5834	22.0641
6	S	4100	4141901	GJ_GCV	I_TAX	EUR	DE	:	25.9800	26.2200	30.1900	27.8800	26.2500	25.3800	28.2500
7	S	4100	4141901	GJ_GCV	I_TAX	EUR	DK	22.6474	24.5874	26.6286	27.2065	22.2989	23.6401	26.4979	26.8147
8	S	4100	4141901	GJ_GCV	I_TAX	EUR	EA	22.3650	23.9777	23.8256	28.0805	25.5991	25.0153	23.4810	27.9344
9	S	4100	4141901	GJ_GCV	I_TAX	EUR	EE	:	10.7211	10.8941	11.2987	10.4182	10.9148	14.2668	
10	S	4100	4141901	GJ_GCV	I_TAX	EUR	EL	:	:	:	:	:	:	:	

“Row normaliser”

Con el componente “row normaliser” pasamos los campos correspondientes a un periodo de tiempo a un nuevo campo llamado “period”. El contenido de los campos anteriores lo añadimos a un nuevo campo llamado “pgas”.

#	Fieldname	Type	new field
1	2007-S1	2007-S1	pgas
2	2007-S2	2007-S2	pgas
3	2008-S1	2008-S1	pgas
4	2008-S2	2008-S2	pgas
5	2009-S1	2009-S1	pgas
6	2009-S2	2009-S2	pgas
7	2010-S1	2010-S1	pgas
8	2010-S2	2010-S2	pgas
9	2011-S1	2011-S1	pgas
10	2011-S2	2011-S2	pgas
11	2012-S1	2012-S1	pgas
12	2012-S2	2012-S2	----

De manera que el resultado es:

Execution Results

#	freq	product	consum	unit	tax	currency	geo	period	pgas
1	S	4100	4141901	GJ_GCV	I_TAX	EUR	AT	2007-S1	:
2	S	4100	4141901	GJ_GCV	I_TAX	EUR	AT	2007-S2	20.9300
3	S	4100	4141901	GJ_GCV	I_TAX	EUR	AT	2008-S1	20.2600
4	S	4100	4141901	GJ_GCV	I_TAX	EUR	AT	2008-S2	20.9000
5	S	4100	4141901	GJ_GCV	I_TAX	EUR	AT	2009-S1	21.8200
6	S	4100	4141901	GJ_GCV	I_TAX	EUR	AT	2009-S2	21.0500
7	S	4100	4141901	GJ_GCV	I_TAX	EUR	AT	2010-S1	20.9900
8	S	4100	4141901	GJ_GCV	I_TAX	EUR	AT	2010-S2	20.4500
9	S	4100	4141901	GJ_GCV	I_TAX	EUR	AT	2011-S1	23.1300

“Replace in string”

El campo “pgas” que debe ser numérico aparecen letras en algunos registros. Con el componente “replace in string” eliminamos cualquier letra de este campo. También se eliminan los dos puntos “:” para obtener valores “NULL” en su lugar.

#	In stream field	Out stream field	use RegEx	Search	Replace with	Set empty string?	Replace with field	Whole Word	Case sensitive	Is Unicode
1	pgas		Y	[a-zA-Z]		N		N	N	N

“String operations”

En este paso eliminamos los posibles espacios que pueda haber antes y después del campo correspondiente especificando “Trim type = both”.

#	In stream field	Out stream field	Trim type	Lower/Upper	Padding	Pad char	Pad Length	InitCap	Escape	Digits	Remove Special character
1	freq		both	none	none			N	None	none	none
2	product		both	none	none			N	None	none	none
3	consum		both	none	none			N	None	none	none
4	unit		both	none	none			N	None	none	none
5	tax		both	none	none			N	None	none	none
6	currency		both	none	none			N	None	none	none
7	geo		both	none	none			N	None	none	none
8	period		both	none	none			N	None	none	none
9	pgas		both	none	none			N	None	none	none

“Select values”

Con el componente “select values” pasamos el campo “pgas” a numeric, especificando el punto “.” como separador decimal y la coma “,” se especifica en “grouping” para especificar el separador de miles.

#	Fieldname	Type	Length	Precision	Binary type	Format	Date Format	Date Locale	Date Time Zone	Lenient number conversion	Encoding	Decimal	Grouping	Currency
1	pgas	Number			N	#,##,##,##	N			N		.	,	

Carga a la tabla intermedia «STG_PGAS»

Finalmente, ya se puede realizar la carga en la tabla intermedia del stage, utilizando el componente «Table Output».

Para ello debemos utilizar la conexión creada, CN_STG, con la variable de entorno. También se debe indicar “Truncate table” para no duplicar los datos. En este caso no es necesario hacer el mapeo de los campos ya que las columnas tienen el mismo nombre.

Step name: Table output

Connection: CN_STG

Target schema: dbo

Target table: STG_PGAS

Commit size: 1000

Truncate table:

Ignore insert errors:

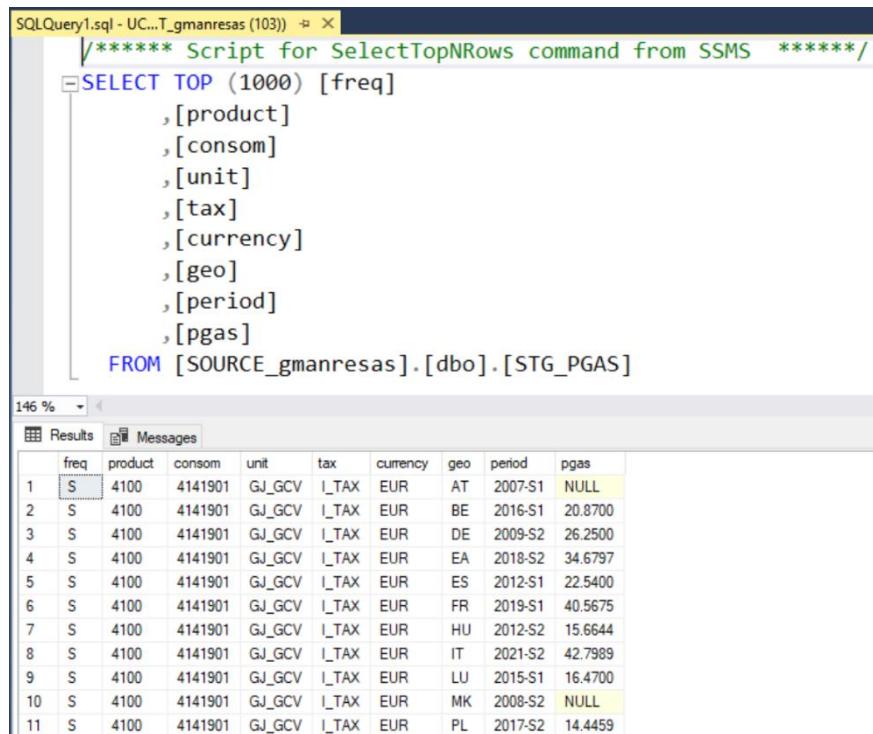
Specify database fields:

El proceso de la transformación completa es el siguiente:



#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)	input/output
1	CSV file input	0	0	1800	1801	0	0	0	0	Finished	0.0s	64,321	-
2	Split fields	0	1800	1800	0	0	0	0	0	Finished	0.1s	12,329	-
3	Row normaliser	0	1800	55800	0	0	0	0	0	Finished	0.9s	59,425	-
4	Replace in string	0	55800	55800	0	0	0	0	0	Finished	1.2s	45,073	-
5	String operations	0	55800	55800	0	0	0	0	0	Finished	1.4s	38,509	-
6	Select values	0	55800	55800	0	0	0	0	0	Finished	1.7s	32,670	-
7	Table output	0	55800	55800	0	55800	0	0	0	Finished	1.9s	29,199	-

Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:



```
SQLQuery1.sql - UC...T_gmanresas (103) ×
*****
 Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [freq]
    ,[product]
    ,[consum]
    ,[unit]
    ,[tax]
    ,[currency]
    ,[geo]
    ,[period]
    ,[pgas]
FROM [SOURCE_gmanresas].[dbo].[STG_PGAS]
```

The screenshot shows a SQL query window with the following content:

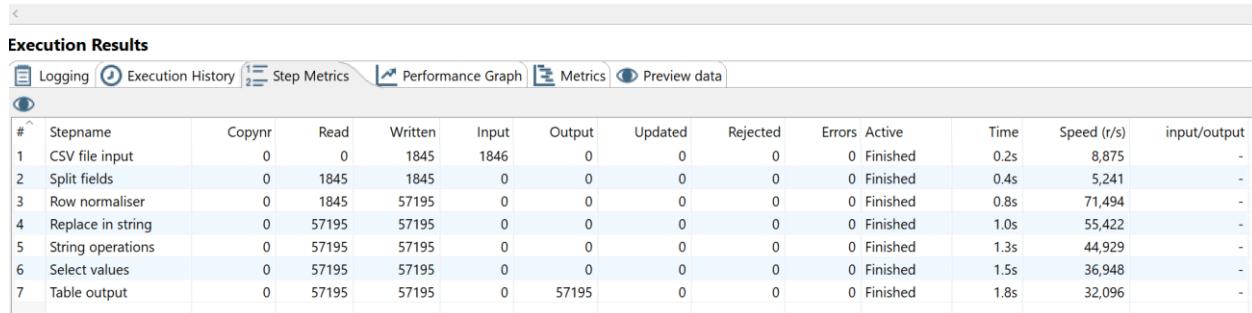
```
146 % ▾
Results Messages
```

	freq	product	consum	unit	tax	currency	geo	period	pgas
1	S	4100	4141901	GJ_GCV	I_TAX	EUR	AT	2007-S1	NULL
2	S	4100	4141901	GJ_GCV	I_TAX	EUR	BE	2016-S1	20.8700
3	S	4100	4141901	GJ_GCV	I_TAX	EUR	DE	2009-S2	26.2500
4	S	4100	4141901	GJ_GCV	I_TAX	EUR	EA	2018-S2	34.6797
5	S	4100	4141901	GJ_GCV	I_TAX	EUR	ES	2012-S1	22.5400
6	S	4100	4141901	GJ_GCV	I_TAX	EUR	FR	2019-S1	40.5675
7	S	4100	4141901	GJ_GCV	I_TAX	EUR	HU	2012-S2	15.6644
8	S	4100	4141901	GJ_GCV	I_TAX	EUR	IT	2021-S2	42.7989
9	S	4100	4141901	GJ_GCV	I_TAX	EUR	LU	2015-S1	16.4700
10	S	4100	4141901	GJ_GCV	I_TAX	EUR	MK	2008-S2	NULL
11	S	4100	4141901	GJ_GCV	I_TAX	EUR	PL	2017-S2	14.4459

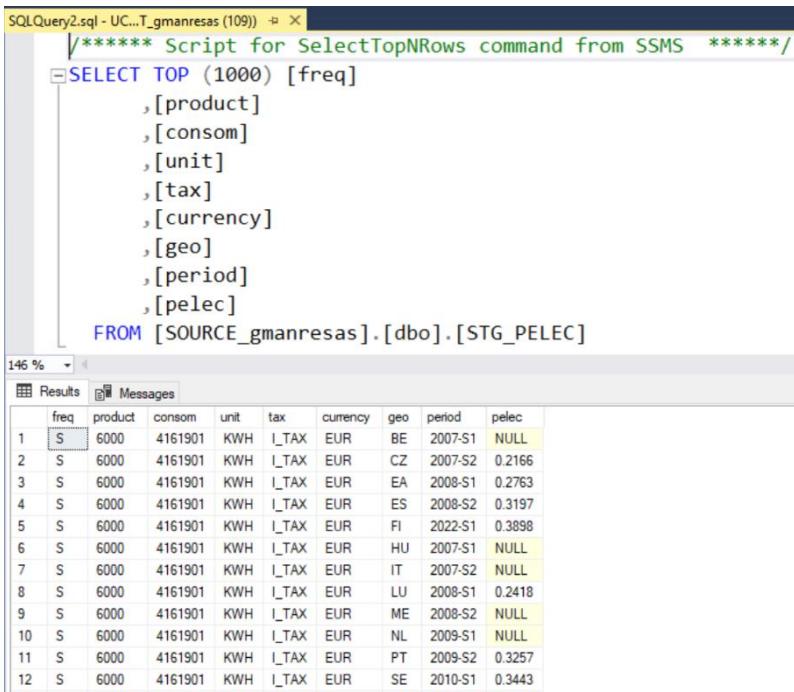
2.2.5. Transformación IN_PELEC

La transformación IN_PELEC es igual a la anterior ya que los archivos de origen son prácticamente iguales en cuanto a estructura se refiere.

