

Caso práctico: almacén de datos para el análisis del impacto conductual de la COVID-19 sobre la población

Solución PRA2- Carga de datos

Índice de contenidos

1. Carga de datos	2
2. Identificación de los procesos ETL	2
2.1. Bloque IN (de las fuentes a tablas intermedias)	3
2.2. Bloque TR (poblar las tablas de nuestro almacén)	4
3. Diseño de los procesos ETL	5
3.1. Creación de tablas intermedias (staging area)	5
3.2. Creación del modelo multidimensional	7
3.2.1. Dimensiones	7
3.2.2. Tablas de hechos	9
3.3. Creación del proceso de extracción, transformación y carga	11
3.3.1. Variables de entorno	11
3.3.2. Conexión a la base de datos SQL Server	12
3.3.3. Bloque IN	13
3.3.4. Bloque TR	33
3.3.5. Bloque TR_DIM	34
3.3.6. Bloque TR_FACT	43
4. Implementación de trabajos con procesos ETL	51

1. Carga de datos

Esta actividad consiste en realizar el diseño de los procesos de extracción, transformación y carga de los datos de las fuentes de datos proporcionadas.

Los apartados que se han desarrollado en esta solución de la PRA2 son los siguientes:

- Identificación de los procesos de extracción, transformación y carga de datos (ETL) hacia el almacén de datos.
- Diseño y desarrollo de los procesos de ETL mediante las herramientas de diseño proporcionadas.
- Implementación con trabajos de los procesos de ETL.

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*).

En vuestro caso, y dentro del ámbito académico que nos ocupa, esta es una carga inicial, por lo que vuestros procesos no se diseñarán con el objetivo de repetirse de manera periódica. También es cierto que desconocéis la ventana de tiempo disponible (lo que no es una condición para esta actividad), pero en un contexto de producción, este es un factor muy relevante. En lo relativo al área intermedia hay que tener en cuenta que su uso, guardar la información de origen en bruto, puede facilitar mucho el trabajo de depuración de la información o de trazabilidad del dato, pudiendo ir desde el dato elaborado al dato en bruto.

En vuestro caso, identificaréis dos bloques y utilizaréis 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, de los procesos de transformación, para la carga de las tablas de hecho. 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 (stage)
IN_DENUNCIAS_INFRAACCIONES	Carga de los datos correspondientes a las estadísticas sobre los expedientes incorporados por el artículo 36.6 LOPSC de desobediencia durante el estado de emergencia sanitaria de la COVID-19 en la comunidad de Euskadi.	ACUMULADO-DENUNCIAS-INFRAACCIONES.xlsx	STG_Denuncias_Infracciones
IN_POBLACION	Carga de los datos correspondientes a las cifras de población por provincia a 1 de enero de 2020.	poblacion_9687bsc.csv	STG_Poblacion
IN_LLAMADAS_112	Carga de los datos correspondientes a los datos de llamadas operativas gestionadas por CAT112.	rows.xml	STG_Llamadas112
IN_MOVILIDAD	Carga de los datos correspondientes a la movilidad de la población durante el estado de alarma.	35167bsc.csv	STG_Movilidad
IN_EVITAR_AGLOMERACION	Carga de datos correspondientes al porcentaje de la población que evitaba las aglomeraciones con motivo del coronavirus.	tatistic_id1104235_covid-19_-poblacion-que-evitaba-las-aglomeraciones-segun-edad-en-espana-2020.xlsx	STG_Evitar_Aglomeracion

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

El bloque «TR_» de procesos de 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_FECHA	Carga y transformación de la dimensión temporal.	SQL	DIM_Fecha
TR_DIM_GRUPO_EDAD	Carga de la dimensión con información de los grupos de edad.	«entrada manual» (DataGrid)	DIM_Grupo_Edad
TR_DIM_AMBITO_GEOGRAFICO	Carga y transformación de la dimensión con datos de los ámbitos geográficos.	STG_Poblacion STG_Llamadas_CAT112 STG_Evitar_Aglomeracion	DIM_Ambito_Geografico
TR_DIM_TIPOLOGIA	Carga y transformación de la dimensión con datos de las tipologías de llamada.	STG_Llamadas_CAT112	DIM_Tipologia
TR_DIM_MEDICION	Carga de la dimensión con información de las unidades de medida.	«entrada manual» (DataGrid)	DIM_Medicion

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

Nombre ETL	Descripción	Tabla origen
TR_FACT_MEDICIONES	Carga y transformación de la tabla de hechos «FACT_Mediciones».	STG_Denuncias_Infracciones STG_Movilidad STG_Evitar_Aglomeracion STG_Poblacion
TR_FACT_LLAMADAS112	Carga y transformación de la tabla de hechos «FACT_Llamadas112».	STG_Llamadas112

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 en nuestro caso, aunque no es recomendable en cargas más complejas y cambiantes.

3. Diseño de los procesos 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.

IN_DENUNCIAS_INFRACCIONES

```
DROP TABLE [dbo].[STG_Denuncias_Infracciones]
GO

CREATE TABLE [dbo].[STG_Denuncias_Infracciones](
    [provincia] [varchar](100) NULL,
    [identificados_ertzaintza] [float] NULL,
    [detenidos_ertzaintza] [float] NULL,
    [denuncias_ertzaintza] [float] NULL,
    [vehic_intercept_ertzaintza] [float] NULL,
    [identificados_pp11] [float] NULL,
    [detenidos_pp11] [float] NULL,
    [denuncias_pp11] [float] NULL,
    [vehic_intercept_pp11] [float] NULL,
    [fecha_final] [datetime] NULL
) ON [PRIMARY]
GO
```

IN_POBLACION

```
DROP TABLE [dbo].[STG_Poblacion]
GO

CREATE TABLE [dbo].[STG_Poblacion](
    [provincia_codigo] [varchar](2) NULL,
    [provincia_nombre] [varchar](100) NULL,
```

```

        [poblacion] [bigint] NULL,
        [periodo] [varchar] (25) NULL
    ) ON [PRIMARY]
GO

```

IN_LLAMADAS_112

```

DROP TABLE [dbo].[STG_Llamadas112]
GO

```

```

CREATE TABLE [dbo].[STG_Llamadas112](
    [año] [int] NULL,
    [mes] [int] NULL,
    [provincia] [varchar] (100) NULL,
    [comarca] [varchar] (100) NULL,
    [municipio] [varchar] (100) NULL,
    [tipo] [varchar] (100) NULL,
    [llamadas] [int] NULL
) ON [PRIMARY]
GO

```

IN_EVITAR_AGLOMERACION

```

DROP TABLE [dbo].[STG_Evitar_Aglomeracion]
GO

```

```

CREATE TABLE [dbo].[STG_Evitar_Aglomeracion](
    [provincia] [varchar] (100) NULL,
    [comunidad_autonoma] [varchar] (100) NULL,
    [grupo_edad] [varchar] (7) NULL,
    [porc_poblacion] [float] NULL
) ON [PRIMARY]
GO

```

```

DROP TABLE [dbo].[STG_Movilidad]
GO

```

IN_MOVILIDAD

```

CREATE TABLE [dbo].[STG_Movilidad](
    [zonas_movilidad] [varchar] (27) NULL,
    [periodo] [datetime] NULL,
    [total] [decimal] (5, 2) NULL
) ON [PRIMARY]
GO

```

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

3.2. Creación del modelo multidimensional

En este punto veréis los *scripts* de creación del modelo físico multidimensional que habéis diseñado 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

DIM_Fecha

```
CREATE TABLE [dbo].[DIM_Fecha](
    [pk_fecha] [int] NOT NULL,
    [año] [int] NOT NULL,
    [mes] [int] NOT NULL,
    [día] [int] NOT NULL,
    [fecha] [date] NOT NULL,
    CONSTRAINT [PK_DIM_Fecha] PRIMARY KEY CLUSTERED (
        [pk_fecha] 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_Ambito_Geografico

```
CREATE TABLE [dbo].[DIM_Ambito_Geografico](
    [pk_ambito_geografico] [int] NOT NULL,
    [provincia_codigo] [varchar] (2) NOT NULL,
    [provincia_nombre] [varchar] (100) NOT NULL,
    [comunidad_autonoma] [varchar] (100) NULL,
    [comarca] [varchar] (100) NULL,
    [municipio] [varchar] (100) NULL,
    CONSTRAINT [PK_DIM_Ambito_Geografico] PRIMARY KEY CLUSTERED (
        [pk_ambito_geografico] 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_Grupo_Edad

```
CREATE TABLE [dbo].[DIM_Grupo_Edad](
    [pk_grupo_edad] [int] NOT NULL,
    [nombre] [varchar] (20) NOT NULL,
    [intervalo] [varchar] (20) NOT NULL,
    CONSTRAINT [PK_DIM_Grupo_Edad] PRIMARY KEY CLUSTERED (
        [pk_grupo_edad] 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_Medicion

```
CREATE TABLE [dbo].[DIM_Medicion](
    [pk_medicion] [int] NOT NULL,
    [nombre] [varchar] (100) NULL,
    [unidad_medida] [varchar] (20) NULL,
    CONSTRAINT [PK_DIM_Medicion] PRIMARY KEY CLUSTERED (
        [pk_medicion] 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_Tipologia

```
CREATE TABLE [dbo].[DIM_Tipologia](
    [pk_tipologia] [int] NOT NULL,
    [nombre] [nvarchar] (100) NOT NULL,
    CONSTRAINT [PK_DIM_Tipologia] PRIMARY KEY CLUSTERED (
        [pk_tipologia] 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
```


3.2.2. Tablas de hechos

FACT_Mediciones

```
CREATE TABLE [dbo].[FACT_Mediciones](
    [pk_id] [int] NOT NULL, [fk_fecha] [int] NOT NULL,
    [fk_ambito_geografico] [int] NOT NULL, [fk_grupo_edad] [int] NOT
    NULL,
    [fk_medicion] [int] NOT NULL,
    [valor] [decimal] (17, 2) NULL,
    CONSTRAINT [PK_DIM_Mediciones] 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_Llamadas112

```
CREATE TABLE [dbo].[FACT_Llamadas112](
    [pk_fk_fecha] [int] NOT NULL,
    [pk_fk_ambito_geografico] [bigint] NOT NULL, [pk_fk_tipologia]
    [int] NOT NULL,
    [llamadas] [int] NULL,
    CONSTRAINT [PK_FACT_MEDICIONES] PRIMARY KEY CLUSTERED (
        [pk_fk_fecha] ASC,
        [pk_fk_ambito_geografico] ASC,
        [pk_fk_tipologia] 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 [dbo].[FACT_Llamadas112] WITH CHECK ADD CONSTRAINT
    [FK_FACT_Llamadas112_DIM_Ambito_Geografico] FOREIGN KEY
    ([pk_fk_ambito_geografico])
    REFERENCES [dbo].[DIM_Ambito_Geografico] ([pk_ambito_geografico])
GO
```

```
ALTER TABLE [dbo].[FACT_Llamadas112] CHECK CONSTRAINT
    [FK_FACT_Llamadas112_DIM_Ambito_Geografico]
GO
```

```
ALTER TABLE [dbo].[FACT_Llamadas112] WITH CHECK ADD CONSTRAINT
    [FK_FACT_Llamadas112_DIM_Fecha] FOREIGN KEY([pk_fk_fecha])
    REFERENCES [dbo].[DIM_Fecha] ([pk_fecha])
GO
```

```
ALTER TABLE [dbo].[FACT_Llamadas112] CHECK CONSTRAINT
    [FK_FACT_Llamadas112_DIM_Fecha]
```

GO

```
ALTER TABLE [dbo].[FACT_Llamadas112] WITH CHECK ADD CONSTRAINT
[FK_FACT_Llamadas112_DIM_Tipologia] FOREIGN KEY([pk_fk_tipologia])
REFERENCES [dbo].[DIM_Tipologia] ([pk_tipologia])
GO
```

```
ALTER TABLE [dbo].[FACT_Llamadas112] CHECK CONSTRAINT
[FK_FACT_Llamadas112_DIM_Tipologia]
GO
```

```
ALTER TABLE [dbo].[FACT_Mediciones] WITH CHECK ADD CONSTRAINT
[FK_FACT_Mediciones_DIM_Ambito_Geografico] FOREIGN
KEY([fk_ambito_geografico])
REFERENCES [dbo].[DIM_Ambito_Geografico] ([pk_ambito_geografico])
GO
```

```
ALTER TABLE [dbo].[FACT_Mediciones] CHECK CONSTRAINT
[FK_FACT_Mediciones_DIM_Ambito_Geografico]
GO
```

```
ALTER TABLE [dbo].[FACT_Mediciones] WITH CHECK ADD CONSTRAINT
[FK_FACT_Mediciones_DIM_Fecha] FOREIGN KEY([fk_fecha])
REFERENCES [dbo].[DIM_Fecha] ([pk_fecha])
GO
```