A continuación, observamos la transformación completa:



Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:



```
SQLQuery2.sql - UC...T_gmanresas (109)  ↳ X
/****** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [freq]
,[product]
,[consum]
,[unit]
,[tax]
,[currency]
,[geo]
,[period]
,[pelec]
FROM [SOURCE_gmanresas].[dbo].[STG_PELEC]
```

The screenshot shows a SQL query in the top pane and its results in the bottom pane. The results grid has columns: freq, product, consum, unit, tax, currency, geo, period, and pelec. The data consists of 12 rows, each with values corresponding to the columns.

	freq	product	consum	unit	tax	currency	geo	period	pelec
1	S	6000	4161901	KWH	I_TAX	EUR	BE	2007-S1	NULL
2	S	6000	4161901	KWH	I_TAX	EUR	CZ	2007-S2	0.2166
3	S	6000	4161901	KWH	I_TAX	EUR	EA	2008-S1	0.2763
4	S	6000	4161901	KWH	I_TAX	EUR	ES	2008-S2	0.3197
5	S	6000	4161901	KWH	I_TAX	EUR	FI	2022-S1	0.3898
6	S	6000	4161901	KWH	I_TAX	EUR	HU	2007-S1	NULL
7	S	6000	4161901	KWH	I_TAX	EUR	IT	2007-S2	NULL
8	S	6000	4161901	KWH	I_TAX	EUR	LU	2008-S1	0.2418
9	S	6000	4161901	KWH	I_TAX	EUR	ME	2008-S2	NULL
10	S	6000	4161901	KWH	I_TAX	EUR	NL	2009-S1	NULL
11	S	6000	4161901	KWH	I_TAX	EUR	PT	2009-S2	0.3257
12	S	6000	4161901	KWH	I_TAX	EUR	SE	2010-S1	0.3443

2.2.6. Transformación IN_CONSUMPTION

La transformación de « IN_PGAS » contiene las siguientes etapas:

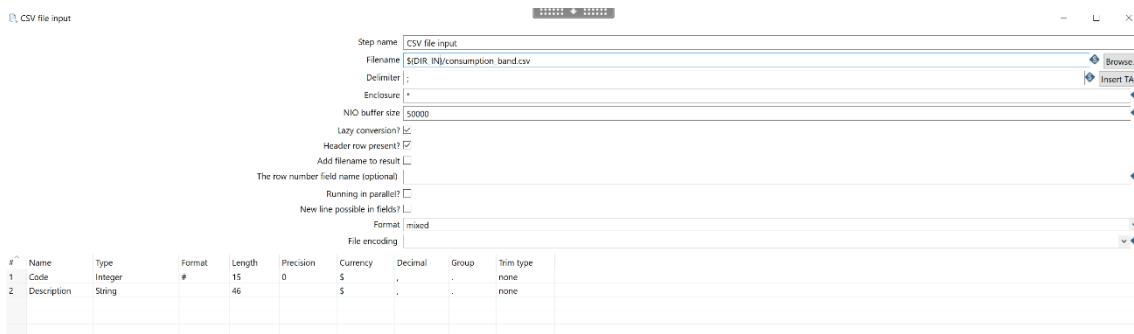
- lectura del fichero .csv,
- carga a la tabla intermedia «STG_CONSUMPTION ».

A continuación, se muestran las capturas de pantalla correspondientes a estos pasos:

Lectura del fichero .csv

Utilizamos la transformación «CSV file input» para extraer los datos del origen. En este paso se indica el fichero desde donde se extraen los datos utilizando la variable de entorno «DIR_IN».

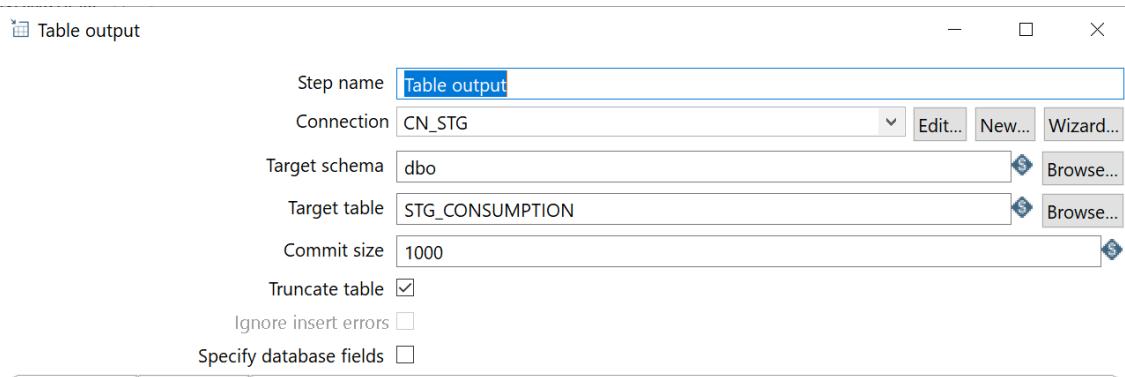
Se debe especificar que el delimitador es punto y coma ";" y se obtienen 2 campos:



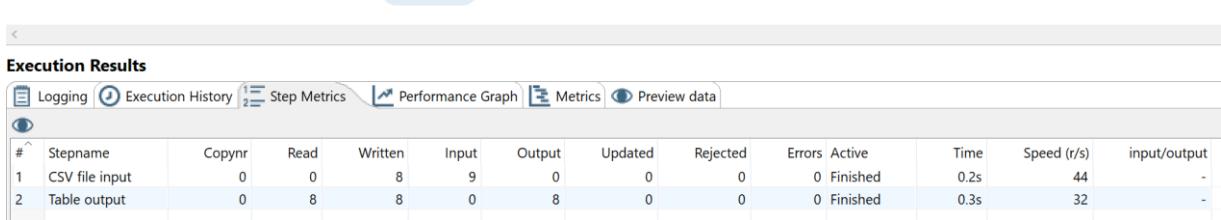
Carga a la tabla intermedia «STG_CONSUMPTION»

Finalmente, ya se puede realizar la carga en la tabla intermedia del stage, utilizando el componente «Table Output».

Para ello debemos utilizar la conexión creada, CN_STG, con la variable de entorno. También se debe indicar “Truncate table” para no duplicar los datos. En este caso no es necesario hacer el mapeo de los campos ya que las columnas tienen el mismo nombre.



El proceso de la transformación completa es el siguiente:



Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:

The screenshot shows a SQL query in the 'SQLQuery3.sql' window. The query selects top 1000 rows from the 'STG_CONSUMPTION' table. The results are displayed in a table with columns 'code' and 'description'.

	code	description
1	4161901	Band DA : Consumption < 1 000 kWh
2	4161902	Band DB : 1 000 kWh < Consumption < 2 500 kWh
3	4161903	Band DC : 2 500 kWh < Consumption < 5 000 kWh
4	4161904	Band DD : 5 000 kWh < Consumption < 15 000 kWh
5	4161905	Band DE : Consumption > 15 000 kWh
6	4141901	Band D1 : Consumption < 20 GJ
7	4141902	Band D2 : 20 GJ < Consumption < 200 GJ
8	4141903	Band D3 : Consumption > 200 GJ

2.3. Transformación Bloque TR_DIM

El bloque TR_DIM 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, a las dimensiones del modelo multidimensional del almacén.

2.3.1. Transformación TR_DIM_COUNTRY

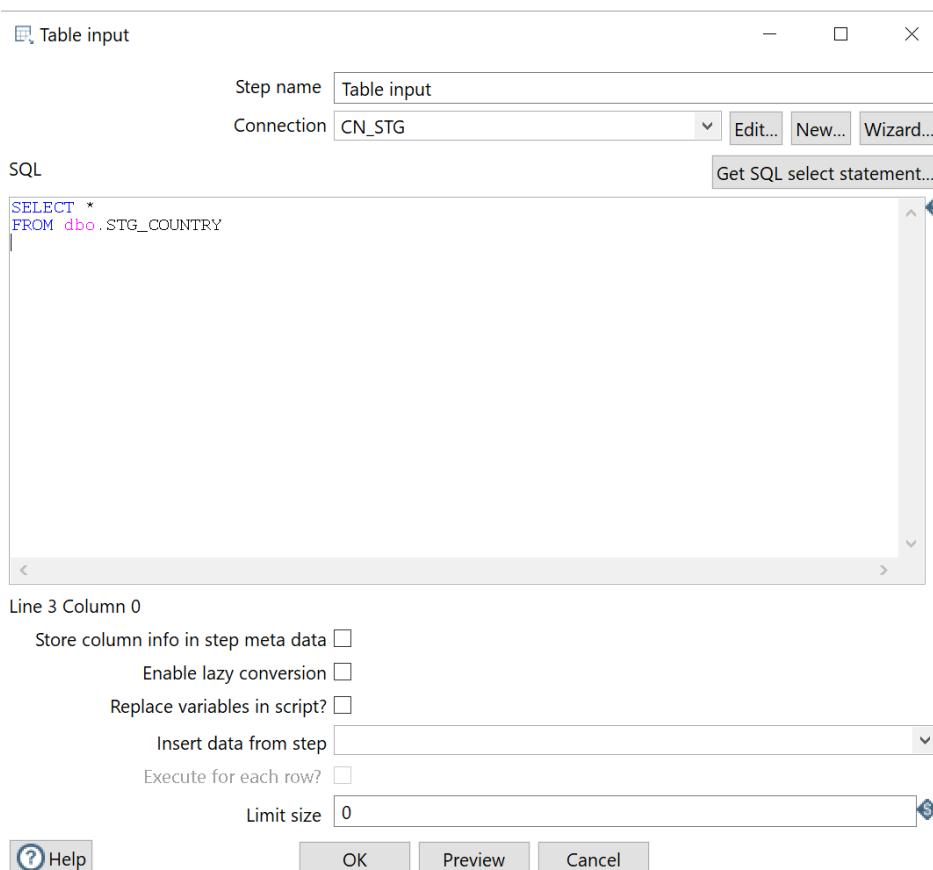
La transformación de «DIM_COUNTRY» contiene las siguientes etapas:

- “Table input”,
- “Sort rows”,
- “Add sequence”,
- “Table output”

A continuación, se muestran las capturas de pantalla correspondientes a estos pasos:

“Table input”

Mediante una sentencia «SELECT» de SQL obtenemos la información de la tabla correspondiente utilizando la conexión creada anteriormente.



“Sort rows”

Con este componente ordenamos las filas de manera ascendente teniendo en cuenta en primer lugar la columna “name” y en segundo la columna “code”.

 Sort rows

Step name	<input type="text" value="Sort rows"/>						
Sort directory	<input type="text" value="%%java.io.tmpdir%%"/>						
TMP-file prefix	<input type="text" value="out"/>						
Sort size (rows in memory)	<input type="text" value="1000000"/>						
Free memory threshold (in %)	<input type="text"/>						
Compress TMP Files? <input checked="" type="checkbox"/>							
Only pass unique rows? (verifies keys only) <input type="checkbox"/>							
Fields :							
#	Fieldname	Ascending	Case sensitive compare?	Sort based on current locale?	Collator Strength	Presorted?	
1	name	Y	N	N	0	N	
2	code	Y	N	N	0	N	

“Add sequence”

Con el componente “Add sequence” creamos una secuencia que hará las funciones de clave primaria de incremento automático.

 Add sequence

Step name	<input type="text" value="Add sequence"/>
Name of value	<input type="text" value="pk"/>
Use a database to generate the sequence	
Use DB to get sequence?	<input type="checkbox"/>
Connection	<input type="text" value="CN_STG"/> <input type="button" value="Edit..."/> <input type="button" value="New..."/> <input type="button" value="Wizard..."/>
Schema name	<input type="text"/> <input type="button" value="Schemas..."/>
Sequence name	<input type="text" value="SEQ"/> <input type="button" value="Sequences..."/>
Use a transformation counter to generate the sequence	
Use counter to calculate sequence?	<input checked="" type="checkbox"/>
Counter name (optional)	<input type="text"/>
Start at value	<input type="text" value="1"/>
Increment by	<input type="text" value="1"/>
Maximum value	<input type="text" value="999999999"/>

“Table output”

Finalmente, cargamos los datos en la tabla de dimensión.

Para ello debemos utilizar la conexión creada, CN_DW, con la variable de entorno. También se debe hacer el mapeo de los campos e indicar “Truncate table” para no duplicar los datos:

Table output

Step name: Table output

Connection: CN_DW

Target schema: dbo

Target table: DIM_COUNTRY

Commit size: 1000

Truncate table:

Ignore insert errors:

Specify database fields:

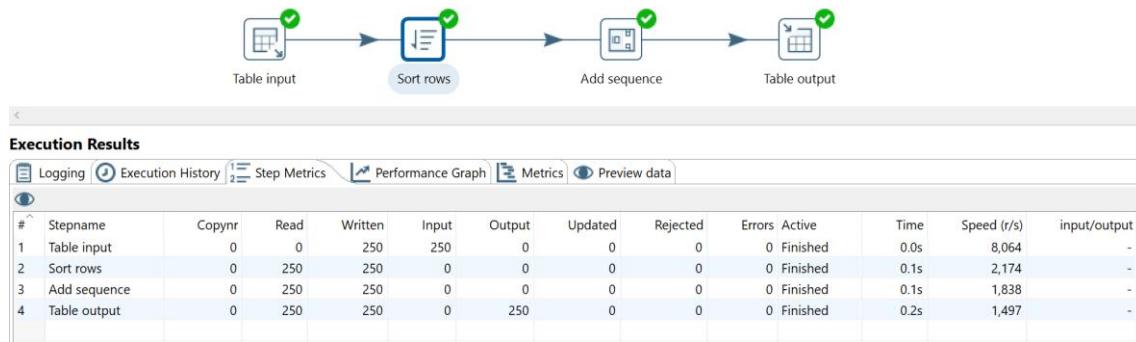
Main options Database fields

Fields to insert:

#	Table field	Stream field
1	pk_country	pk
2	country_name	name
3	code	code

Get fields Enter field mapping

El proceso de la transformación completa es el siguiente:



Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:

SQLQuery4.sql - U...NT_gmanresas (71) X

```

/*
***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [pk_country]
      ,[code]
      ,[country_name]
  FROM [SOURCE_gmanresas].[dbo].[DIM_COUNTRY]
  
```

146 % ▼

Results Messages

	pk_country	code	country_name
1	1	AF	Afghanistan
2	2	AL	Albania
3	3	DZ	Algeria
4	4	AS	American Samoa
5	5	AD	Andorra
6	6	AO	Angola
7	7	AI	Anguilla
8	8	AQ	Antarctica

2.3.2. Transformación TR_DIM_DATE

Para optimizar la actual base de datos se ha optado por cargar las dimensiones temporales únicamente con los valores existentes en el conjunto de datos. Hay que tener en cuenta que para futuras actualizaciones se deberán insertar nuevos registros.

La transformación de « DIM_DATE » contiene las siguientes etapas:

- “Table input”,
- “Table output”

A continuación, se muestran las capturas de pantalla correspondientes a estos pasos:

“Table input”

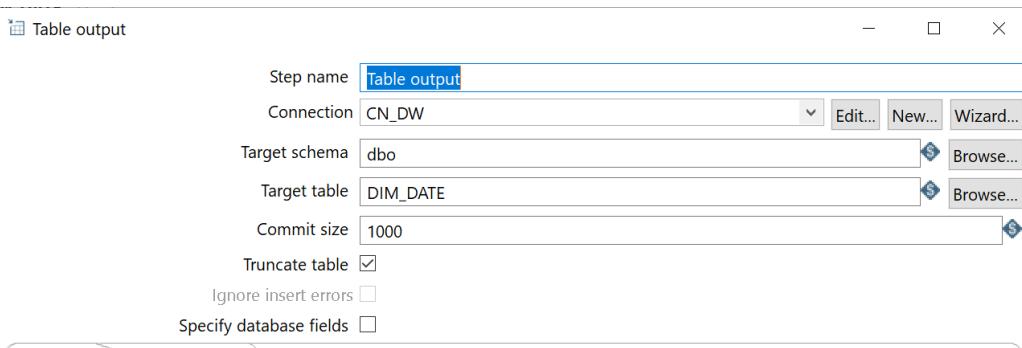
Mediante la siguiente sentencia de SQL obtenemos la información que nos interesa de la tabla correspondiente (STG_HICP) utilizando la conexión creada anteriormente.

```
Step name: Table input
Connection: CN_DW
SQL:
SELECT
    ROW_NUMBER() OVER (ORDER BY Year, Mo, Quarter) AS pk_date,
    Year,
    Mo,
    Quarter
FROM (
    SELECT DISTINCT
        CAST(SUBSTRING(CONVERT(VARCHAR(6), period), 1, 4) AS INT) AS Year,
        CAST(RIGHT(CONVERT(VARCHAR(6), period), 2) AS INT) AS Mo,
        CASE
            WHEN CAST(RIGHT(CONVERT(VARCHAR(6), period), 2) AS INT) <= 3 THEN 1
            WHEN CAST(RIGHT(CONVERT(VARCHAR(6), period), 2) AS INT) <= 6 THEN 2
            WHEN CAST(RIGHT(CONVERT(VARCHAR(6), period), 2) AS INT) <= 9 THEN 3
            ELSE 4
        END AS Quarter
    FROM dbo.STG_HICP
) AS subquery
```

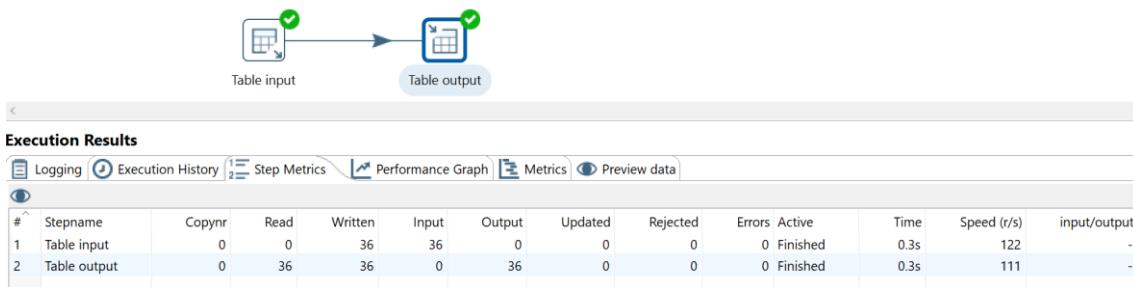
Observamos que ya hemos creado la secuencia que hará de clave primaria.

“Table output”

Finalmente, cargamos los datos en la tabla de dimensión.



El proceso de la transformación completa es el siguiente:



Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:

```
SQLQuery4.sql - U...NT_gmanresas (55) 146 % **** Script for SelectTopNRows command from SSMS ****  
SELECT TOP (1000) [pk_date]  
    ,[Year]  
    ,[Mo]  
    ,[Quarter]  
FROM [SOURCE_gmanresas].[dbo].[DIM_DATE]
```

	pk_date	Year	Mo	Quarter
1	1	2020	1	1
2	2	2020	2	1
3	3	2020	3	1
4	4	2020	4	2
5	5	2020	5	2
6	6	2020	6	2
7	7	2020	7	3
8	8	2020	8	3
9	9	2020	9	3

2.3.3. Transformación TR_DIM_DATE_SEMESTER

De la misma manera que en la transformación anterior para optimizar la actual base de datos se ha optado por cargar las dimensiones temporales únicamente con los valores existentes en el conjunto de datos. Hay que tener en cuenta que para futuras actualizaciones se deberán insertar nuevos registros.

La transformación de « DIM_DATE_SEMESTER » contiene las siguientes etapas:

- “Table input”,
- “Table output”

A continuación, se muestran las capturas de pantalla correspondientes a estos pasos:

“Table input”

Mediante la siguiente sentencia de SQL obtenemos la información que nos interesa de la tabla correspondiente (STG_PELEC) utilizando la conexión creada anteriormente.

Se ha utilizado la table STG_PELEC como origen de los datos, pero podría haber sido también STG_PGAS ya que el periodo de tiempo es el mismo.

```
Step name: Table input
Connection: CN_DW
SQL:
SELECT
    ROW_NUMBER() OVER (ORDER BY Year, Semester) AS pk_date_semester,
    Year,
    Semester
FROM (
    SELECT DISTINCT
        LEFT(CONVERT(VARCHAR(10), period), 4) AS Year,
        RIGHT(CONVERT(VARCHAR(10), period), 1) AS Semester
    FROM dbo.STG_PELEC
) AS subquery
```

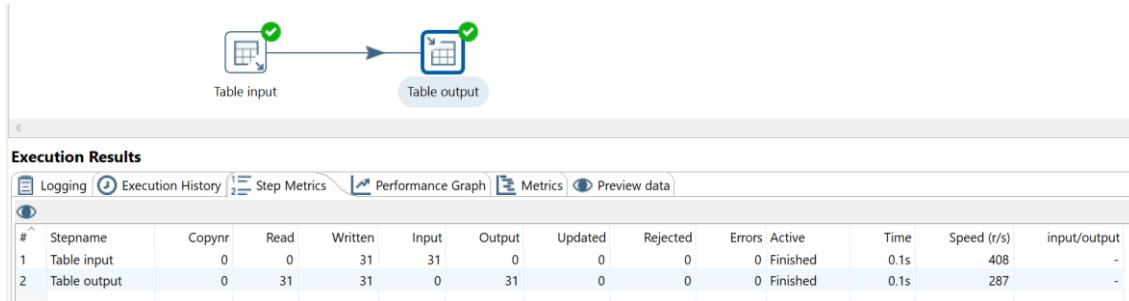
Observamos que ya hemos creado la secuencia que hará de clave primaria.

“Table output”

Finalmente, cargamos los datos en la tabla de dimensión.

```
Step name: Table output
Connection: CN_DW
Target schema: dbo
Target table: DIM_DATE_SEMESTER
Commit size: 1000
Truncate table: 
Ignore insert errors: 
Specify database fields: 
```

El proceso de la transformación completa es el siguiente:



Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:

```
SQLQuery7.sql - U...NT_gmanresas (65)  X
***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [pk_date_semester]
    ,[Year]
    ,[Semester]
FROM [SOURCE_gmanresas].[dbo].[DIM_DATE_SEMESTER]
```

	pk_date_semester	Year	Semester
1	1	2007	1
2	10	2011	2
3	11	2012	1
4	12	2012	2
5	13	2013	1
6	14	2013	2
7	15	2014	1

2.3.4. Transformación TR_DIM_COICOP

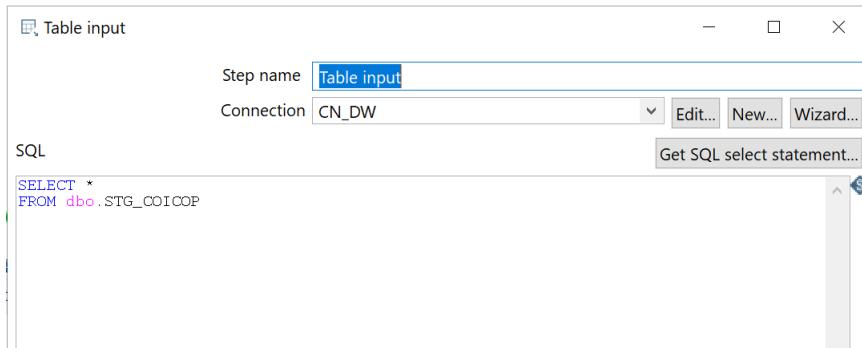
La transformación de « DIM_COICOP » contiene las siguientes etapas:

- “Table input”,
- “Sort rows”,
- “Add sequence”,
- “Table output”

A continuación, se muestran las capturas de pantalla correspondientes a estos pasos:

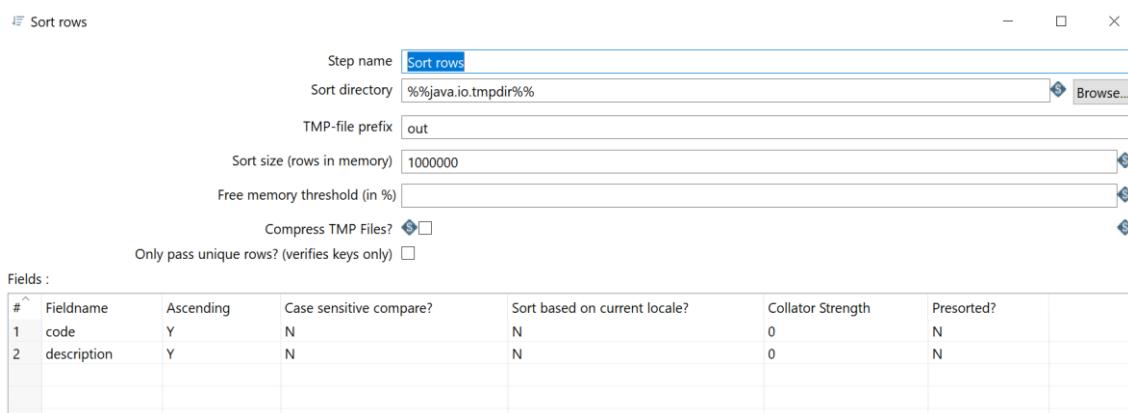
“Table input”

Mediante una sentencia «SELECT» de SQL obtenemos la información de la tabla correspondiente utilizando la conexión creada anteriormente.



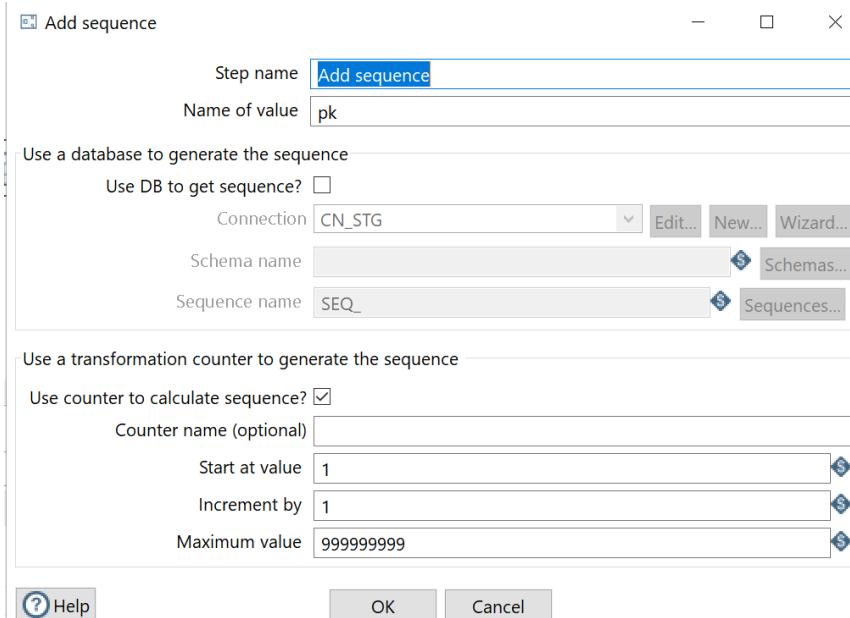
“Sort rows”

Con este componente ordenamos las filas de manera ascendente teniendo en cuenta en primer lugar la columna “code” y en segundo la columna “description”.