```
ALTER TABLE [dbo].[FACT_Mediciones] CHECK CONSTRAINT
[FK_FACT_Mediciones_DIM_Fecha]
GO
```

```
ALTER TABLE [dbo].[FACT_Mediciones] WITH CHECK ADD CONSTRAINT
[FK_FACT_Mediciones_DIM_Grupo_Edad] FOREIGN KEY([fk_grupo_edad])
REFERENCES [dbo].[DIM_Grupo_Edad] ([pk_grupo_edad])
GO
```

```
ALTER TABLE [dbo].[FACT_Mediciones] CHECK CONSTRAINT
[FK_FACT_Mediciones_DIM_Grupo_Edad]
GO
```

```
ALTER TABLE [dbo].[FACT_Mediciones] WITH CHECK ADD CONSTRAINT
[FK_FACT_Mediciones_DIM_Medicion] FOREIGN KEY([fk_medicion])
REFERENCES [dbo].[DIM_Medicion] ([pk_medicion])
GO
```

```
ALTER TABLE [dbo].[FACT_Mediciones] CHECK CONSTRAINT
[FK_FACT_Mediciones_DIM_Medicion]
GO
```

3.3. Creación del proceso de extracción, transformación y carga

Una vez tenéis implementado el modelo físico del almacén, pasaréis 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* que habéis diseñado.

Antes del diseño de las transformaciones, definiréis en PDI las variables de entorno que usaréis en la implementación de los procesos de ETL, así como la conexión a la base de datos que utilizaréis en todos ellos.

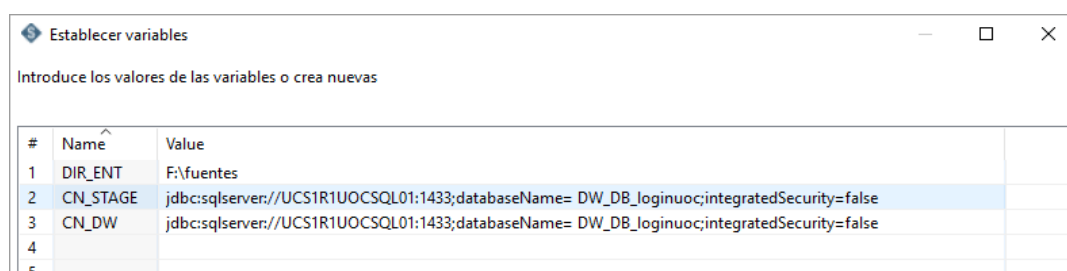
3.3.1. Variables de entorno

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».

En vuestro caso, utilizaréis 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_ENT	F:\fuentes
CN_STAGE	jdbc:sqlserver://UCS1R1UOCSQL01:1433;databaseName= DB_loginuoc;integratedSecurity=false
CN_DW	jdbc:sqlserver://UCS1R1UOCSQL01:1433;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_ENT}, {CN_STAGE}, {CN_DW}.

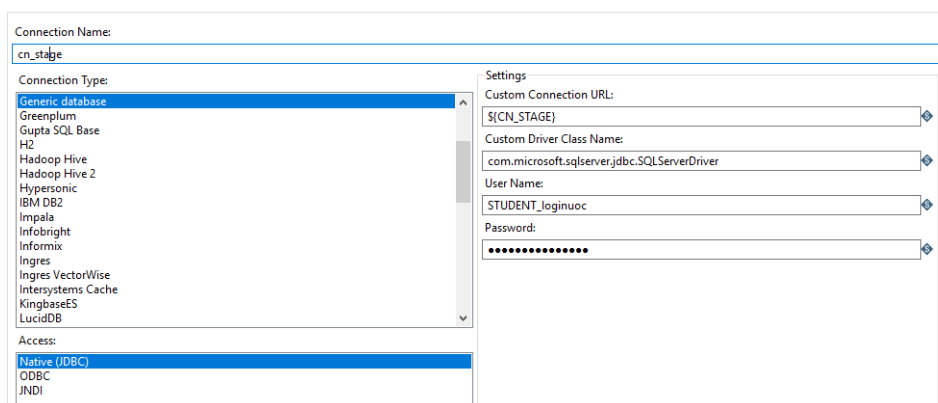


3.3.2. Conexión a la base de datos SQL Server

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 diferenciaréis claramente su uso, aunque físicamente se refieran al mismo esquema de la base de datos.

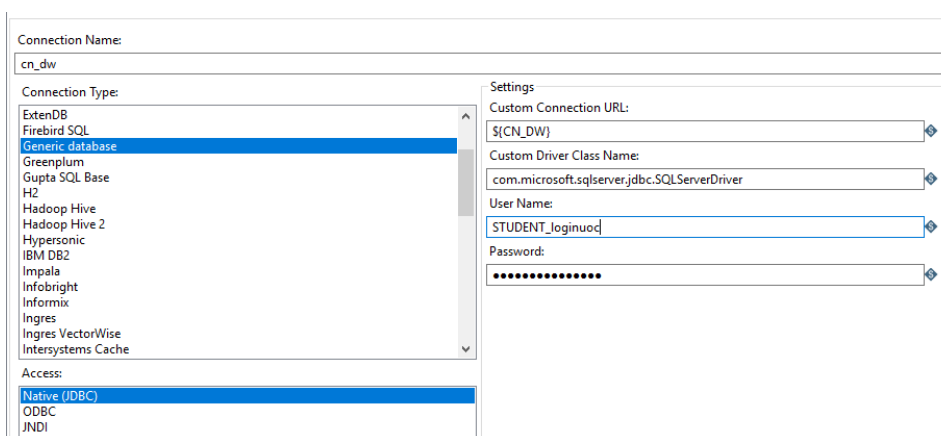
En la creación de la conexión al «STAGE», el nombre que utilizaréis es «cn_stage»:



The screenshot shows a connection configuration window. The 'Connection Name' field is set to 'cn_stage'. The 'Connection Type' dropdown 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: S(CN_STAGE)
- Custom Driver Class Name: com.microsoft.sqlserver.jdbc.SQLServerDriver
- User Name: STUDENT_loginuoc
- Password: (masked with dots)

En la creación de la conexión al «DW», el nombre que le daréis es «cn_dw»:



The screenshot shows a connection configuration window. The 'Connection Name' field is set to 'cn_dw'. The 'Connection Type' dropdown 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: S(CN_DW)
- Custom Driver Class Name: com.microsoft.sqlserver.jdbc.SQLServerDriver
- User Name: STUDENT_loginuoc
- Password: (masked with dots)

3.3.3. Bloque IN

En el bloque IN minimizaréis 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. Esto os permitirá trazar el dato en caso de ser necesario.

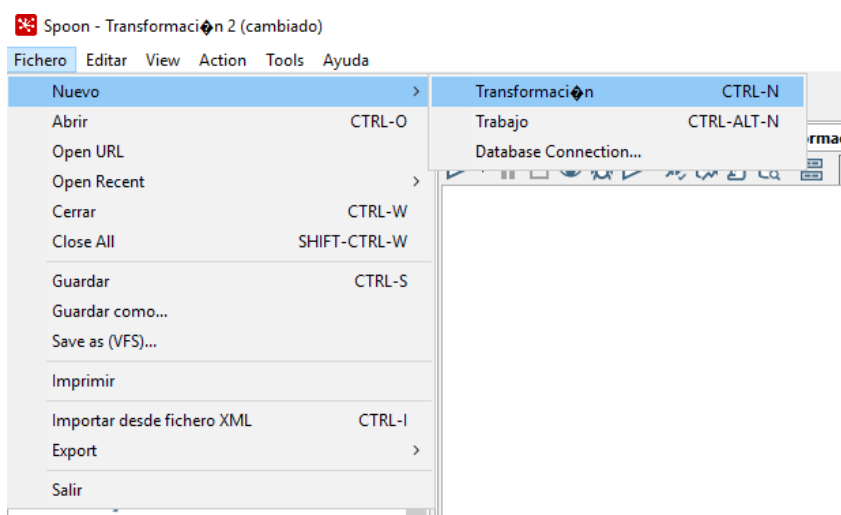
Transformación IN_DENUNCIAS_INFRACCIONES

STG_Denuncias_Infracciones

El primer proceso que se desarrollará será la carga de la fuente de «ACUMULADO-DENUNCIAS-INFRACCIONES.xlsx» a la tabla intermedia «STG_Denuncias_Infracciones» del *staging area*.

Para este caso práctico se utilizarán fuentes externas que emplearéis 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.

La transformación «IN_DENUNCIAS_INFRACCIONES» contiene cuatro conversiones: la lectura del fichero XLSX, las operaciones con cadenas, la clasificación de filas y la carga a la tabla intermedia «STG_Denuncias_Infracciones».



La entrada del fichero XLSX es el primer paso de la transformación. Para ello utilizaréis el tipo «Entrada Excel».



Entrada Excel (ACUMULADO-DENUNCIAS-INFRACCIONES.xlsx)

Indicaréis el fichero desde el que extraemos los datos; para ello, utilizaréis la variable de entorno «DIR_ENT» y señalaréis el tipo de motor que se deberá usar y si es para ficheros de tipo XLS o XLSX.

Indicaréis qué hojas del fichero origen deberá tener en cuenta y desde qué fila y columna tendrá que empezar a leer los datos.

Diseño y uso de bases de datos analíticas
Estudios de Informática, Multimedia y Telecomunicación

2021

pág. 14

También tendréis que indicarle que recupere los campos que vamos a tratar mediante el botón «Traer campos» y completará la definición de los campos, especificando donde se considere necesario la precisión y la longitud de los campos e indicando el formato de fecha en los campos «Date».

Si en los campos de tipo «Number» se indica una longitud y precisión con un valor de -1 estos se tratarán como *float*, o, lo que lo mismo, un dato numérico de coma flotante.

Entrada Excel

Nombre paso: Entrada Excel (ACUMULADO-DENUNCIAS-INFRACCIONES.xlsx)

#	Nombre	Tipo	Longitud	Precisión	Tipo de poda	Repetir	Formato	Moneda
1	provincia	String	100	-1	ninguno	N		
2	identificados_ertaintza	Number	-1	-1	ninguno	N	#	
3	detenidos_ertaintza	Number	-1	-1	ninguno	N	#	
4	denuncias_ertaintza	Number	-1	-1	ninguno	N	#	
5	vehic_intercept_ertaintza	Number	-1	-1	ninguno	N	#	
6	identificados_ppll	Number	-1	-1	ninguno	N	#	
7	detenidos_ppll	Number	-1	-1	ninguno	N	#	
8	denuncias_ppll	Number	-1	-1	ninguno	N	#	
9	vehic_intercept_ppll	Number	-1	-1	ninguno	N	#	
10	fecha_final	Date	-1	-1	ninguno	N	dd/MM/yyyy	

Obtener campos de cabecera...

Help Vale Previsualizar filas Cancelar


Para realizar una visualización previa de los datos que se cargarán se utiliza el botón «Previsualizar filas».

Examine preview data

Rows of step: Entrada Excel (ACUMULADO-DENUNCIAS-INFRACCIONES.xlsx) (219 rows)

#	provincia	identificados_ertaintza	detenidos_ertaintza	denuncias_ertaintza	vehic_intercept_ertaintza	identificados_ppll	detenidos_ppll	denuncias_ppll	vehic_intercept_ppll	fecha_final
1	ARABA	10717	40	2586	15182	21599	29	2748	19250	18/06/2020
2	BIZKAIA	29955	228	6249	56139	27160	65	9209	50339	18/06/2020
3	GIPIUZKOA	26051	62	4884	17246	27069	40	4473	37787	18/06/2020
4	ARABA	10708	40	2586	15180	21598	29	2748	19250	17/06/2020
5	BIZKAIA	29867	228	6249	56009	27138	65	9209	50518	17/06/2020
6	GIPIUZKOA	26004	62	4882	17183	27059	40	4472	37791	17/06/2020
7	ARABA	10705	40	2585	15176	21598	29	2748	19250	16/06/2020
8	BIZKAIA	29751	228	6249	55859	27124	65	9209	50468	16/06/2020
9	GIPIUZKOA	25942	62	4882	17173	27038	38	4465	37782	16/06/2020
10	ARABA	10704	40	2585	15176	21593	29	2746	19250	15/06/2020
11	BIZKAIA	29674	228	6247	55754	27106	65	9203	50436	15/06/2020
12	GIPIUZKOA	25872	62	4879	17103	27022	38	4465	37770	15/06/2020
13	ARABA	10700	40	2585	15173	21573	29	2739	19250	14/06/2020
14	BIZKAIA	29597	228	6246	55674	27063	65	9202	50428	14/06/2020
15	GIPIUZKOA	25775	62	4873	17003	27015	38	4465	37754	14/06/2020
16	ARABA	10652	40	2572	15134	21572	29	2738	19250	13/06/2020
17	BIZKAIA	29477	228	6236	55470	27055	65	9201	50386	13/06/2020
18	GIPIUZKOA	25672	62	4862	16924	27004	38	4464	37744	13/06/2020

El siguiente paso de la transformación es asegurar la calidad de los datos mediante la normalización de los valores de los campos «String»; para ello, convertiréis los datos de las fuentes origen a mayúsculas y eliminaréis los espacios en blanco que pudiera haber al inicio y al final de la cadena de caracteres mediante el componente «String operations».



String operations

String operations

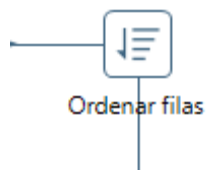
Step name: String operations

The fields to process:

#	In stream field	Out stream field	Trim type	Lower/Upper	Padding	Pad char
1	provincia		both	upper	none	

Help Vale Get fields Cancelar