“Add sequence”

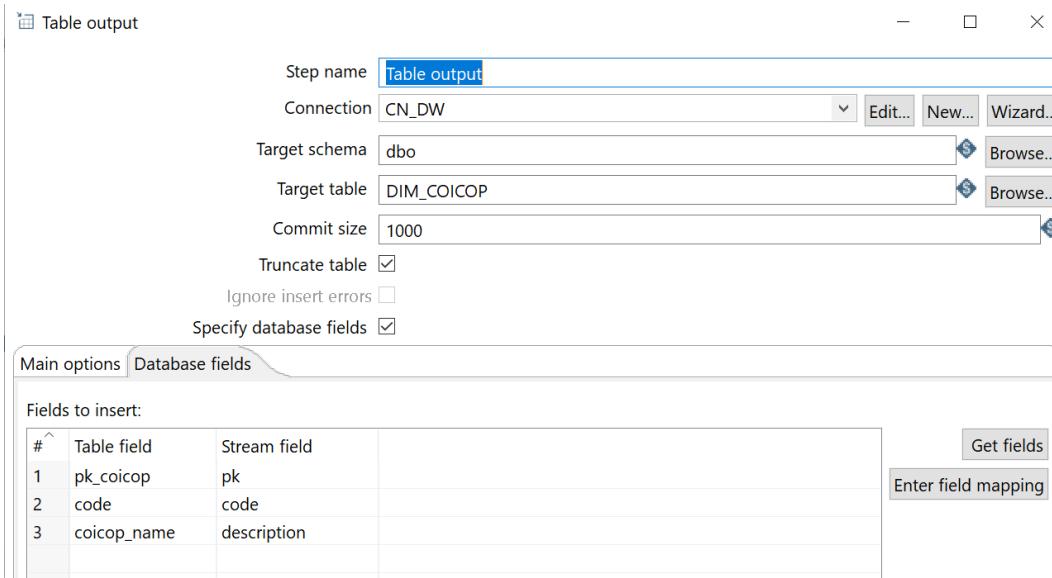
Con el componente “Add sequence” creamos una secuencia que hará las funciones de clave primaria de incremento automático.



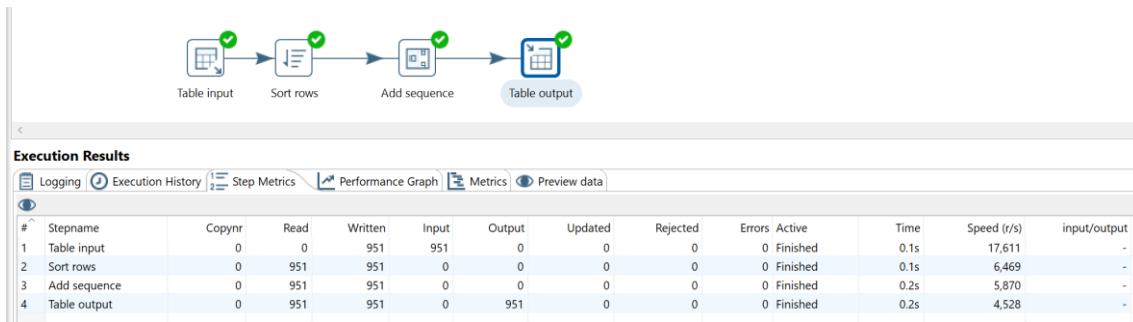
“Table output”

Finalmente, cargamos los datos en la tabla de dimensión.

Para ello debemos utilizar la conexión creada, CN_DW, con la variable de entorno. También se debe hacer el mapeo de los campos e indicar “Truncate table” para no duplicar los datos:



El proceso de la transformación completa es el siguiente:



Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:

```

SQLQuery10.sql -> NT_gmanresa (65) [x]
***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [pk_coicop]
,[code]
,[coicop_name]
FROM [SOURCE_gmanresa].[dbo].[DIM_COICOP]

```

The results table contains the following data:

pk_coicop	code	coicop_name
1	00FLASH	Flash estimate - All items HICP
2	00XALCOTOB	Overall index excluding alcohol and tobacco
3	00XAP	All-items excluding administered prices
4	00XAPFULL	All-items excluding fully administered prices
5	00XAPMAIN	All-items excluding mainly administered prices
6	00XE	Overall index excluding energy
7	00XEDUHEASOC	Overall index excluding Education, health and soci...
8	00XEFOOD	Overall index excluding energy, food, alcohol and t...
9	00XEFOODUNP	Overall index excluding energy and unprocessed f...
10	00XESCAIS	Overall index excluding energy and processed food

2.3.5. Transformación TR_DIM_TAX

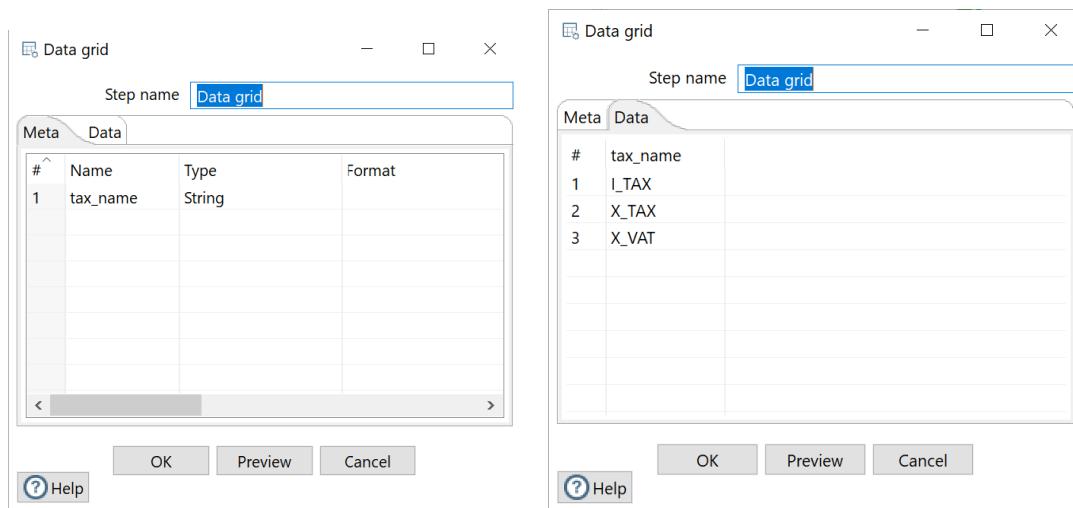
La transformación de « DIM_TAX » contiene las siguientes etapas:

- “Data grid”,
- “Add sequence”,
- “Table output”

A continuación, se muestran las capturas de pantalla correspondientes a estos pasos:

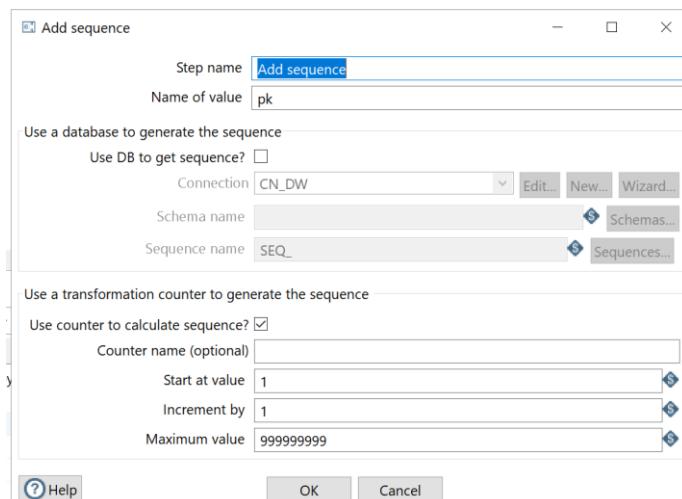
“Data grid”

Puesto que los valores a insertar son muy reducidos se decide hacer manualmente con el componente “Data grid”. La configuración de dicho componente es la siguiente:



“Add sequence”

Con el componente “Add sequence” creamos una secuencia que hará las funciones de clave primaria de incremento automático.



“Table output”

Finalmente, cargamos los datos en la tabla de dimensión.

Para ello debemos utilizar la conexión creada, CN_DW, con la variable de entorno. También se debe hacer el mapeo de los campos e indicar “Truncate table” para no duplicar los datos:

Table output

Step name: Table output

Connection: CN_DW

Target schema: dbo

Target table: DIM_TAX

Commit size: 1000

Truncate table:

Ignore insert errors:

Specify database fields:

Main options Database fields

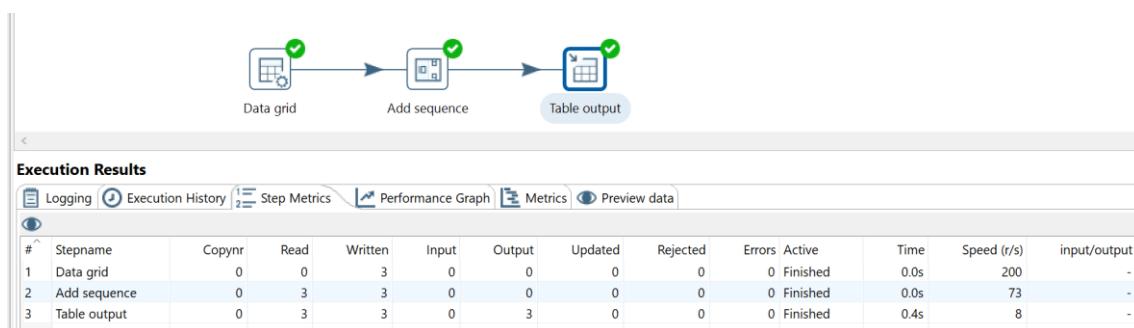
Fields to insert:

#	Table field	Stream field
1	pk_tax	pk
2	tax_name	tax_name

Get fields

Enter field mapping

El proceso de la transformación completa es el siguiente:



Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:

SQLQuery1.sql - U...NT_gmanresas (73) ➔

```
***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [pk_tax]
 ,[tax_name]
 FROM [SOURCE_gmanresas].[dbo].[DIM_TAX]
```

Results

pk_tax	tax_name
1	I_TAX
2	X_TAX
3	X_VAT

2.3.6. Transformación TR_DIM_PRODUCT

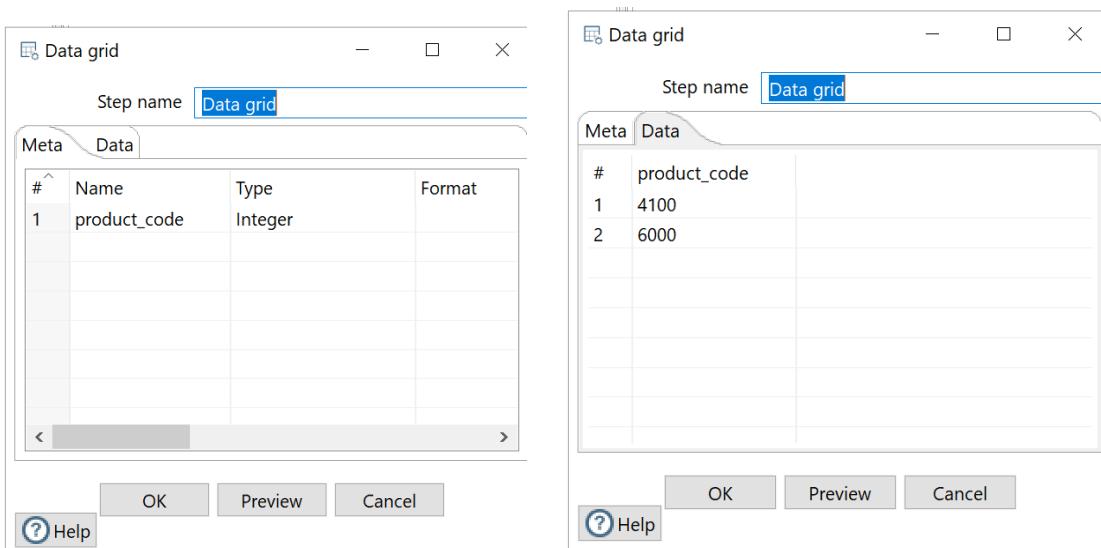
La transformación de « DIM_PRODUCT » contiene las siguientes etapas:

- “Data grid”,
- “Add sequence”,
- “Table output”

A continuación, se muestran las capturas de pantalla correspondientes a estos pasos:

“Data grid”

Puesto que los valores a insertar son muy reducidos se decide hacer manualmente con el componente “Data grid”. La configuración de dicho componente es la siguiente:

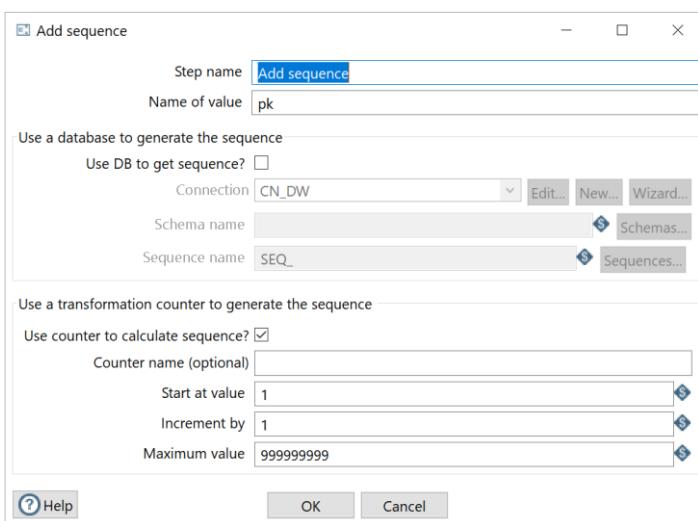


#	Name	Type	Format
1	product_code	Integer	

#	product_code
1	4100
2	6000

“Add sequence”

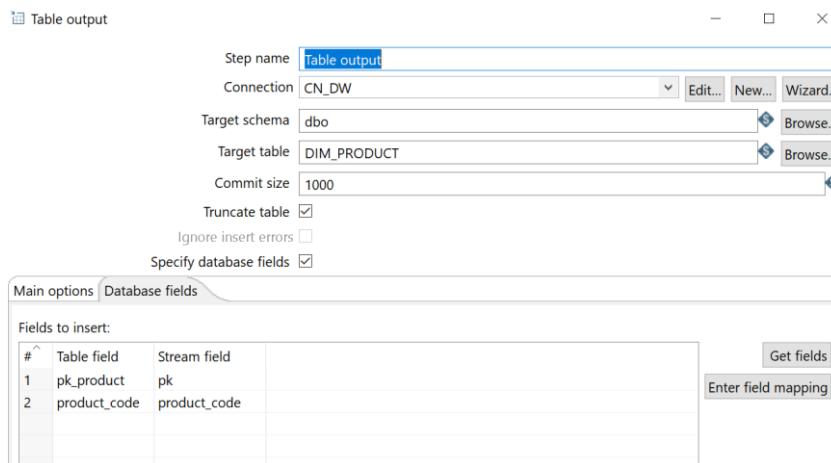
Con el componente “Add sequence” creamos una secuencia que hará las funciones de clave primaria de incremento automático.



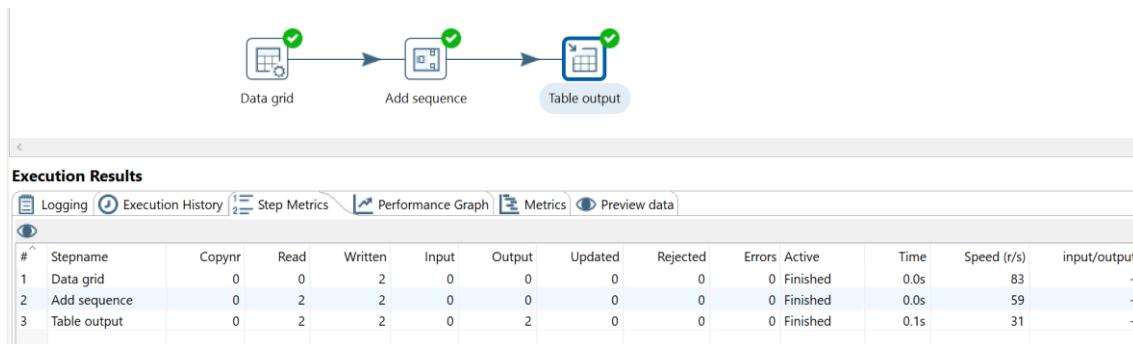
“Table output”

Finalmente, cargamos los datos en la tabla de dimensión.

Para ello debemos utilizar la conexión creada, CN_DW, con la variable de entorno. También se debe hacer el mapeo de los campos e indicar “Truncate table” para no duplicar los datos:



El proceso de la transformación completa es el siguiente:



Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:

```
SQLQuery2.sql - U...NT_gmanresa (54)  ↗ X
***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [pk_product]
,[product_code]
FROM [SOURCE_gmanresa].[dbo].[DIM_PRODUCT]
```

pk_product	product_code
1	4100
2	6000

2.3.7. Transformación TR_DIM_CONSUMPTION

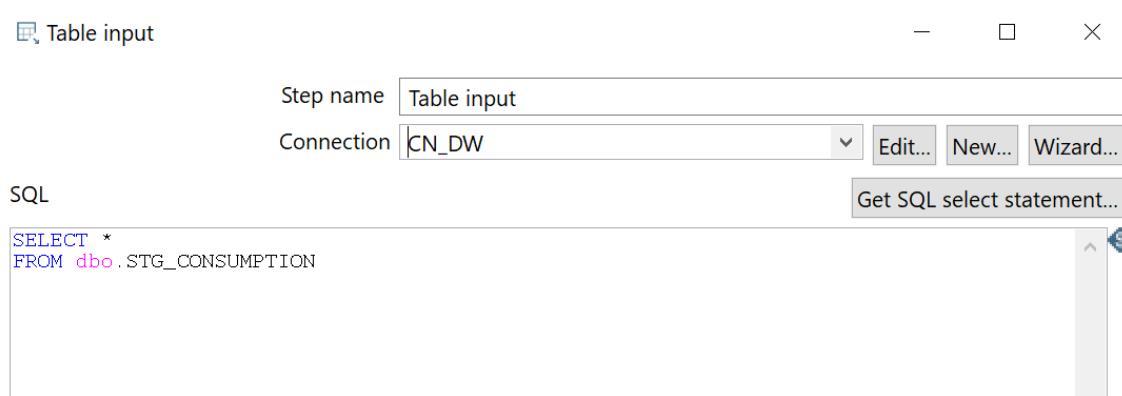
La transformación de « DIM_COICOP » contiene las siguientes etapas:

- “Table input”,
- “Add sequence”,
- “Table output”

A continuación, se muestran las capturas de pantalla correspondientes a estos pasos:

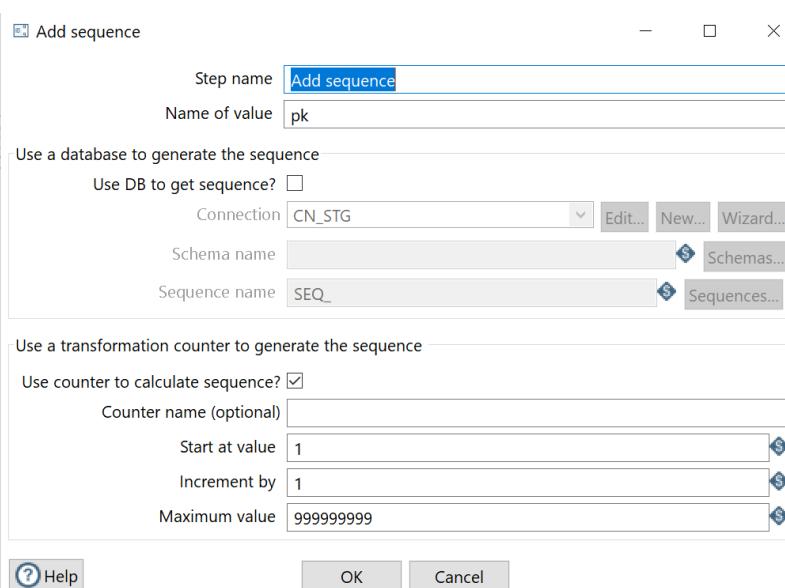
“Table input”

Mediante una sentencia «SELECT» de SQL obtenemos la información de la tabla correspondiente utilizando la conexión creada anteriormente.



“Add sequence”

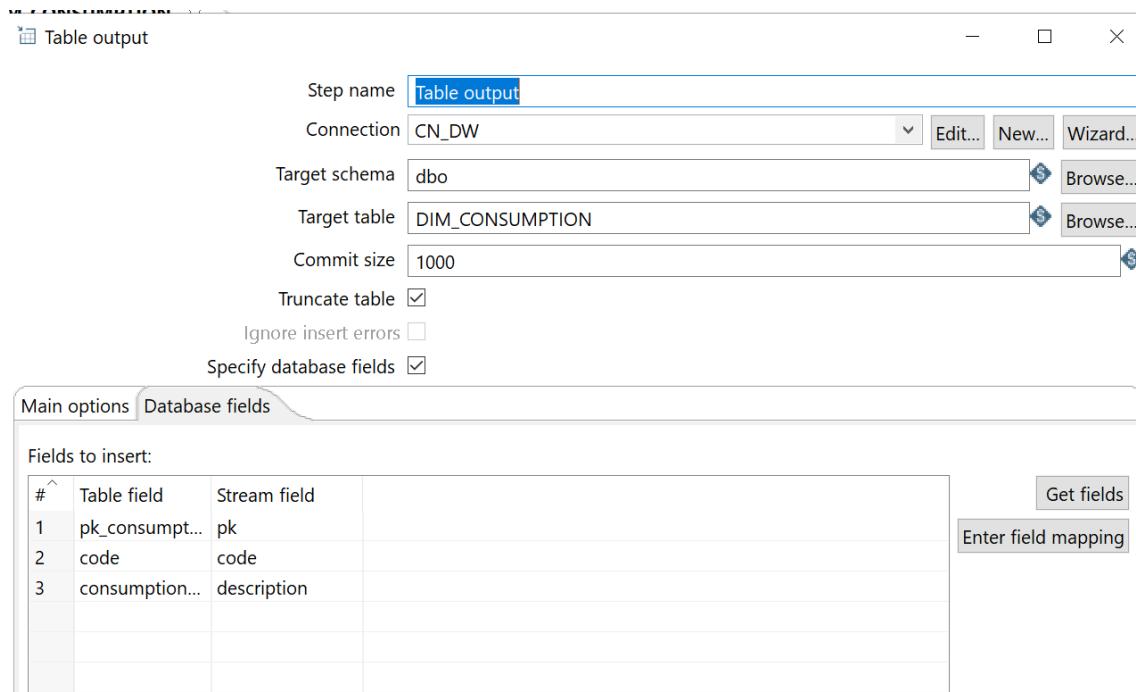
Con el componente “Add sequence” creamos una secuencia que hará las funciones de clave primaria de incremento automático.



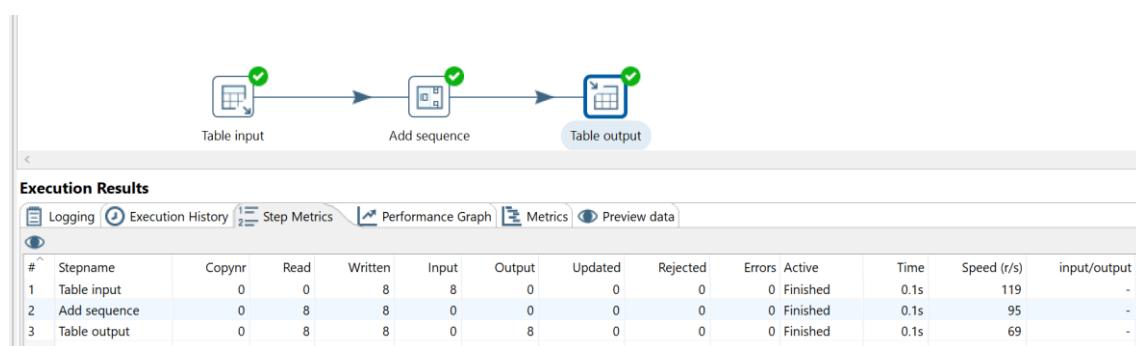
“Table output”

Finalmente, cargamos los datos en la tabla de dimensión.

Para ello debemos utilizar la conexión creada, CN_DW, con la variable de entorno. También se debe hacer el mapeo de los campos e indicar “Truncate table” para no duplicar los datos:



El proceso de la transformación completa es el siguiente:



Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:

SQLQuery3.sql - U...NT_gmanresas (54) X |

```
***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [pk_consumption]
    ,[code]
    ,[consumption_name]
FROM [SOURCE_gmanresas].[dbo].[DIM_CONSUMPTION]
```

146 %

	Results	Messages	
1	pk_consumption	code	consumption_name
1	1	4161901	Band DA : Consumption < 1 000 kWh
2	2	4161902	Band DB : 1 000 kWh < Consumption < 2 500 kWh
3	3	4161903	Band DC : 2 500 kWh < Consumption < 5 000 kWh
4	4	4161904	Band DD : 5 000 kWh < Consumption < 15 000 kWh
5	5	4161905	Band DE : Consumption > 15 000 kWh
6	6	4141901	Band D1 : Consumption < 20 GJ
7	7	4141902	Band D2 : 20 GJ < Consumption < 200 GJ
8	8	4141903	Band D3 : Consumption > 200 GJ

2.3.8. Transformación TR_DIM_CURRENCY

La transformación de « DIM_CURRENCY » contiene las siguientes etapas:

- “Data grid”,
- “Add sequence”,
- “Table output”

A continuación, se muestran las capturas de pantalla correspondientes a estos pasos:

“Data grid”

Puesto que los valores a insertar son muy reducidos se decide hacer manualmente con el componente “Data grid”. La configuración de dicho componente es la siguiente:

Left Screenshot (Meta Tab):

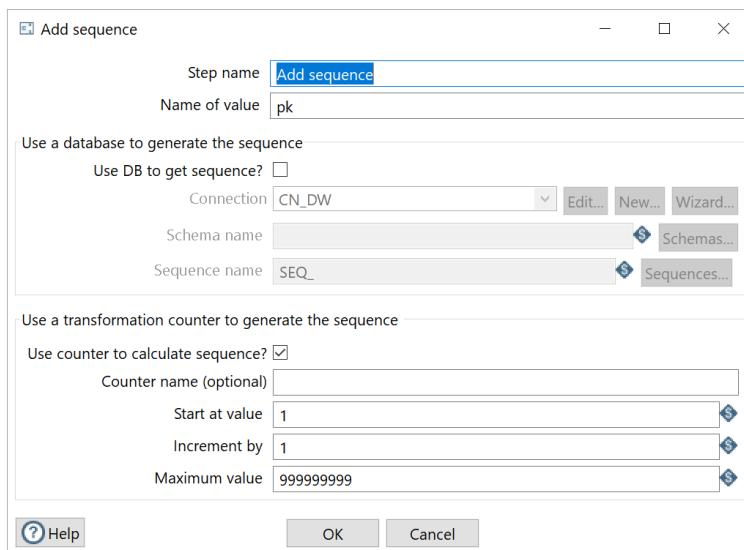
#	Name	Type	Format
1	currency_name	String	

Right Screenshot (Data Tab):

#	currency_name
1	EUR
2	NAC
3	PPS

“Add sequence”

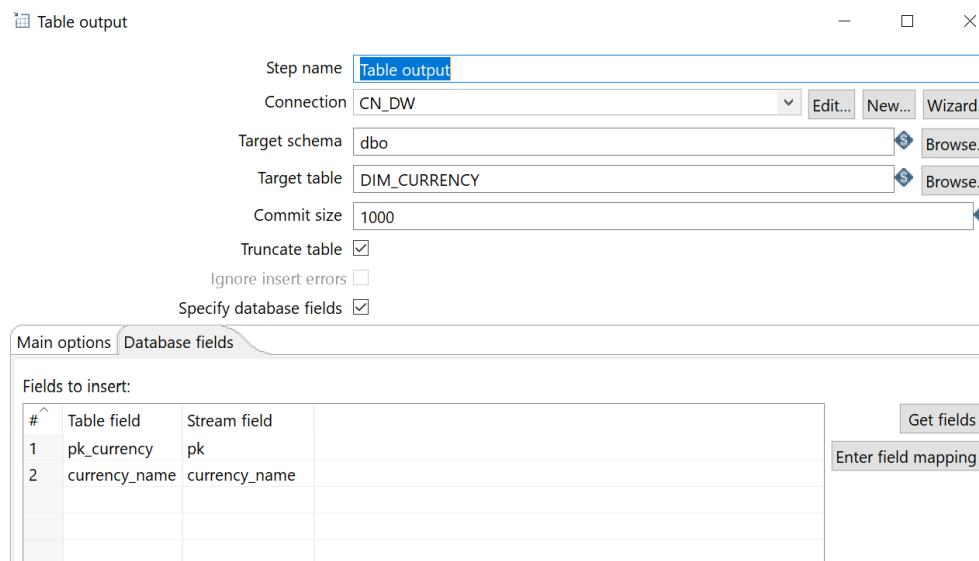
Con el componente “Add sequence” creamos una secuencia que hará las funciones de clave primaria de incremento automático.