El siguiente paso de la transformación sería la ordenación ascendente de los campos. Para ello, utilizaréis el componente «Ordenar filas» de las posibles transformaciones disponibles.



Ordenar filas

Ordenar filas

Nombre paso: Ordenar filas

Directorio ordenación: %%java.io.tmpdir%% Examinar...

Prefijo para ficheros temporales: out

Tamaño de ordenación (filas en memoria): 1000000

Free memory threshold (in %):

Comprimir ficheros temporales? ☐

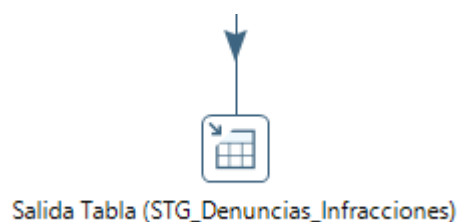
Only pass unique rows? (verifies keys only) ☐

Campos:

#	Nombre Campo	Ascendente	Case sensitive compare?	Sort based on current locale?	Collator Strength	Presorted?
1	provincia	S	N	N	0	N
2	identificados_ertaintza	S	N	N	0	N
3	detenidos_ertaintza	S	N	N	0	N
4	denuncias_ertaintza	S	N	N	0	N
5	vehic_intercept_ertaintza	S	N	N	0	N
6	identificados_ppll	S	N	N	0	N
7	detenidos_ppll	S	N	N	0	N
8	denuncias_ppll	S	N	N	0	N
9	vehic_intercept_ppll	S	N	N	0	N
10	fecha_final	S	N	N	0	N

Help Vale Cancelar Traer Campos

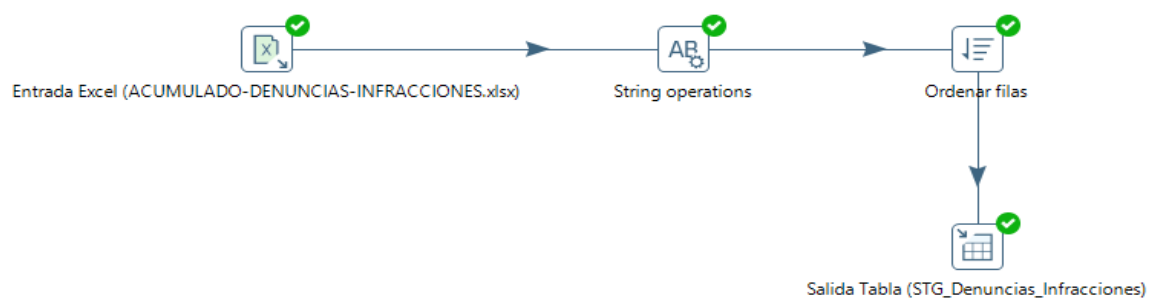
Por último, cargaréis los datos en la tabla intermedia del stage, utilizando el paso «Salida Tabla» de la carpeta «Salida». Este paso necesita especificar la conexión de la base de datos; para ello utilizaréis la variable de entorno «CN_STAGE» que habéis definido.



El paso de la carga de datos a la tabla intermedia del *stage* lo configuráis como se muestra en el menú principal:

Para dejar la transformación preparada para posibles reprocesos, es necesario realizar un borrado previo que actualice los datos en caso de que se vuelva a procesar. Para esto, activáis el botón «Vaciar tabla». En los campos de la base de datos:

La transformación completa es la siguiente:



El resultado de la ejecución es el siguiente:

Execution Results

Logging Execution History Step Metrics Performance Graph Metrics Preview data

#	Nombre paso	Numero Copia	Leído	Escrito	Entrada	Salida	Actualizado	Rejected	Errores	Activo	Tiempo
1	Entrada Excel (ACUMULADO-DENUNCIAS-INFRACCIONES.xlsx)	0	0	219	219	0	0	0	0	Finalizado	2.4s
2	String operations	0	219	219	0	0	0	0	0	Finalizado	2.4s
3	Ordenar filas	0	219	219	0	0	0	0	0	Finalizado	2.4s
4	Salida Tabla (STG_Denuncias_Infracciones)	0	219	219	0	219	0	0	0	Finalizado	2.6s

Como se observa en las métricas, se cargan los 219 registros del fichero de entrada.

Transformación IN_POBLACION

STG_Poblacion

Esta transformación contiene cinco pasos: la lectura del fichero CSV, la partición de campos, el cambio a mayúsculas, la ordenación de datos y la carga a la tabla intermedia.

Para poder cargar la tabla «STG_Poblacion», el origen de datos debe ser un fichero CSV. En el paso «CSV file input» seleccionaréis el archivo que queréis cargar utilizando la variable «DIR_ENT» y determinaréis, con la ayuda del PDI, los campos, el delimitador de campos y el tipo de dato.



CSV file input (población_9687.csv)

CSV file input

Step name: CSV file input (población_9687.csv)

Filename: \${DIR_ENT}\poblacion_9687bsc.csv Examinar...

Delimiter: ; Insert TAB

Enclosure: "

NIO buffer size: 50000

Lazy conversion? ☒

Header row present? ☒

Add filename to result ☐

The row number field name (optional):

Running in parallel? ☐

New line possible in fields? ☐

Format: mixed

File encoding:

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type
1	Edad Simple	String		5		€	,	.	ninguno
2	Provincias	String		25		€	,	.	ninguno
3	Sexo	String		11		€	,	.	ninguno
4	Periodo	String		25		€	,	.	ninguno
5	Total	Number	#,###,###.#	15	0	€	,	.	ninguno

< >

? Help Vale Traer Campos Previsualizar Cancelar

Previsualizaréis los datos:

Examine preview data

Rows of step: CSV file input (población_9687.csv) (52 rows)

#	Edad Simple	Provincias	Sexo	Periodo	Total
1	Total	02 Albacete	Ambos sexos	1 de enero de 2020	389.830
2	Total	03 Alicante/Alacant	Ambos sexos	1 de enero de 2020	1.885.214
3	Total	04 Almería	Ambos sexos	1 de enero de 2020	715.406
4	Total	01 Araba/Álava	Ambos sexos	1 de enero de 2020	329.857
5	Total	33 Asturias	Ambos sexos	1 de enero de 2020	1.018.775
6	Total	05 Ávila	Ambos sexos	1 de enero de 2020	158.930
7	Total	06 Badajoz	Ambos sexos	1 de enero de 2020	670.782
8	Total	07 Baleares, Illes	Ambos sexos	1 de enero de 2020	1.210.750

En la siguiente transformación separaréis el campo «Provincias» en dos campos nuevos: «provincia_codigo» y «provincia_nombre».



Para ello indicaréis como separador el espacio entre palabras o caracteres.

Partir campo

Nombre paso: Partir campos

Campo a partir: Provincias

Separador:

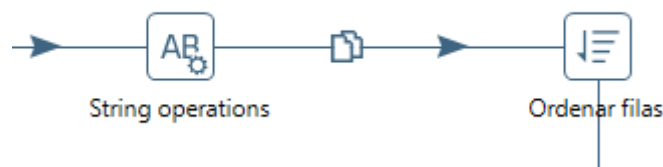
Enclosure:

Campos

#	Nuevo campo	ID	Eliminar ID?	Tipo	Longitud	Precisión	F
1	provincia_codigo		N	String	2		
2	provincia_nombre		N	String	100		

Help Vale Cancelar

En las dos siguientes transformaciones normalizaréis el texto en mayúsculas, eliminaréis los espacios en blanco de cada campo de tipo *string* y ordenaréis los registros.



Finalmente, cargaréis los datos en la tabla intermedia.

Salida de Tabla

Nombre paso: Salida Tabla (STG_Poblacion)

Conexión: cn_stage [Editar...] [Nuevo...] [Wizard...]

Esquema destino: dbo [Examinar...]

Tabla destino: STG_Poblacion [Examinar...]

Tamaño transacción (commit): 1000

Vaciar tabla: ☒

Ignorar errores de inserción: ☐

Specify database fields: ☒

Main options Database fields

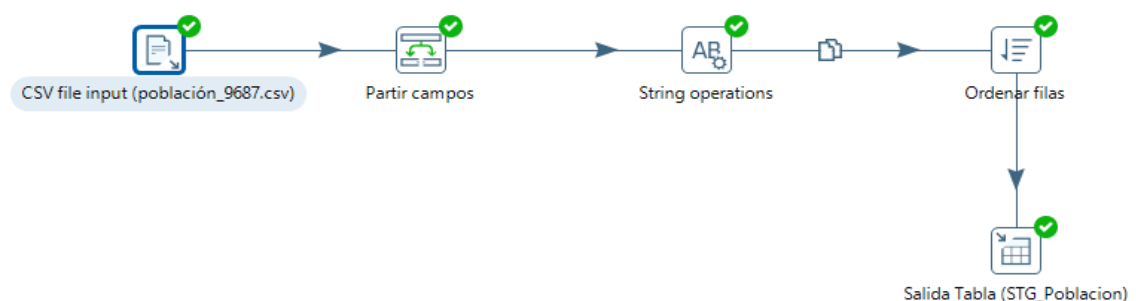
Fields to insert:

#	Table field	Stream field
1	provincia_codigo	provincia_codigo
2	provincia_nombre	provincia_nombre
3	periodo	Periodo
4	poblacion	Total

[Get fields] [Enter field mapping]

[?] Help [Vale] [Cancelar] [SQL]

La transformación completa es la siguiente:



El resultado de la ejecución es el siguiente:

Execution Results

Logging Execution History Step Metrics Performance Graph Metrics Preview data

#	Nombre paso	Numero Copia	Leído	Escrito	Entrada	Salida	Actualizado	Rejected	Errores	Activo	Tiempo
1	CSV file input (población_9687.csv)	0	0	52	53	0	0	0	0	Finalizado	0.0
2	Partir campos	0	52	52	0	0	0	0	0	Finalizado	0.0
3	String operations	0	52	52	0	0	0	0	0	Finalizado	0.1
4	Ordenar filas	0	52	52	0	0	0	0	0	Finalizado	0.1
5	Salida Tabla (STG_Poblacion)	0	52	52	0	52	0	0	0	Finalizado	0.2

Como se observa en las métricas, se cargan los 52 registros del fichero de entrada.

Transformación IN_MOVILIDAD

STG_Movilidad

Esta transformación contiene cuatro pasos: la lectura del fichero CSV, el cambio a mayúsculas, la ordenación de datos y la carga a la tabla intermedia.

Para cargar la tabla «STG_Movilidad», el origen de los datos debe ser un fichero CSV. En el paso «CSV file input» seleccionará el archivo a cargar utilizando la variable «DIR_ENT» y determinará, con la ayuda del PDI, los campos, el delimitador de campos y el tipo de dato.



CSV file input (35167bsc.csv)

CSV file input

Step name: CSV file input (35167bsc.csv)

Filename: \${DIR_ENT}\35167bsc.csv Examinar...

Delimiter: ; Insert TAB

Enclosure: "

NIO buffer size: 50000

Lazy conversion? ☒

Header row present? ☒

Add filename to result ☐

The row number field name (optional):

Running in parallel? ☐

New line possible in fields? ☐


Format:

File encoding:

#	Name	Type	Format	Length	Precision	Currency	Decima
1	Zonas de movilidad	String		27		€	,
2	Periodo	Date	dd/MM/yyyy			€	,
3	Total	Number	#,##	5	2	€	,

? Help
 Vale
Traer Campos
Previsualizar
Cancelar

Previsualizaréis los datos:

 Examine preview data


Rows of step: CSV file input (35167bsc.csv) (1000 rows)

#	Zonas de movilidad	Periodo	Total
1	Almería	20/06/2020	15,15
2	Almería	19/06/2020	16,91
3	Almería	18/06/2020	16,83
4	Almería	17/06/2020	16,9
5	Almería	16/06/2020	16,64
6	Almería	15/06/2020	16,43
7	Almería	14/06/2020	12,70

En las dos siguientes transformaciones normalizaréis el texto en mayúsculas, eliminaréis los espacios en blanco de cada campo de tipo *string* y ordenaréis los registros.



Finalmente, cargaréis los datos en la tabla intermedia.

 Salida de Tabla

Nombre paso: Salida Tabla (STG_Movilidad)

Conexión: cn_stage Editar... Nuevo... Wizard...

Esquema destino: dbo Examinar...

Tabla destino: STG_Movilidad Examinar...

Tamaño transacción (commit): 1000

Vaciar tabla: ☒

Ignorar errores de inserción: ☐

Specify database fields: ☒

Main options Database fields

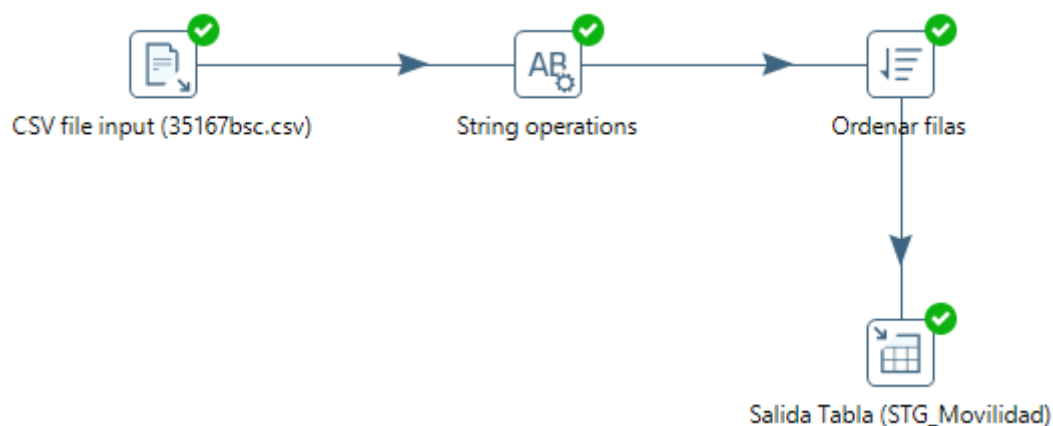
Fields to insert:

#	Table field	Stream field
1	zonas_movilidad	Zonas de movilidad
2	periodo	Periodo
3	total	Total

Get fields Enter field mapping

Help Vale Cancelar SQL

La transformación completa es la siguiente:



El resultado de la ejecución es el siguiente:

Execution Results

Logging Execution History Step Metrics Performance Graph Metrics Preview data

#	Nombre paso	Numero Copia	Leído	Escrito	Entrada	Salida	Actualizado	Rejected	Errores	Activo	Tiempo
1	CSV file input (35167bsc.csv)	0	0	4732	4733	0	0	0	0	Finalizado	0.0s
2	String operations	0	4732	4732	0	0	0	0	0	Finalizado	0.2s
3	Ordenar filas	0	4732	4732	0	0	0	0	0	Finalizado	0.2s
4	Salida Tabla (STG_Movilidad)	0	4732	4732	0	4732	0	0	0	Finalizado	0.5s

Como se observa en las métricas, se cargan los 4732 registros del fichero de entrada.

Transformación IN_LLAMADAS112

STG_Llamadas112

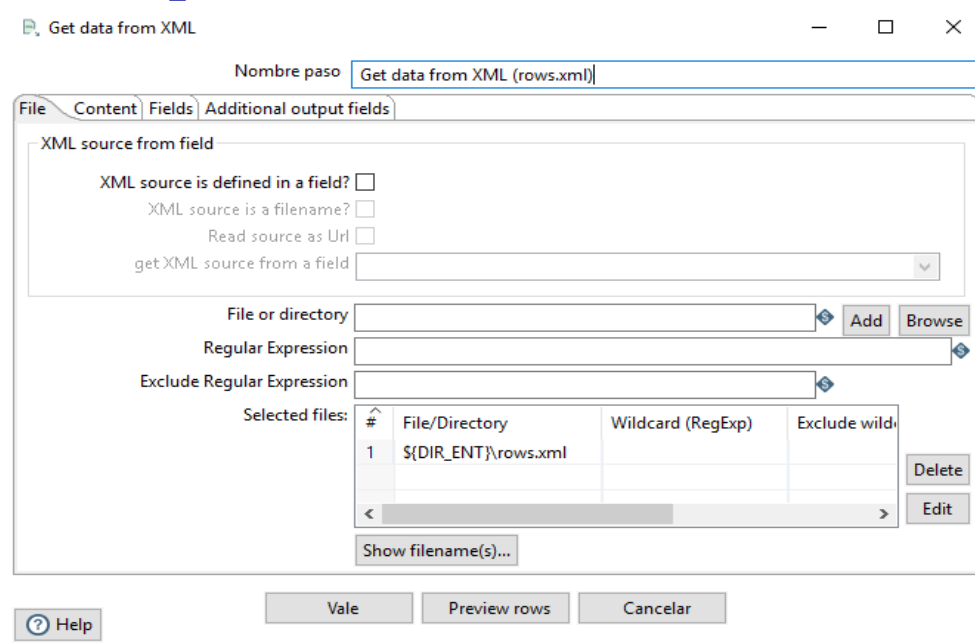
Esta transformación contiene cuatro pasos: la lectura del fichero XML, el cambio a mayúsculas, la ordenación de datos y la carga a la tabla intermedia.

Para cargar la tabla «STG_Llamadas112» el origen de datos debe ser un fichero XML. En el paso «Get data from XML» seleccionaréis el archivo a cargar utilizando la variable «DIR_ENT» y determinaréis, con la ayuda de PDI, los campos, el delimitador de campos y el tipo de dato.



Get data from XML (rows.xml)

Indicaréis el fichero desde donde extraéis los datos; para ello, utilizaréis la variable de entorno «DIR_ENT».



Get data from XML

Nombre paso: Get data from XML (rows.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: Add Browse

Regular Expression:

Exclude Regular Expression:

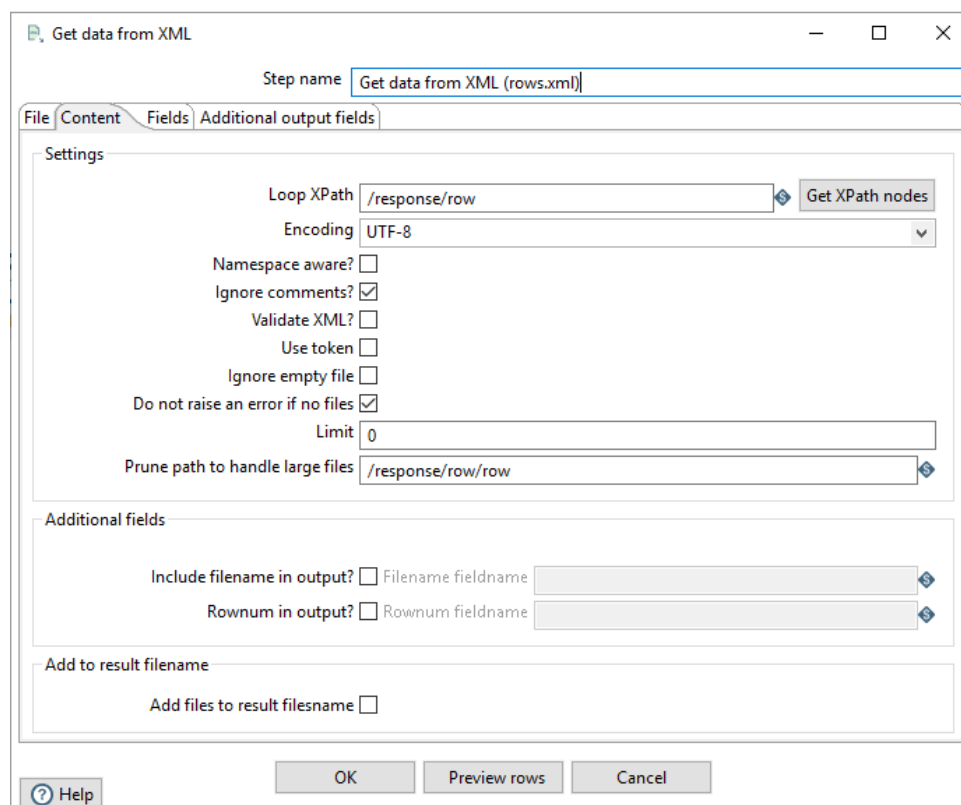
Selected files:

#	File/Directory	Wildcard (RegExp)	Exclude wild	
1	\$(DIR_ENT)\rows.xml			Delete Edit

Show filename(s)...

Help Vale Preview rows Cancelar

A continuación, deberéis indicar la ruta o *path* que el PDI deberá iterar para leer cada registro. Para ello indicaréis en «Loop XPath» la ruta correspondiente al registro. Dado que es un fichero grande, tendréis que utilizar la opción «Prune path to handle large files», de modo que PDI sea capaz de procesar los registros en modo *streaming* o por trozos de datos.



Get data from XML

Step name: Get data from XML (rows.xml)

File Content Fields Additional output fields

Settings

Loop XPath: /response/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: /response/row/row

Additional fields

Include filename in output? ☐ Filename fieldname:

Rownum in output? ☐ Rownum fieldname:

Add to result filename

Add files to result filename ☐

Help OK Preview rows Cancel

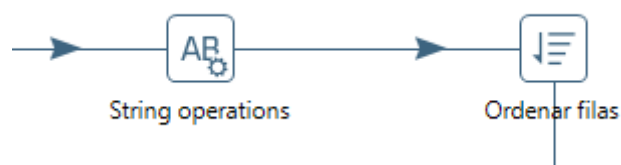
Así mismo tendréis que indicarle que recupere los campos que vamos a tratar mediante el botón «Get fields» y completará la definición de los campos, especificando el tipo de dato y, donde se considere necesario, la precisión y la longitud de los campos.

#	Name	XPath	Element	Result type	Type	Format	Length	Precs
1	any	row/any	Node	Value of	Integer			
2	mes	row/mes	Node	Value of	Integer			
3	provincia	row/provincia	Node	Value of	String			
4	comarca	row/comarca	Node	Value of	String			
5	municipi	row/municipi	Node	Value of	String			
6	tipus	row/tipus	Node	Value of	String			
7	trucades	row/trucades	Node	Value of	Integer			

Para realizar una visualización previa de los datos que se cargarán se utiliza el botón «Preview rows».

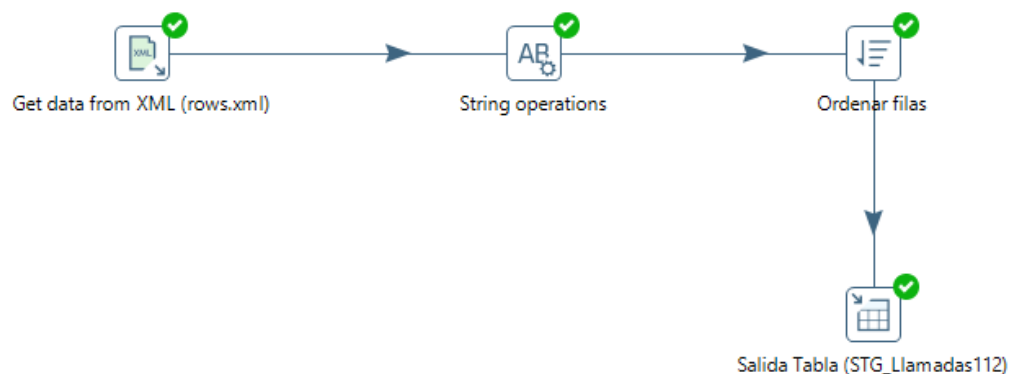
#	any	mes	provincia	comarca	municipi	tipus	trucades
1	2014	1	BARCELONA	ALT PENEDES	AVINYONET DEL PENEDES	Seguretat	10
2	2014	1	BARCELONA	ALT PENEDES	AVINYONET DEL PENEDES	Trànsit	7
3	2014	1	BARCELONA	ALT PENEDES	AVINYONET DEL PENEDES	Incendi	1
4	2014	1	BARCELONA	ALT PENEDES	AVINYONET DEL PENEDES	Assistència sanitària	6
5	2014	1	BARCELONA	ALT PENEDES	AVINYONET DEL PENEDES	Medi ambient	1
6	2014	1	BARCELONA	ALT PENEDES	CASTELLET I LA GORNAL	Seguretat	12
7	2014	1	BARCELONA	ALT PENEDES	CASTELLET I LA GORNAL	Trànsit	30
8	2014	1	BARCELONA	ALT PENEDES	CASTELLET I LA GORNAL	Civisme	1

En las dos siguientes transformaciones normalizaréis el texto en mayúsculas, eliminaréis los espacios en blanco de cada campo de tipo *string* y ordenaréis los registros.



Finalmente, cargaréis los datos en la tabla intermedia.

La transformación completa es la siguiente:



El resultado de la ejecución es el siguiente:

Execution Results

Logging Execution History Step Metrics Performance Graph Metrics Preview data

#	Nombre paso	Numero Copia	Leído	Escrito	Entrada	Salida	Actualizado	Rejected	Errores	Activo	Tiempo
1	Get data from XML (rows.xml)	0	0	340307	340307	0	0	0	0	Finalizado	11.5s
2	String operations	0	340307	340307	0	0	0	0	0	Finalizado	11.5s
3	Ordenar filas	0	340307	340307	0	0	0	0	0	Finalizado	19.6s
4	Salida Tabla (STG_Llamadas112)	0	340307	340307	0	340307	0	0	0	Finalizado	19.8s

Como se observa en las métricas, se cargan los 340.307 registros del fichero de entrada.

Transformación IN_EVITAR_AGLOMERACION

STG_Evitar_Aglomeracion

Esta transformación contiene once pasos: la lectura del fichero XLSX, el reemplazo en *string*, la filtración de las filas, dos pasos tipo «partir campos», la fundición de las filas, el reemplazo en *string*, la normalización de las filas, el cambio a mayúsculas, la ordenación de datos y la carga a la tabla intermedia.

Para cargar la tabla «STG_Evitar_Aglomeracion» el origen de datos debe ser un fichero XLSX. En el paso «Entrada Excel» seleccionaráis el archivo a cargar utilizando la variable «DIR_ENT» e indicaréis el tipo de motor que deberá usar el PDI y si es para ficheros de tipo XLS o XLSX.



Entrada Excel (evitar_aglomeraciones.xlsx)

Le indicaráis qué hojas del fichero origen deberá tener en cuenta y desde qué fila y columna deberá empezar a leer los datos.

Entrada Excel

Nombre paso: Entrada Excel (evitar_aglomeraciones.xlsx)

Ficheros | Hojas | Contenido | Manejador de Error | Campos | Additional output fields

Lista de hojas a leer

#	Nombre hoja	Fila inicial	Columna inicial
1	Datos_provincias	3	2

Obtener hoja(s)...

Help Vale Previsualizar filas Cancelar

En la pestaña de «Contenido» le indicaráis que el fichero contiene una fila a modo de cabecera y que deberá eliminar filas vacías.

Entrada Excel

Nombre paso: Entrada Excel (evitar_aglomeraciones.xlsx)

Ficheros | Hojas | Contenido | Manejador de Error | Campos | Additional output fields

Cabecera ☒

Eliminar filas vacías ☒

Detener al encontrar fila vacía ☐

Límite: 0

Encoding: [dropdown]

Result filenames

Add filenames to result ☒

Help Vale Previsualizar filas Cancelar

Así mismo le indicaráis que recupere los campos que vais a tratar mediante el botón «Obtener campos de cabecera...» y completaráis la definición de los campos, especificando donde se considere necesario la precisión y la longitud de los campos.