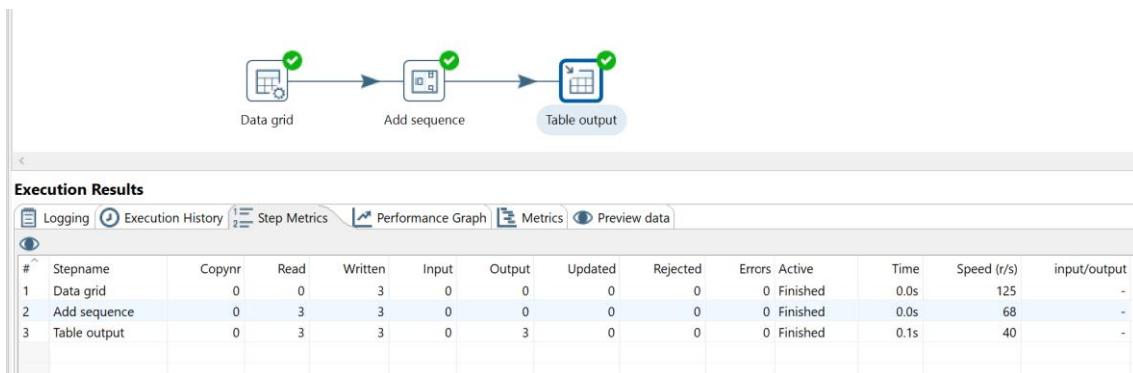
“Table output”

Finalmente, cargamos los datos en la tabla de dimensión.

Para ello debemos utilizar la conexión creada, CN_DW, con la variable de entorno. También se debe hacer el mapeo de los campos e indicar “Truncate table” para no duplicar los datos:



El proceso de la transformación completa es el siguiente:



Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:

```

SQLQuery4.sql - U...NT_gmanresas (54)  p X
***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [pk_currency]
      ,[currency_name]
   FROM [SOURCE_gmanresas].[dbo].[DIM_CURRENCY]
  
```

Results

pk_currency	currency_name
1	EUR
2	NAC
3	PPS

2.3.9. Transformación TR_DIM_UNIT

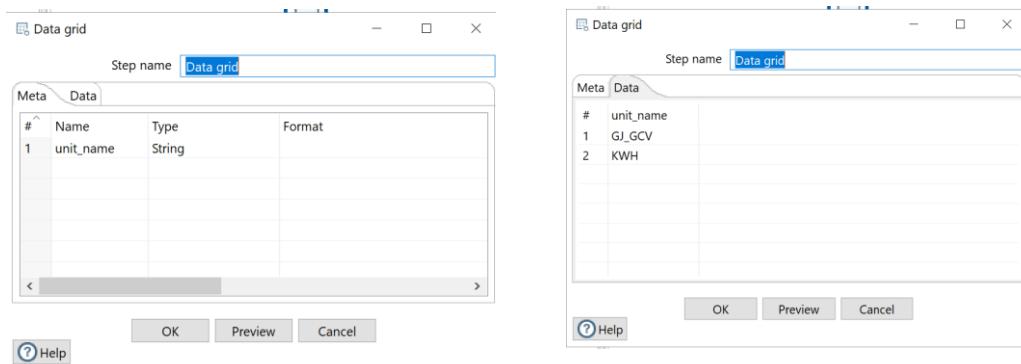
La transformación de « DIM_UNIT » contiene las siguientes etapas:

- “Data grid”,
- “Add sequence”,
- “Table output”

A continuación, se muestran las capturas de pantalla correspondientes a estos pasos:

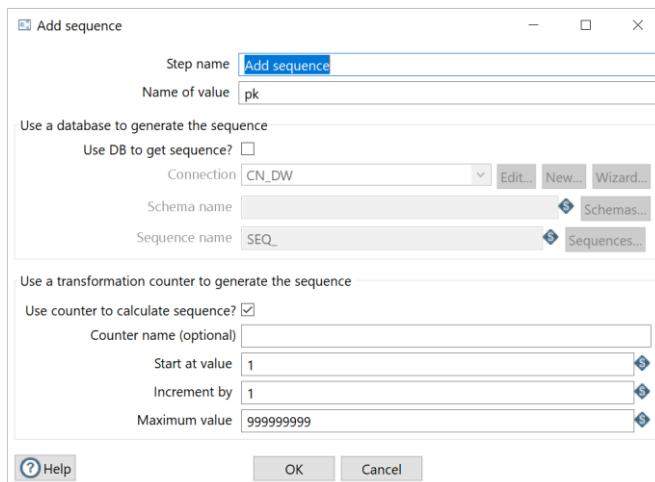
“Data grid”

Puesto que los valores a insertar son muy reducidos se decide hacer manualmente con el componente “Data grid”. La configuración de dicho componente es la siguiente:



“Add sequence”

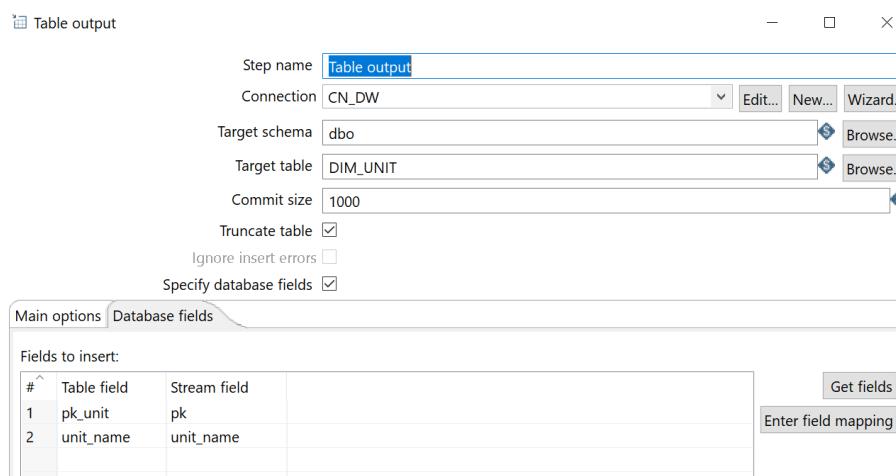
Con el componente “Add sequence” creamos una secuencia que hará las funciones de clave primaria de incremento automático.



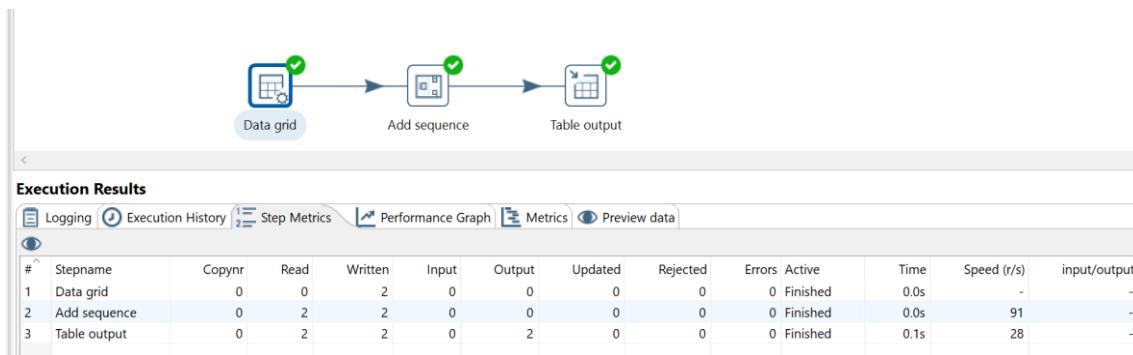
“Table output”

Finalmente, cargamos los datos en la tabla de dimensión.

Para ello debemos utilizar la conexión creada, CN_DW, con la variable de entorno. También se debe hacer el mapeo de los campos e indicar “Truncate table” para no duplicar los datos:



El proceso de la transformación completa es el siguiente:



Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:

The screenshot shows a SQL Server Management Studio window with a query results grid. The query is:

```
***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [pk_unit]
      ,[unit_name]
  FROM [SOURCE_gmanresas].[dbo].[DIM_UNIT]
```

The results grid shows two rows of data:

pk_unit	unit_name
1	GJ_GCV
2	KWH

2.4. Transformación Bloque TR_FACT

El bloque TR_FACT 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, a los hechos del modelo multidimensional del almacén.

2.4.1. Transformación FACT_EUROZONE_INDICATORS

Para cargar la información correspondiente en las tablas de hechos hay que buscar los valores de las claves foráneas en las tablas de dimensiones cargadas anteriormente.

Para ello es necesario unir la información de diferentes tablas. En este caso la transformación se ha realizado siguiendo los siguientes pasos:

- “Table input”,
- “Sort rows”,
- “Add sequence”,
- “Table output”

“Table input”

Con la siguiente sentencia SQL en el paso “Table input” accedemos a la información de las tablas de dimensiones correspondientes, así como de la tabla STG_HICP.

```

SELECT pk_date, pk_country.pk_coicop.HICP
FROM dbo.STG_HICP
JOIN dbo.DIM_DATE ON dbo.STG_HICP.period = CONCAT(dbo.DIM_DATE.Year, RIGHT('0' + CAST(dbo.DIM_DATE.Mo AS VARCHAR(2)), 2))
JOIN dbo.DIM_COUNTRY ON dbo.STG_HICP.geo = dbo.DIM_COUNTRY.code
JOIN dbo.DIM_COICOP ON dbo.STG_HICP.coicop = dbo.DIM_COICOP.code|

```

“Sort rows”

A continuación, se ordenan las filas antes de crear la secuencia.

#	Fieldname	Ascending	Case sensitive compare?	Sort based on current locale?	Collator Strength	Presorted?
1	pk_date	Y	N	N	0	N
2	pk_country	Y	N	N	0	N
3	pk_coicop	Y	N	N	0	N
4	HICP	Y	N	N	0	N

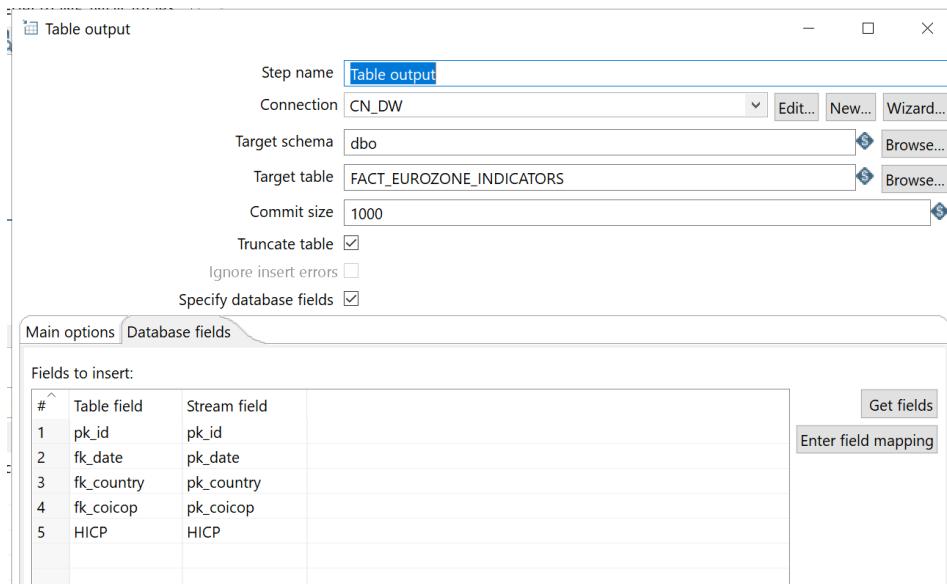
“Add sequence”

Con el componente “Add sequence” creamos una secuencia que hará las funciones de clave primaria de incremento automático.