Si en los campos de tipo «Number» se indica una longitud y precisión con un valor de -1 estos se tratarán como *float* o, lo que lo mismo, dato numérico de coma flotante.

Entrada Excel

Nombre paso: Entrada Excel (evitar_aglomeraciones.xlsx)

#	Nombre	Tipo	Longitud	Precisión	Tipo de poda	Repetir
1	ambito_geografico	String	100	-1	ninguno	N
2	14 - 24	Number	-1	-1	ninguno	N
3	25 - 34	Number	-1	-1	ninguno	N
4	35 - 44	Number	-1	-1	ninguno	N
5	45 - 54	Number	-1	-1	ninguno	N
6	55 - 64	Number	-1	-1	ninguno	N
7	> 64	Number	-1	-1	ninguno	N

Obtener campos de cabecera...

Help Vale Previsualizar filas Cancelar

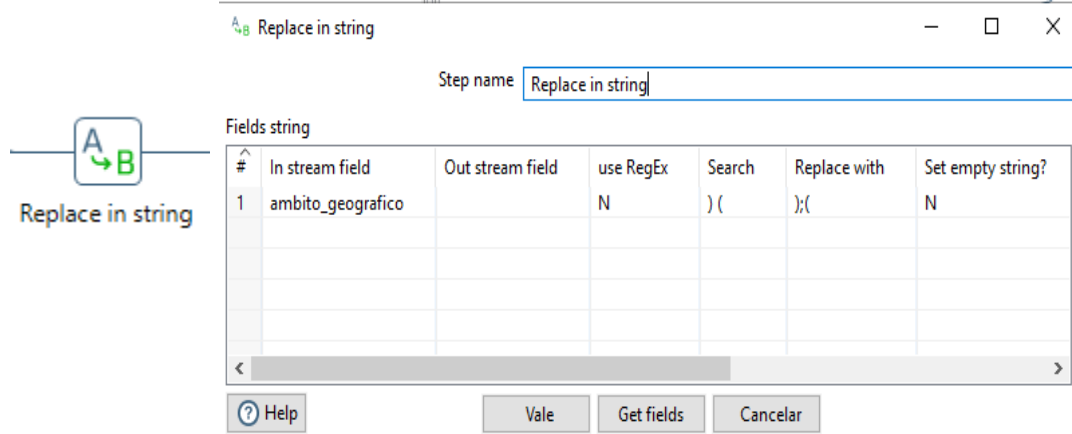
Previsualizaréis los datos:

Examine preview data

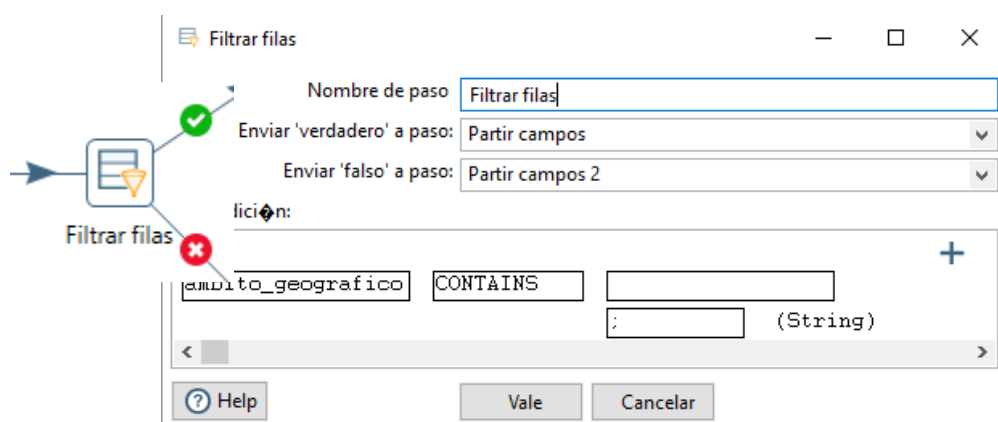
Rows of step: Entrada Excel (evitar_aglomeraciones.xlsx) (50 rows)

#	ambito_geografico	14 - 24	25 - 34	35 - 44	45 - 54	55 - 64	> 64
1	Álava (Araba) (País Vasco)	36,9946304598	52,4568043714	42,702313554	46,6583491132	35,7307759155	51,1323225462
2	Albacete (Castilla-La Mancha)	48,781028035	38,8466188827	42,8388157265	45,8441179625	54,8485278305	44,7822885168
3	Alicante (Comunidad Valenciana)	41,2393919951	52,2916491389	56,2879441856	32,4585638862	49,2640560672	52,3947287942
4	Almería (Andalucía)	51,7484218525	52,837236963	35,2176770346	39,1624911133	37,8470777254	55,9155999017
5	Asturias (Principado de Asturias)	36,6318400468	44,7314456548	38,3242946263	51,0233449144	50,9015839034	33,7238117377
6	Ávila (Castilla y León)	32,8598609002	37,8402077605	55,7358592272	50,8733114005	41,6009734858	44,7055901675
7	Badajoz (Extremadura)	46,4487281735	51,9255091772	52,8612449593	43,2467592732	53,7738867396	36,767456579
8	Barcelona (Cataluña)	39,5744935922	36,0866268039	48,2311885489	56,7298617297	48,8668217476	46,0904554301
9	Burgos (Castilla y León)	55,2515186534	36,8898852459	47,1193940544	54,4680144175	48,2893362105	54,7569172862
10	Cáceres (Extremadura)	38,8936189496	36,9597259747	52,7106496589	36,3819314146	46,8114877198	49,3691457522
11	Cádiz (Andalucía)	41,6653808098	34,5743199759	51,6344805451	53,5085784724	56,2599932981	50,9276458268

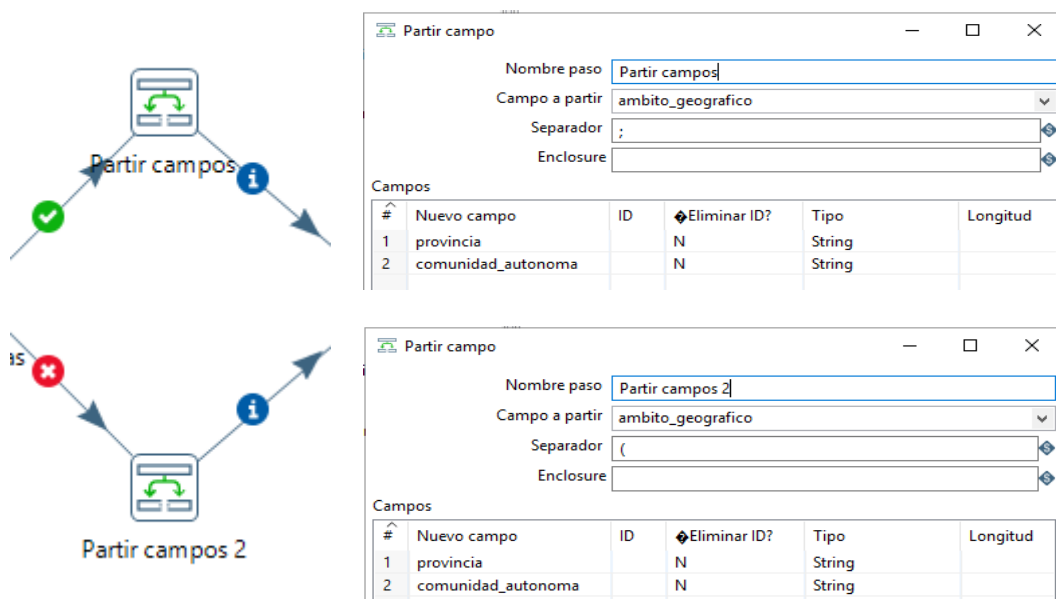
En la siguiente transformación cambiaréis la secuencia de caracteres «)» (» – cierre de paréntesis, espacio en blanco y apertura de paréntesis– por «);» (» – cierre de paréntesis, punto y coma y apertura de paréntesis– en el campo «ambito_geografico». De esta manera prepararéis el campo para su posterior separación en dos campos, de forma que diferenciéis la provincia de la comunidad autónoma.



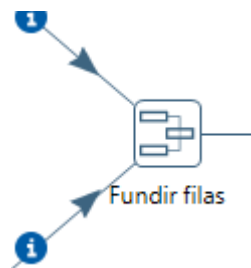
Separaréis los registros que contienen punto y coma de los que no lo contienen.



A continuación, utilizaréis dos pasos de tipo «Partir campos», con los que separaréis la provincia de la comunidad autónoma. Para ello indicaréis como separador en un caso el punto y coma, y en el otro la apertura de paréntesis.



La siguiente transformación unirá los registros de los dos pasos de tipo «Partir campos» anteriores.



Se deben indicar los dos orígenes, así como los campos clave de cada uno de los orígenes por los que se van a unir los registros. En este caso, el campo «provincia».

Mezclar filas

Nombre de paso: Fundir filas

Origen filas Referencia: Partir campos

Origen filas: Partir campos 2

Nombre de campo: flagfield

Claves coincidentes:

#	Campo clave
1	provincia

Valores a comparar:

#	Campo valor
1	provincia

Obtener campos clave Obtener campos valor

Help Vale Cancelar

El siguiente paso limpiará el campo «comunidad autónoma» a través de la sustitución del carácter de paréntesis por un espacio en blanco.

Replace in string

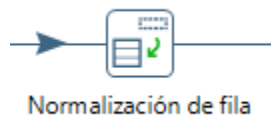
Step name: Replace in string 2

Fields string

#	In stream field	Out stream field	use RegEx	Search	Replace with	Set
1	comunidad_autonoma		N	(N
2	comunidad_autonoma		N)		N

Help Vale Get fields Cancelar

Después se realizará un paso para normalizar las filas, de modo que cada ámbito geográfico tenga tantos registros como grupos de edad de los que se dispongan. El campo que contendrá el valor de cada combinación de ámbito geográfico y grupo de edad se llamará «porc_poblacion».



Normalizar filas

Nombre de paso: Normalización de fila

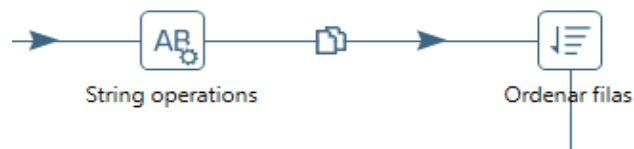
Tipo de campo: grupo_edad

Campos

#	Nombre campo	Tipo	campo nuevo
1	14 - 24	14 - 24	porc_poblacion
2	25 - 34	25 - 34	porc_poblacion
3	35 - 44	35 - 44	porc_poblacion
4	45 - 54	45 - 54	porc_poblacion
5	55 - 64	55 - 64	porc_poblacion
6	> 64	> 64	porc_poblacion

Help Vale Cancelar Obtener campos

En las dos siguientes transformaciones normalizaréis el texto en mayúsculas, eliminaréis los espacios en blanco de cada campo de tipo *string* y ordenaréis los registros.



Finalmente, cargaréis los datos en la tabla intermedia.

Salida de Tabla

Nombre paso: Salida Tabla (STG_Evitar_Aglomeracion)

Conexión: cn_stage [Editar...] [Nuevo...] [Wizard...]

Esquema destino: dbo [Examinar...]

Tabla destino: STG_Evitar_Aglomeracion [Examinar...]

Tamaño transacción (commit): 1000

Vaciar tabla: ☒

Ignorar errores de inserción: ☐

Specify database fields: ☒

Main options Database fields

Fields to insert:

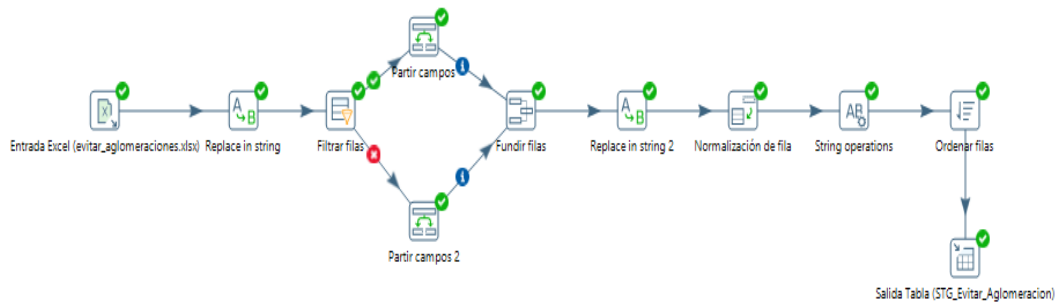
#	Table field	Stream field
1	provincia	provincia
2	comunidad_autonoma	comunidad_autonoma
3	grupo_edad	grupo_edad
4	porc_poblacion	porc_poblacion

Get fields

Enter field mapping

Help Vale Cancelar SQL

La transformación completa es la siguiente:



El resultado de la ejecución es el siguiente:

Execution Results

Logging Execution History Step Metrics Performance Graph Metrics Preview data

#	Nombre paso	Numero Copia	Leído	Escrito	Entrada	Salida	Actualizado	Rejected	Errores	Activo	Tiempo
1	Entrada Excel (evitar_aglomeraciones.xlsx)	0	0	50	50	0	0	0	0	Finalizado	0.1s
2	Replace in string	0	50	50	0	0	0	0	0	Finalizado	0.1s
3	Filtrar filas	0	50	50	0	0	0	0	0	Finalizado	0.1s
4	Partir campos 2	0	43	43	0	0	0	0	0	Finalizado	0.1s
5	Partir campos	0	7	7	0	0	0	0	0	Finalizado	0.1s
6	Fundir filas	0	50	50	0	0	0	0	0	Finalizado	0.4s
7	Replace in string 2	0	50	50	0	0	0	0	0	Finalizado	0.5s
8	Normalización de fila	0	50	300	0	0	0	0	0	Finalizado	0.5s
9	String operations	0	300	300	0	0	0	0	0	Finalizado	0.5s
10	Ordenar filas	0	300	300	0	0	0	0	0	Finalizado	0.5s
11	Salida Tabla (STG_Evitar_Aglomeracion)	0	300	300	0	300	0	0	0	Finalizado	0.5s

Como se observa en las métricas, se cargan los cincuenta registros del fichero de entrada que, al normalizar las filas, se transforman en 300 registros en la salida.

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 que habéis 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. Tampoco se detallarán las transformaciones de selección de campos o de modificación de tipos de campos.

3.3.5. Bloque TR_DIM

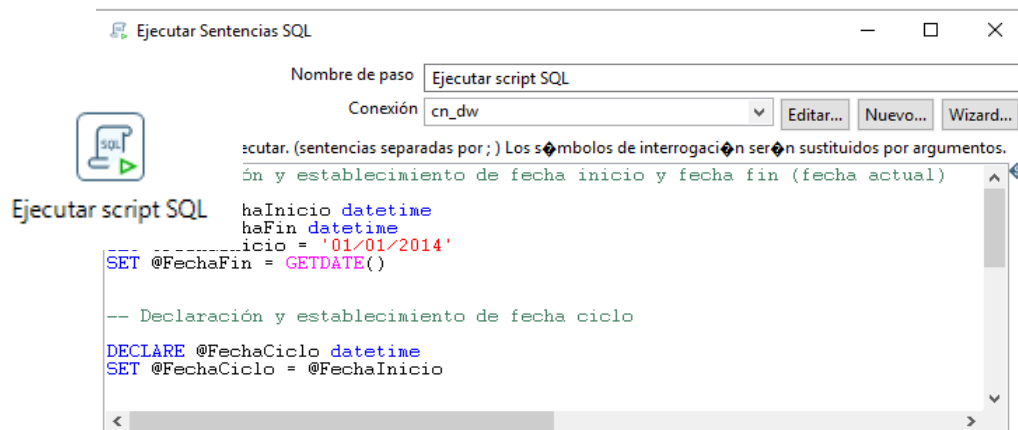
Este bloque contiene las transformaciones para la carga inicial de las dimensiones al almacén desde las tablas intermedias «IN_ del staging area».

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

Transformación TR_DIM_FECHA

En esta transformación cargaréis la dimensión «Fecha». 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 utilizáis una dimensión temporal sencilla, utilizaréis un *script* SQL para generar todos los registros necesarios.

Será necesario indicar manualmente una fecha de inicio, la fecha de fin será el día que se ejecute el *script* SQL.



El *script* SQL completo es el siguiente:

```

-- Declaración y establecimiento de fecha inicio y fecha fin
(fecha actual)

DECLARE @FechaInicio datetime
DECLARE @FechaFin datetime
SET @FechaInicio = '01/01/2014'
SET @FechaFin = GETDATE()

-- Declaración y establecimiento de fecha ciclo

DECLARE @FechaCiclo datetime
SET @FechaCiclo = @FechaInicio

-- Bucle hasta fecha fin

WHILE @FechaCiclo <= @FechaFin

```

BEGIN

-- Insertar un registro en la dimensión Fecha

```

INSERT INTO DIM_Fecha VALUES (
    cast(cast(Year(@FechaCiclo)
    varchar(4))+right('0'+cast(Month(@FechaCiclo)
    varchar(2)),2)+right('0'+cast(Day(@FechaCiclo)
    as int),
    Year(@FechaCiclo),
    Month(@FechaCiclo),
    Day(@FechaCiclo),
    @FechaCiclo
    )
    as varchar(2)),2)
    
```

-- Incrementar la FechaCiclo en un día

SET @FechaCiclo = DateAdd(d, 1, @FechaCiclo)

END

Transformación TR_DIM_GRUPO_EDAD

En esta transformación cargaréis los datos manualmente con un *Data Grid*; el origen de datos no será una tabla del *stage*.

Los grupos de edad serán fijos y, dado que se trata de pocos registros, los introduciréis manualmente.



Data Grid (Grupos de edad)

Data grid

Nombre paso
Data Grid (Grupos de edad)

Meta
Data

#	nombre	intervalo
1	Grupo edad 14-24	14 - 24
2	Grupo edad 25-34	25 - 34
3	Grupo edad 35-44	35 - 44
4	Grupo edad 45-54	45 - 54
5	Grupo edad 55-64	55 - 64
6	Grupo edad > 64	> 64
7	NI	NO IDENTIFICADO
8	NA	NO DISPONIBLE

? Help

Vale

Previsualizar

Cancelar

Data grid

Nombre paso: Data Grid (Grupos de edad)

Meta Data

#	Name	Type	Format	Length
1	nombre	String		20
2	intervalo	String		20

Help Vale Previsualizar Cancelar

Crearéis una secuencia que hará las funciones de clave primaria de incremento automático.



Obtener valor de la secuencia de la base de datos

Nombre de paso: Añadir secuencia

Nombre de valor: pk_grupo_edad

Utilizar una base de datos para generar la secuencia

¿Utilizar base datos para ☐

Conexión: [dropdown] [Editar...] [Nuevo...] [Wizard...]

Nombre de esquema: [text] [Schemas...]

Nombre de secuencia: SEQ_ [Sequences...]

Utilizar un contador de la transformación para generar la secuencia

¿Utilizar contador para ☒

Nombre contador (opcional): [text]

Valor inicial: 1

Incremento: 1

Valor máximo: 999999999

Help Vale Cancelar

Finalmente, cargaréis los datos en la tabla de dimensión.




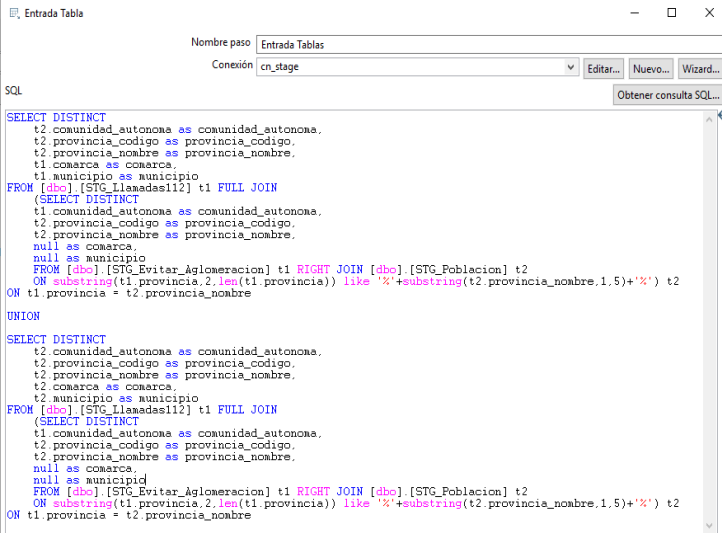

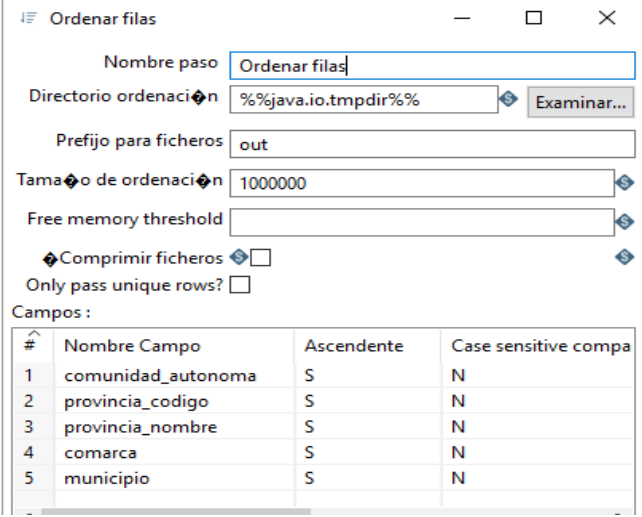

La transformación completa es la siguiente:

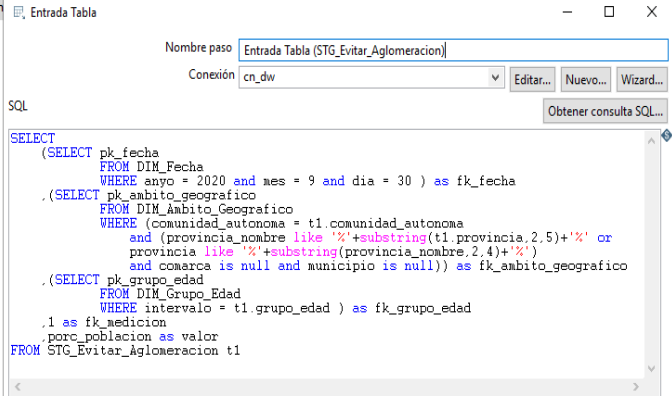

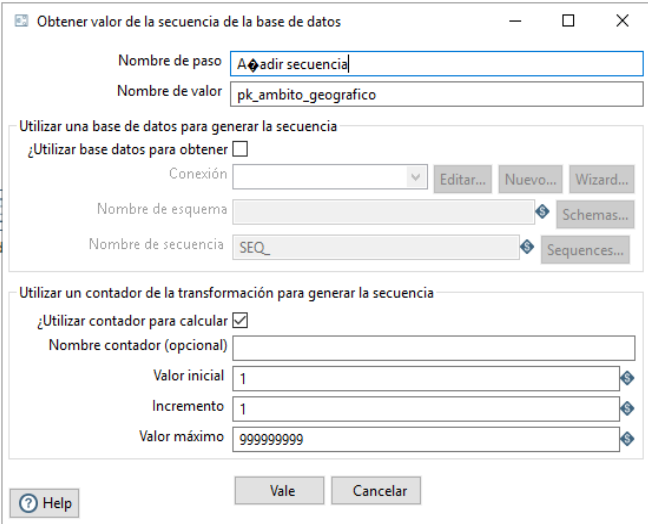


Transformación TR_DIM_AMBITO_GEOGRAFICO

Esta transformación consta de cuatro pasos: la entrada de tabla, la organización de las filas, la agregación de la secuencia y la salida de tabla.

 <p>Entrada Tablas</p>	<p>Mediante una sentencia «SELECT» de SQL obtendréis los distintos valores geográficos.</p> <p>En la SQL se utilizarán tres tablas intermedias. Ante la discrepancia en las denominaciones prevalecen las de «STG_Poblacion», ya que su fuente de origen es el Instituto Nacional de Estadística (INE).</p>
-----------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	
 <p>Ordenar filas</p>	<p>Ordenaréis los registros con base en los valores de los campos disponibles.</p> 
 <p>Entrada Tabla (STG_Evitar_Aglomeracion)</p>	<p>Mediante una sentencia «SELECT» de SQL obtendréis la información referente a la población por ámbito geográfico.</p>

	
	<p>Crearéis una secuencia que hará las funciones de clave primaria de incremento automático.</p> 

Finalmente, cargaréis los datos en la tabla de dimensión.

Salida Tabla (DIM_Ambito_Geografico)

Nombre paso: Salida Tabla (DIM_Ambito_Geografico)

Conexión: cn_dw

Esquema destino: dbo

Tabla destino: DIM_Ambito_Geografico

Tamaño de transacción (commit): 1000

Vaciar tabla: ☐

Ignorar errores de inserción: ☐

Specify database fields: ☒

Main options: Database fields

Fields to insert:

#	Table field	Stream field
1	provincia_codigo	provincia_codigo
2	provincia_nombre	provincia_nombre
3	comunidad_autonoma	comunidad_autonoma
4	comarca	comarca
5	municipio	municipio

Buttons: Get fields, Enter field mapping, Help, Vale, Cancelar, SQL

La transformación completa es la siguiente:



Transformación TR_DIM_TIPOLOGIA

En esta transformación cargaréis los datos mediante una sentencia «SELECT» de SQL sobre una tabla intermedia.

Mediante una sentencia «SELECT» de SQL obtendremos los distintos valores de tipos de llamada.



Entrada Tabla

Nombre paso: Entrada Tabla

Conexión: cn_stage

SQL:

```

select distinct tipo as nombre
from [dbo].[STG_Llamadas112]
union
select 'NO IDENTIFICADO' as nombre
union
select 'NO DISPONIBLE' as nombre
  
```

Buttons: Obtener consulta SQL...

Crearéis una secuencia que hará las funciones de clave primaria de incremento automático.



Obtener valor de la secuencia de la base de datos

Nombre de paso: Añadir secuencia

Nombre de valor: pk_tipologia

Utilizar una base de datos para generar la secuencia

¿Utilizar base de datos para: ☐

Conexión: [dropdown] [Editar...] [Nuevo...] [Wizard...]

Nombre de esquema: [dropdown] [Schemas...]

Nombre de secuencia: SEQ_ [Sequences...]

Utilizar un contador de la transformación para generar la secuencia

¿Utilizar contador para: ☒

Nombre contador (opcional): [text box]

Valor inicial: 1

Incremento: 1

Valor máximo: 99999999

Finalmente, cargaréis los datos en la tabla de dimensión.



Salida de Tabla

Nombre paso: Salida Tabla (DIM_Tipologia)

Conexión: cn_dw [Editar...] [Nuevo...] [Wizard...]

Esquema destino: dbo [Examinar...]

Tabla destino: DIM_Tipologia [Examinar...]

Tamaño transacción (commit): 1000

Vaciar tabla: ☐

Ignorar errores de inserción: ☐

Specify database fields: ☒

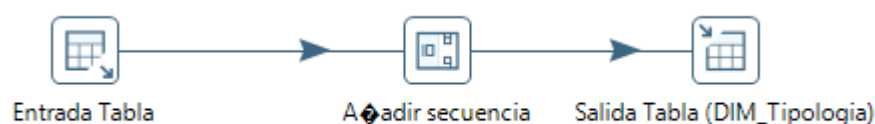
Main options / Database fields

#	Table field	Stream field
1	nombre	nombre
2	pk_tipologia	pk_tipologia

[Get fields] [Enter field mapping]


[Help] [Vale] [Cancelar] [SQL]

La transformación completa es la siguiente:




Transformación TR_DIM_MEDICION

En esta transformación cargaremos los datos manualmente con un *Data Grid*; el origen de datos no será una tabla del *stage*.



Data Grid (Unidades de medida)

Las unidades de medida serán fijas y, dado que se trata de pocos registros, los introduciréis manualmente.



Añadir secuencia

Crearéis una secuencia que hará las funciones de clave primaria de incremento automático.

Finalmente, cargaréis los datos en la tabla de dimensión.

Salida Tabla (DIM_Medicion)

Nombre paso: Salida Tabla (DIM_Medicion)

Conexión: cn_dw

Esquema destino: dbo

Tabla destino: DIM_Medicion

Tamaño de transacción (commit): 1000

Vaciar tabla: ☐

Ignorar errores de inserción: ☐

Specify database fields: ☒

Main options | Database fields

Fields to insert:

#	Table field	Stream field
1	nombre	nombre
2	unidad_medida	unidad_medida
3	pk_medicion	pk_medicion

Buttons: Get fields, Enter field mapping, Help, Vale, Cancelar, SQL

La transformación completa es la siguiente:



3.3.6. Bloque TR_FACT

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

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

Transformación TR_FACT_MEDICIONES

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 esta transformación se integrarán las tablas de datos sobre denuncias, población, movilidad y aglomeraciones 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.



Entrada Tabla (STG_Denuncias_Infracciones)

Mediante cuatro sentencias «SELECT» de SQL unidas con «UNION» obtendréis los distintos valores referentes a personas identificadas o detenidas, denuncias interpuestas y vehículos interceptados.



La sentencia SQL completa es la siguiente:

```

SELECT
    (SELECT pk_fecha
      FROM DIM_Fecha
      WHERE año = year(t1.fecha_final) and mes = month(t1.fecha_final) and dia =
month(t1.fecha_final) and dia = day(t1.fecha_final) ) as fk_fecha
    , (SELECT pk_ambito_geografico
      FROM DIM_Ambito_Geografico
      WHERE (provincia_nombre like '%' + substring(t1.provincia,2,5) + '%' ) and comarca is null and municipio
is null) as fk_ambito_geografico
    ,8 as fk_grupo_edad
    ,4 as fk_medicion
    ,identificados_ertzaintza + identificados_ppll as valor
FROM STG_Denuncias_Infracciones t1
UNION
SELECT
    (SELECT pk_fecha
      FROM DIM_Fecha
      WHERE año = year(t1.fecha_final) and mes = month(t1.fecha_final) and dia =
month(t1.fecha_final) and dia = day(t1.fecha_final) ) as fk_fecha
    , (SELECT pk_ambito_geografico
      FROM DIM_Ambito_Geografico
      WHERE (provincia_nombre like