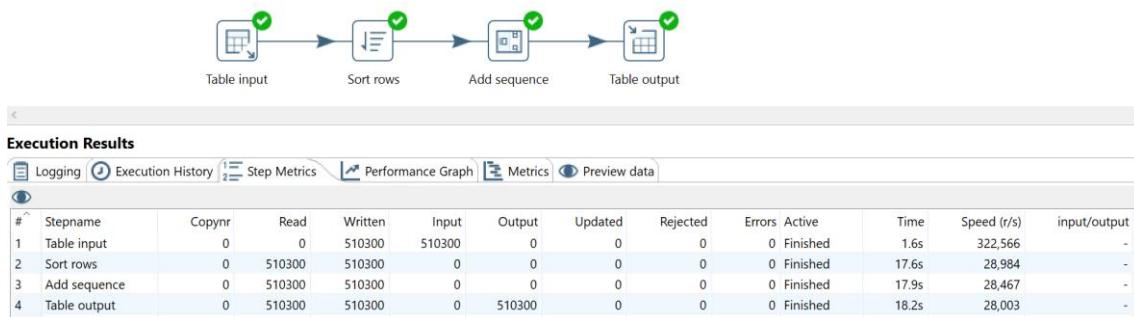
“Table output”

Finalmente, cargamos los datos en la tabla de hechos.

Para ello debemos utilizar la conexión creada, CN_DW, con la variable de entorno. También se debe hacer el mapeo de los campos e indicar “Truncate table” para no duplicar los datos:



El proceso de la transformación completa es el siguiente:



Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:

pk_id	fk_date	fk_country	fk_coicop	HICP
1	1	2	25	1.60
2	2	1	26	3.20
3	3	1	27	3.30
4	4	1	28	0.30
5	5	1	29	0.30
6	6	1	31	0.20
7	7	1	34	0.20
8	8	1	36	0.50
9	9	1	37	1.80
10	10	1	39	0.10
11	11	1	41	-0.30
12	12	1	42	0.00
13	13	1	43	0.50
14	14	1	44	0.20

2.4.2. Transformación FACT_ENERY_PRICE

Para cargar la información correspondiente en las tablas de hechos hay que buscar los valores de las claves foráneas en las tablas de dimensiones cargadas anteriormente.

Para ello es necesario unir la información de diferentes tablas. En este caso la transformación se ha realizado siguiendo los siguientes pasos:

- “Table input”,
- “Database lookup” (x6)
- “Sort rows”,
- “Add sequence”,
- “Table output”

“Table input”

Con la siguiente sentencia SQL en el paso “Table input” realizamos un UNION con las tablas STG_PELEC y STG_PGAS y a continuación con el resultado realizamos un JOIN con la tabla DIM_DATE_SEMESTER para obtener pk_date correspondiente.

```

Step name: Table input
Connection: CN_STG
SQL:
SELECT *
FROM
(
    SELECT freq, product, consom, unit, tax, currency, geo, period, pelec AS price
    FROM dbo.STG_PELEC
    UNION
    SELECT freq, product, consom, unit, tax, currency, geo, period, pgas AS price
    FROM dbo.STG_PGAS
) AS merged_data
JOIN DIM_DATE_SEMESTER ON DIM_DATE_SEMESTER.Year = CAST(SUBSTRING(merged_data.period, 1, 4) AS INT)
AND DIM_DATE_SEMESTER.Semester = CAST(SUBSTRING(merged_data.period, 7, 1) AS INT);
  
```

“Database lookup”

Utilizamos el componente “Database lookup” para buscar las claves foráneas de cada una de las dimensiones de nuestro diseño. En este caso se ha repetido el paso 6 veces.

Database lookup

Step name: Database lookup

Connection: CN_DW

Lookup schema: dbo

Lookup table: DIM_COUNTRY

Enable cache?

Cache size in rows (0=cache everything): 0

Load all data from table

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

#	Table field	Comparator	Field1	Field2
1	code	=	geo	

Values to return from the lookup table :

#	Field	New name	Default	Type
1	pk_country			None

Database lookup

Step name: Database lookup 2

Connection: CN_DW

Lookup schema: dbo

Lookup table: DIM_CURRENCY

Enable cache?

Cache size in rows (0=cache everything): 0

Load all data from table

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

#	Table field	Comparator	Field1	Field2
1	currency_name	=	currency	

Values to return from the lookup table :

#	Field	New name	Default	Type
1	pk_currency			None

Database lookup

Step name: Database lookup 3

Connection: CN_DW

Lookup schema: dbo

Lookup table: DIM_TAX

Enable cache?

Cache size in rows (0=cache everything): 0

Load all data from table

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

#	Table field	Comparator	Field1	Field2
1	tax_name	=	tax	

Values to return from the lookup table :

#	Field	New name	Default	Type
1	pk_tax			None

Database lookup

Step name: Database lookup 4

Connection: CN_DW

Lookup schema: dbo

Lookup table: DIM_CONSUMPTION

Enable cache?

Cache size in rows (0=cache everything): 0

Load all data from table

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

#	Table field	Comparator	Field1	Field2
1	code	=	consum	

Values to return from the lookup table :

#	Field	New name	Default	Type
1	pk_consumption			None

The image shows two windows of the Informatica Data Integration tool. Both windows are titled "Database lookup".

Left Window (Step name: Database lookup 5):

- Step name: Database lookup 5
- Connection: CN_DW
- Lookup schema: dbo
- Lookup table: DIM_UNIT
- Enable cache?
- Cache size in rows (0=cache everything): 0
- Load all data from table
- The key(s) to look up the value(s):

#	Table field	Comparator	Field1	Field2
1	unit_name	=	unit	

- Values to return from the lookup table :

#	Field	New name	Default	Type
1	pk_unit			None

Right Window (Step name: Database lookup 6):

- Step name: Database lookup 6
- Connection: CN_DW
- Lookup schema: dbo
- Lookup table: DIM_PRODUCT
- Enable cache?
- Cache size in rows (0=cache everything): 0
- Load all data from table
- The key(s) to look up the value(s):

#	Table field	Comparator	Field1	Field2
1	product_code	=	product	

- Values to return from the lookup table :

#	Field	New name	Default	Type
1	pk_product			None

“Sort rows”

A continuación, se ordenan las filas antes de crear la secuencia.

The image shows the "Sort rows" configuration window.

Step name: Sort rows

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

TMP-file prefix: out

Sort size (rows in memory): 1000000

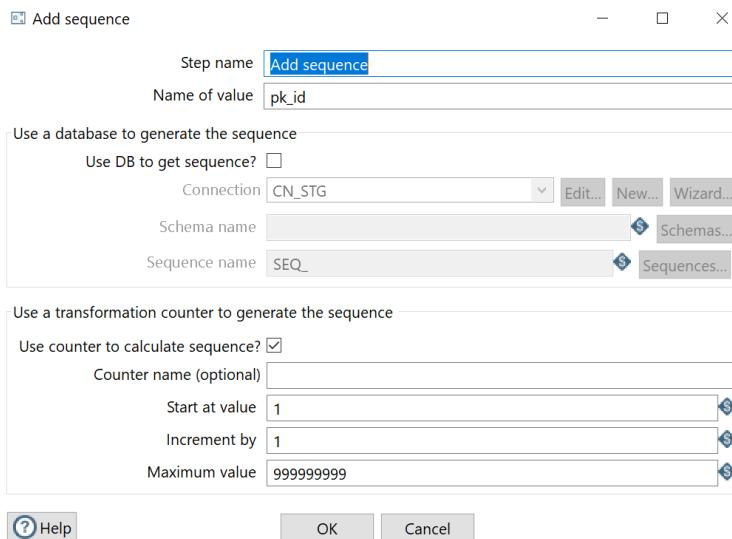
Free memory threshold (in %):

Fields :

#	Fieldname	Ascending	Case sensitive compare?	Sort based on current locale?	Collator Strength	Presorted?
1	pk_date_semester	Y	N	N	0	N
2	pk_country	Y	N	N	0	N
3	pk_currency	Y	N	N	0	N
4	pk_tax	Y	N	N	0	N
5	pk_consumption	Y	N	N	0	N
6	pk_unit	Y	N	N	0	N
7	pk_product	Y	N	N	0	N
8	price	Y	N	N	0	N

“Add sequence”

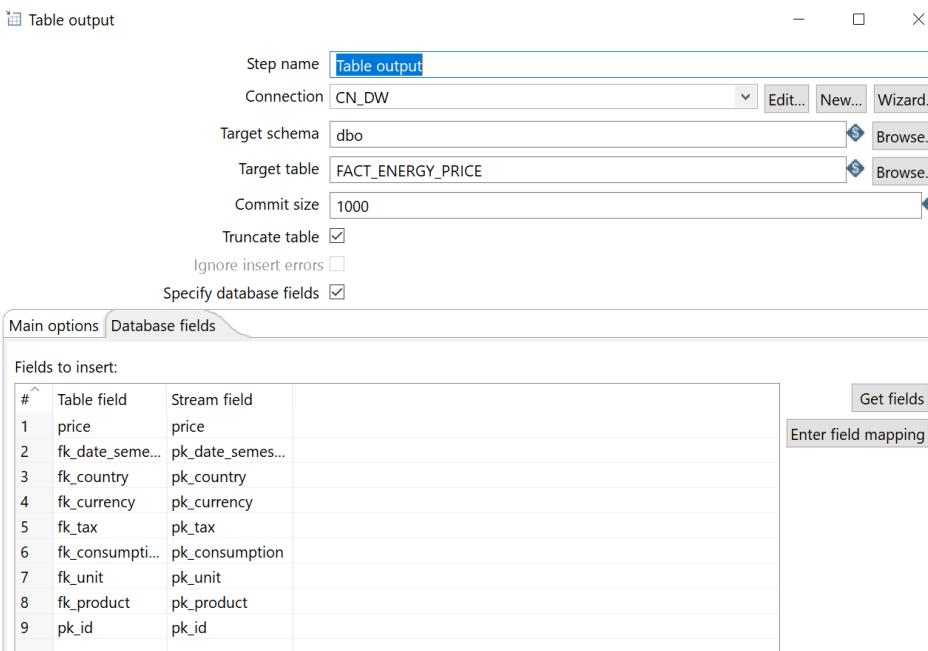
Con el componente “Add sequence” creamos una secuencia que hará las funciones de clave primaria de incremento automático.



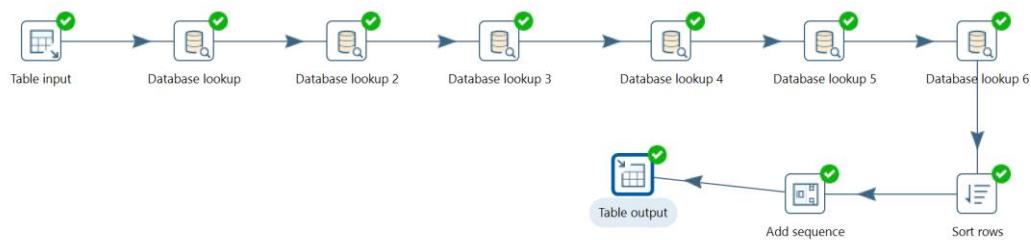
“Table output”

Finalmente, cargamos los datos en la tabla de hechos.

Para ello debemos utilizar la conexión creada, CN_DW, con la variable de entorno. También se debe hacer el mapeo de los campos e indicar “Truncate table” para no duplicar los datos:



El proceso de la transformación completa es el siguiente:



Execution Results													
#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)	input/output
1	Table input	0	0	112995	112995	0	0	0	0	Finished	47.2s	2,395	-
2	Database lookup	0	112995	112995	112995	0	0	0	0	Finished	51.7s	2,184	-
3	Database lookup 2	0	112995	112995	112995	0	0	0	0	Finished	51.8s	2,180	-
4	Database lookup 3	0	112995	112995	112995	0	0	0	0	Finished	51.9s	2,178	-
5	Database lookup 4	0	112995	112995	112995	0	0	0	0	Finished	51.9s	2,176	-
6	Database lookup 5	0	112995	112995	112995	0	0	0	0	Finished	52.0s	2,175	-
7	Database lookup 6	0	112995	112995	112995	0	0	0	0	Finished	52.0s	2,172	-
8	Sort rows	0	112995	112995	0	0	0	0	0	Finished	56.9s	1,984	-
9	Add sequence	0	112995	112995	0	0	0	0	0	Finished	57.5s	1,966	-
10	Table output	0	112995	112995	0	112995	0	0	0	Finished	57.9s	1,952	-

Y podemos comprobar en la base de datos desde el SQL Server Management Studio el resultado del proceso realizado:

```
SQLQuery1.sql - U...NT_gmanresas (57)* -> X
SELECT *
FROM [SOURCE_gmanresas].[dbo].[FACT_ENERGY_PRICE]
```

	pk_id	fk_date_semester	fk_country	fk_currency	fk_tax	fk_consumption	fk_unit	fk_product	price
47532	47532	21	21	1	1	3	2	2	0.2857
47533	47533	21	21	1	1	4	2	2	0.2588
47534	47534	21	21	1	1	5	2	2	0.2339
47535	47535	21	21	1	1	6	1	1	20.2584
47536	47536	21	21	1	1	6	2	1	0.0729
47537	47537	21	21	1	1	7	1	1	14.5114
47538	47538	21	21	1	1	7	2	1	0.0522
47539	47539	21	21	1	1	8	1	1	13.0783
47540	47540	21	21	1	1	8	2	1	0.0471
47541	47541	21	21	1	2	1	2	2	0.2916
47542	47542	21	21	1	2	2	2	2	0.2084
47543	47543	21	1	2	3		2	2	0.1929

3. IMPLEMENTACIÓN TRABAJOS (JOBS) DE LOS PROCESOS DE ETL

Una vez realizados los procesos siguientes:

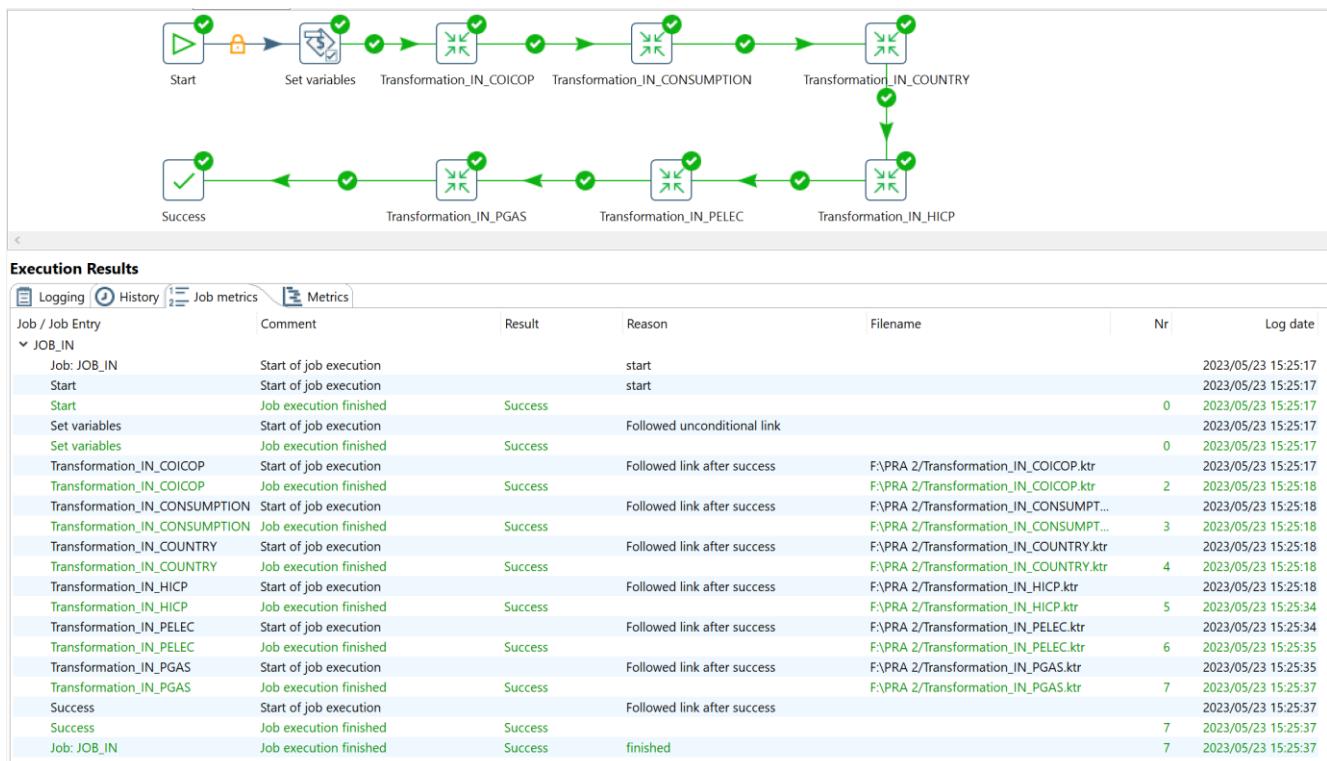
- Bloque IN_: procesos ETL de transformación y carga al área intermedia.
- Bloque TR_DIM: procesos ETL de transformación y carga de dimensiones.
- Bloque TR_FACT: procesos ETL de transformación y carga de hechos.

Falta realizar el proceso completo diseñando los trabajos (Jobs) que permitan la ejecución secuencial de todos los procesos ETL mencionados anteriormente.

3.1. Jobs Bloque IN

El trabajo “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 es el siguiente:



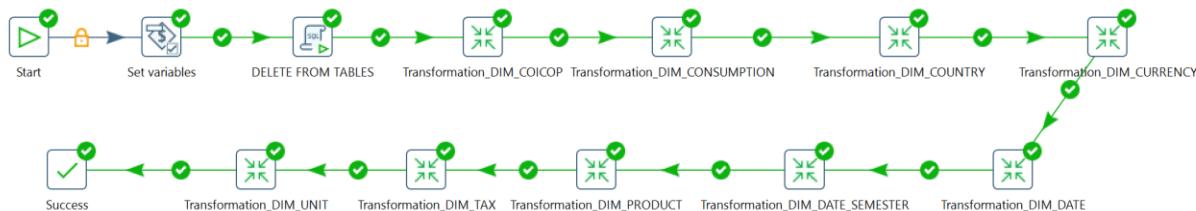
Los pasos incluidos en el trabajo «JOB_IN» son:

- Inicio del job.
- Configuración de las variables de entorno.
- Ejecución de las transformaciones “IN”.
- Finalización del job.

3.2. Jobs Bloque TR_DIM

El trabajo “JOB_TR_DIMS” procesa todas las transformaciones del bloque “TR_DIMS” para la carga de datos, desde las tablas intermedias hasta las tablas de dimensiones del almacén.

El diseño completo del trabajo es el siguiente:



Execution Results

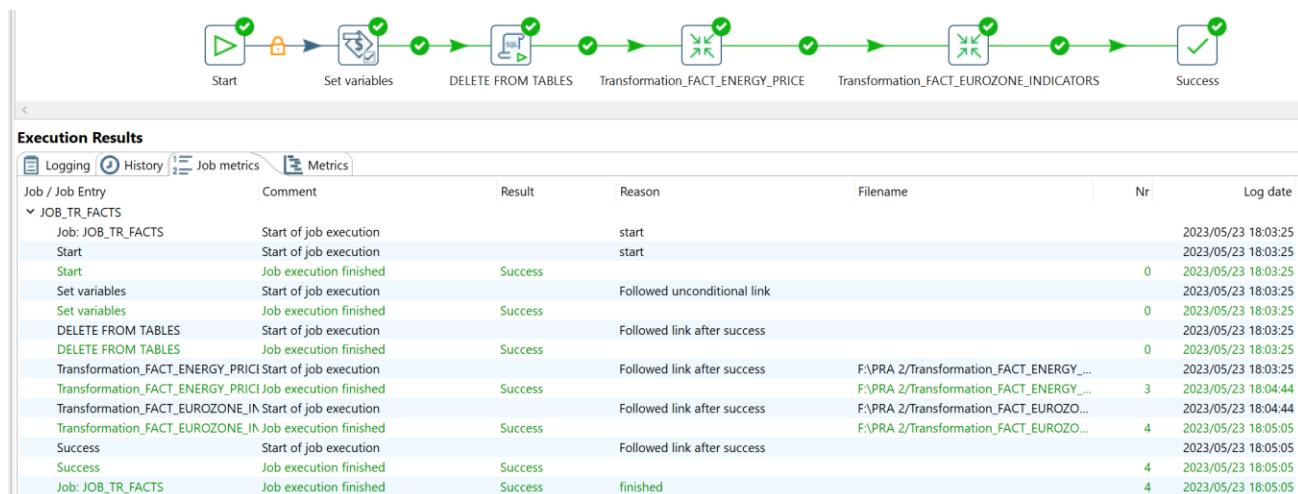
Job / Job Entry	Comment	Result	Reason	Filename	Nr	Log date
Job: JOB_TR_DIMS	Start of job execution		start			2023/05/23 15:54:00
Start	Start of job execution		start			2023/05/23 15:54:00
Start	Job execution finished	Success			0	2023/05/23 15:54:00
Set variables	Start of job execution		Followed unconditional link			2023/05/23 15:54:00
Set variables	Job execution finished	Success			0	2023/05/23 15:54:00
DELETE FROM TABLES	Start of job execution		Followed link after success			2023/05/23 15:54:00
DELETE FROM TABLES	Job execution finished	Success			0	2023/05/23 15:54:00
Transformation_DIM_COICOP	Start of job execution		Followed link after success	F:\PRA 2\Transformation_TR_DIM_COIC...		2023/05/23 15:54:00
Transformation_DIM_COICOP	Job execution finished	Success			3	2023/05/23 15:54:01
Transformation_DIM_CONSUMPTION	Start of job execution		Followed link after success	F:\PRA 2\Transformation_TR_DIM_CONS...		2023/05/23 15:54:01
Transformation_DIM_CONSUMPTION	Job execution finished	Success			4	2023/05/23 15:54:01
Transformation_DIM_COUNTRY	Start of job execution		Followed link after success	F:\PRA 2\Transformation_TR_DIM_COUN...		2023/05/23 15:54:01
Transformation_DIM_COUNTRY	Job execution finished	Success			5	2023/05/23 15:54:01
Transformation_DIM_CURRENCY	Start of job execution		Followed link after success	F:\PRA 2\Transformation_TR_DIM_CURR...		2023/05/23 15:54:01
Transformation_DIM_CURRENCY	Job execution finished	Success			6	2023/05/23 15:54:01
Transformation_DIM_DATE	Start of job execution		Followed link after success	F:\PRA 2\Transformation_TR_DIM_DATE...		2023/05/23 15:54:01
Transformation_DIM_DATE	Job execution finished	Success			7	2023/05/23 15:54:02
Transformation_DIM_DATE_SEMESTER	Start of job execution		Followed link after success	F:\PRA 2\Transformation_TR_DIM_DATE...		2023/05/23 15:54:02
Transformation_DIM_DATE_SEMESTER	Job execution finished	Success			8	2023/05/23 15:54:02
Transformation_DIM_PRODUCT	Start of job execution		Followed link after success	F:\PRA 2\Transformation_TR_DIM_PROD...		2023/05/23 15:54:02
Transformation_DIM_PRODUCT	Job execution finished	Success			9	2023/05/23 15:54:02
Transformation_DIM_TAX	Start of job execution		Followed link after success	F:\PRA 2\Transformation_TR_DIM_TAX.ktr		2023/05/23 15:54:02
Transformation_DIM_TAX	Job execution finished	Success			10	2023/05/23 15:54:02
Transformation_DIM_UNIT	Start of job execution		Followed link after success	F:\PRA 2\Transformation_TR_DIM_UNIT....		2023/05/23 15:54:02
Transformation_DIM_UNIT	Job execution finished	Success			11	2023/05/23 15:54:02
Success	Start of job execution		Followed link after success			2023/05/23 15:54:02
Success	Job execution finished	Success			11	2023/05/23 15:54:02
Job: JOB_TR_DIMS	Job execution finished	Success	finished		11	2023/05/23 15:54:02

Los pasos incluidos son:

- Inicio del job.
- Carga de variables de entorno.
- Borrado del contenido de las tablas de hechos ya que sino no se podrían ejecutar correctamente las transformaciones de las dimensiones. No se borra el contenido las tablas de dimensiones ya que en cada transformación se ha especificado “truncate table”.
- Ejecución secuencial de todas las transformaciones “TR_DIM”.
- Finalización del job.

3.3. Jobs Bloque TR_FACT

El trabajo “JOB_TR_FACTS” procesa todas las transformaciones del bloque “TR_FACT” para la carga de datos desde las tablas intermedias a las tablas de hechos del almacén.



Los pasos incluidos en el trabajo son:

- Inicio del job.
- Carga de variables de entorno.
- Eliminación del contenido de las tablas de hechos.
- Ejecución de las transformaciones «TR_FACT».
- Finalización del job.

3.4. Proceso completo (utiliza los jobs IN y TR)

El trabajo “JOB_CARGA_DW” une todos los “jobs” anteriores en un único proceso.

Los pasos incluidos este job son:

- Inicio del job.
- Carga de variables de entorno.
- Ejecución de los jobs de carga de todas las transformaciones (“JOB_IN”, “JOB_TR_DIMS”, “JOB_TR_FACTS”).
- Finalización del job.

La transformación completa y su ejecución se observa en la siguiente imagen:



Execution Results						
Job / Job Entry	Comment	Result	Reason	Filename	Nr	Log date
Job: JOB_CARGA_DW						
Start	Start of job execution		start			2023/05/23 18:06:36
Start	Start of job execution		start			2023/05/23 18:06:36
Job: JOB_IN	Job execution finished	Success			0	2023/05/23 18:06:36
Set variables	Start of job execution		Followed unconditional link			2023/05/23 18:06:36
Set variables	Job execution finished	Success			0	2023/05/23 18:06:36
Job: JOB_IN	Start of job execution		Followed link after success	F:\PRA 2\JOB_IN.kjb		2023/05/23 18:06:36
Job: JOB_TR_FACTS	Job execution finished	Success		F:\PRA 2\JOB_TR_FACTS.kjb	1	2023/05/23 18:06:55
Job: JOB_TR_DIMS	Start of job execution		Followed link after success	F:\PRA 2\JOB_TR_DIMS.kjb		2023/05/23 18:06:55
Job: JOB_TR_FACTS	Job execution finished	Success		F:\PRA 2\JOB_TR_FACTS.kjb	2	2023/05/23 18:06:59
Job: JOB_TR_DIMS	Start of job execution		Followed link after success	F:\PRA 2\JOB_TR_DIMS.kjb		2023/05/23 18:06:59
Job: JOB_CARGA_DW	Job execution finished	Success	finished		4	2023/05/23 18:08:22
Success	Start of job execution		Followed link after success			2023/05/23 18:08:22
Success	Job execution finished	Success			4	2023/05/23 18:08:22

Se observa que se ejecuta el “job” entero correctamente. Observamos que el tiempo total de la carga inicial del data warehouse es de aproximadamente un minuto y medio.