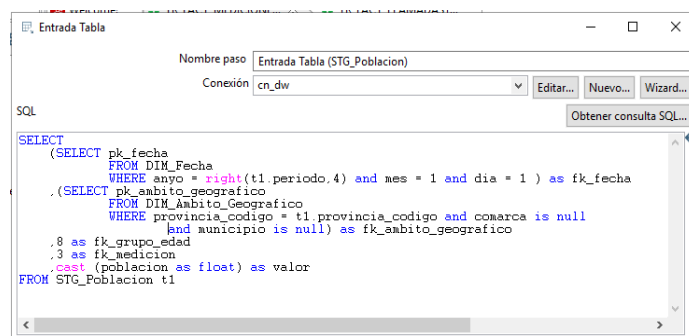
```

```
'%'+substring(t1.provincia,2,5)+'%') and comarca is null and municipio
is null) as fk_ambito_geografico
,8 as fk_grupo_edad
,5 as fk_medicion
,detenidos_ertzaintza + detenidos_pp11 as valor
FROM STG_Denuncias_Infracciones t1
UNION
SELECT
(SELECT pk_fecha
FROM DIM_Fecha
WHERE año = year(t1.fecha_final) and mes =
month(t1.fecha_final) and dia = day(t1.fecha_final) ) as fk_fecha
,(SELECT pk_ambito_geografico
FROM DIM_Ambito_Geografico
WHERE (provincia_nombre like
'%'+substring(t1.provincia,2,5)+'%') and comarca is null and municipio
is null) as fk_ambito_geografico
,8 as fk_grupo_edad
,6 as fk_medicion
,denuncias_ertzaintza + denuncias_pp11 as valor
FROM STG_Denuncias_Infracciones t1
UNION
SELECT
(SELECT pk_fecha
FROM DIM_Fecha
WHERE año = year(t1.fecha_final) and mes =
month(t1.fecha_final) and dia = day(t1.fecha_final) ) as fk_fecha
,(SELECT pk_ambito_geografico
FROM DIM_Ambito_Geografico
WHERE (provincia_nombre like
'%'+substring(t1.provincia,2,5)+'%') and comarca is null and municipio
is null) as fk_ambito_geografico
,8 as fk_grupo_edad
,7 as fk_medicion
,vehic_intercept_ertzaintza + vehic_intercept_pp11 as valor
FROM STG_Denuncias_Infracciones t1
```



Entrada Tabla (STG_Poblacion)

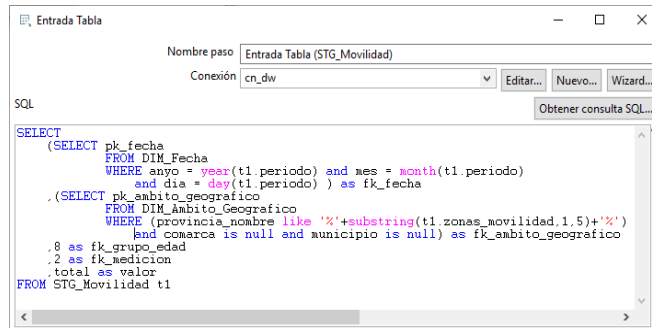
Mediante una sentencia «SELECT» de SQL obtendréis la información referente a la población por ámbito geográfico.





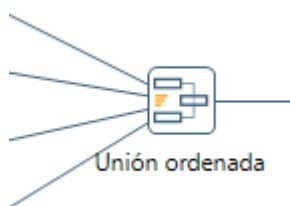
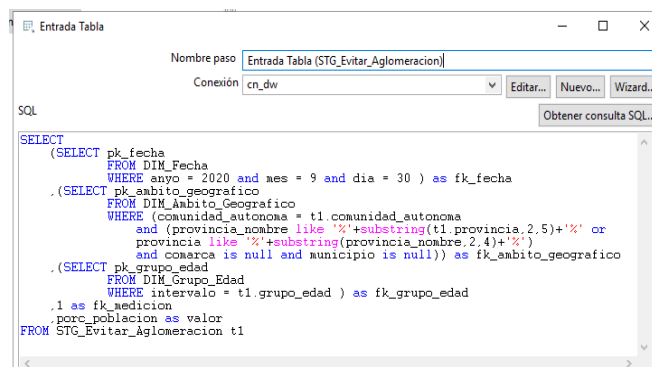
Entrada Tabla (STG_Movilidad)

Mediante una sentencia «SELECT» de SQL obtendréis la información referente a la movilidad por ámbito geográfico y fecha.



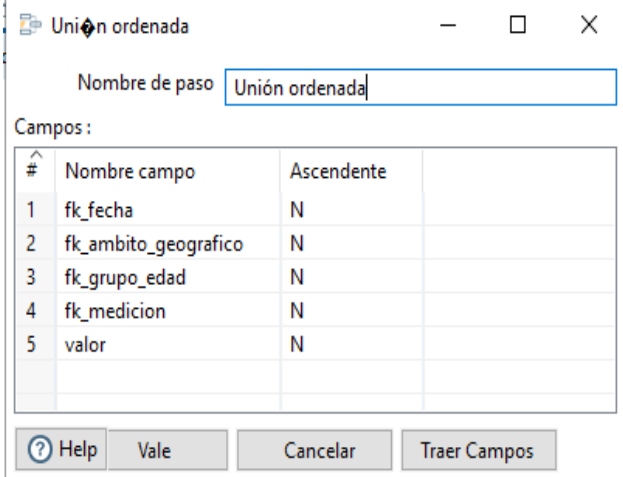

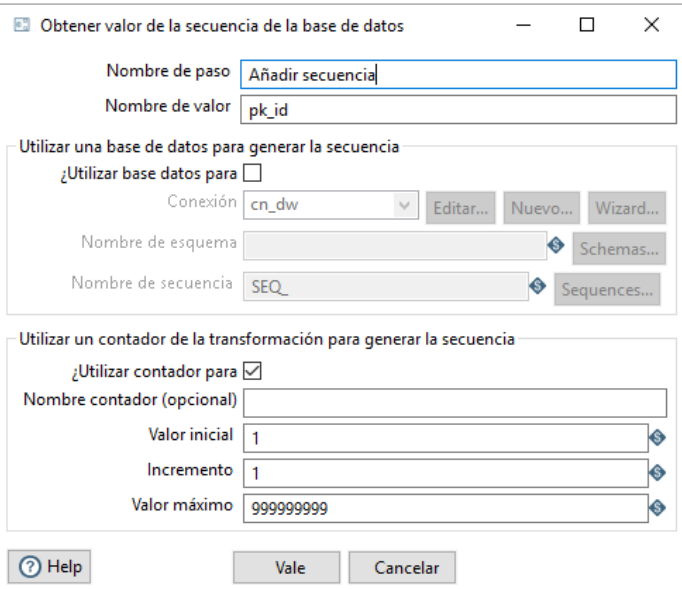

Entrada Tabla (STG_Evitar_Aglomeracion)


Mediante una sentencia «SELECT» de SQL obtendréis la información referente al porcentaje de la ciudadanía que ha evitado las aglomeraciones por ámbito geográfico.



A través de un paso de tipo «Unión ordenada» uniréis los resultados de las cuatro entradas de las tablas previas.

Para ello es importante que cada entrada disponga de los mismos campos en nombre, tipo y formato.

	 <p>Unión ordenada</p> <p>Nombre de paso Unión ordenada</p> <p>Campos :</p> <table><thead><tr><th>#</th><th>Nombre campo</th><th>Ascendente</th></tr></thead><tbody><tr><td>1</td><td>fk_fecha</td><td>N</td></tr><tr><td>2</td><td>fk_ambito_geografico</td><td>N</td></tr><tr><td>3</td><td>fk_grupo_edad</td><td>N</td></tr><tr><td>4</td><td>fk_medicion</td><td>N</td></tr><tr><td>5</td><td>valor</td><td>N</td></tr></tbody></table> <p>Buttons: Help, Vale, Cancelar, Traer Campos</p>	#	Nombre campo	Ascendente	1	fk_fecha	N	2	fk_ambito_geografico	N	3	fk_grupo_edad	N	4	fk_medicion	N	5	valor	N
#	Nombre campo	Ascendente																	
1	fk_fecha	N																	
2	fk_ambito_geografico	N																	
3	fk_grupo_edad	N																	
4	fk_medicion	N																	
5	valor	N																	
 <p>Añadir secuencia</p>	<p>Crearéis una secuencia que hará las funciones de clave primaria de incremento automático.</p>  <p>Obtener valor de la secuencia de la base de datos</p> <p>Nombre de paso Añadir secuencia</p> <p>Nombre de valor pk_id</p> <p>Utilizar una base de datos para generar la secuencia</p> <p>¿Utilizar base de datos para <input type="checkbox"/></p> <p>Conexión cn_dw (Buttons: Editar..., Nuevo..., Wizard...)</p> <p>Nombre de esquema (Button: Schemas...)</p> <p>Nombre de secuencia SEQ_ (Button: Sequences...)</p> <p>Utilizar un contador de la transformación para generar la secuencia</p> <p>¿Utilizar contador para <input checked="" type="checkbox"/></p> <p>Nombre contador (opcional)</p> <p>Valor inicial 1</p> <p>Incremento 1</p> <p>Valor máximo 99999999</p> <p>Buttons: Help, Vale, Cancelar</p>																		
 <p>Selecciona/Renombra valores</p>	<p>A través del paso «Selecciona / Renombra valores» redefiniréis el tipo, la longitud y la precisión de cada campo que queráis introducir posteriormente en la tabla de salida.</p>																		



Salida Tabla (FACT_Mediciones)

Selección/Renombra valores

Nombre paso: Selección/Renombra valores

Selección & Modifica | Eliminar | Meta-información

Campos a modificar meta información:

#	Nombre campo	Renombrar a	Tipo	Longitud	Precisión
1	fk_fecha		None	9	0
2	fk_ambito_geografico		None	9	0
3	fk_grupo_edad		None	9	0
4	fk_medicion		None	9	0
5	valor		Number	15	2
6	pk_id		None		0

Finalmente, cargaréis los datos en la tabla de hechos.

Salida de Tabla

Nombre paso: Salida Tabla (FACT_Mediciones)

Conexión: cn_dw [Editar...] [Nuevo...] [Wizard...]

Esquema destino: dbo [Examinar...]

Tabla destino: FACT_Mediciones [Examinar...]

Tamaño transacción (commit): 1000

Vaciar tabla ☒

Ignorar errores de inserción ☐

Specify database fields ☒

Main options | Database fields

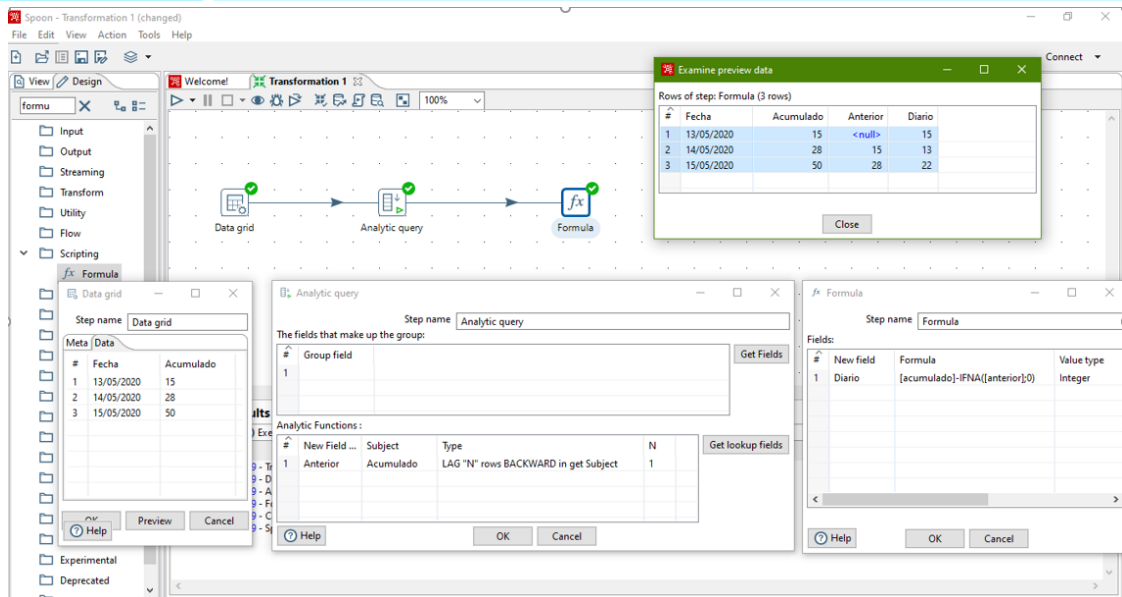
Fields to insert:

#	Table field	Stream field
1	fk_fecha	fk_fecha
2	fk_ambito_geografico	fk_ambito_geografico
3	fk_grupo_edad	fk_grupo_edad
4	fk_medicion	fk_medicion
5	valor	valor
6	pk_id	pk_id

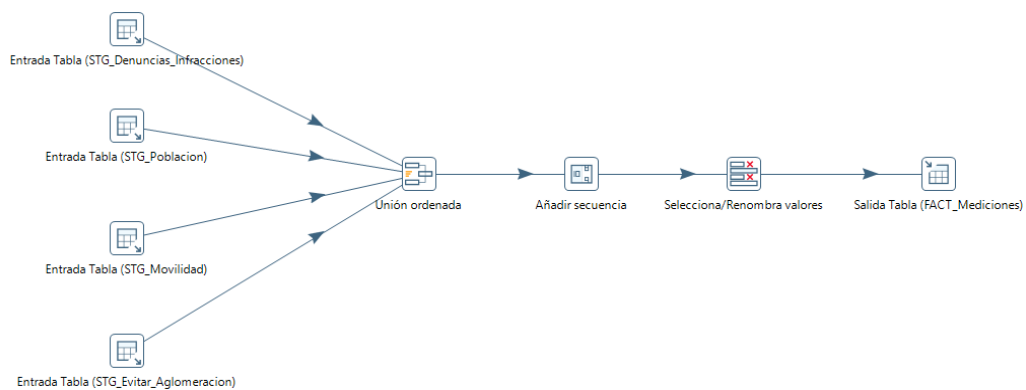
[Get fields] [Enter field mapping]

Para las mediciones 4, 5, 6 y 7 de «fk_medicion» de la «FACT_Mediciones», a partir de la lectura de la tabla «STG_Denuncias_Infracciones», se han tratado valores acumulados en lugar de valores diarios para una mejor comprensión de la secuencia de pasos, así como un tratamiento más simple y directo en esta primera tabla de hechos.

Para tratar los valores diarios, para cada medida tratada, una de las opciones sería añadir en *Spoon* dos pasos adicionales «Analytic query» y «Formula». El paso «Analytic query» obtiene justamente el valor anterior, mientras que el paso «Formula» realiza la operación para obtener el valor diario (valor acumulado-valor anterior). A continuación, se muestra un ejemplo para este tratamiento:



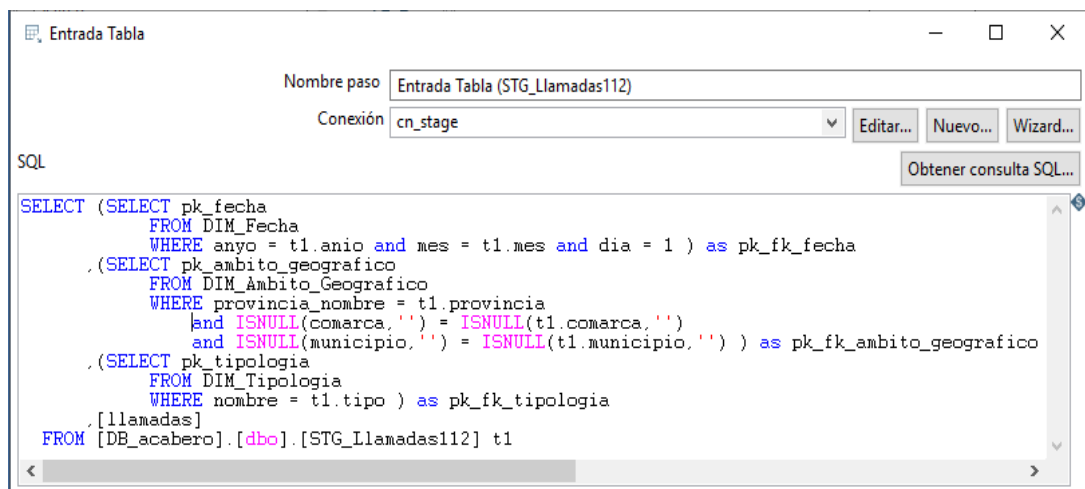
La transformación completa quedará de la siguiente manera:



Transformación TR_FACT_LLAMADAS112

La carga de la tabla de hechos «FACT_Llamadas112» es bastante más sencilla, ya que únicamente habrá que obtener los datos de una tabla intermedia.

A pesar de ello, también es necesario obtener los valores de las claves foráneas. A modo de ejemplo, se mostrará cómo obtener estos valores directamente desde la extracción en SQL. No obstante, este mecanismo únicamente es recomendable en cargas pequeñas y relativamente sencillas (con poca lógica y poco control de flujo en la transformación).



La transformación completa quedará de la siguiente manera:



4. Implementación de trabajos con procesos ETL

Habrá que 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.

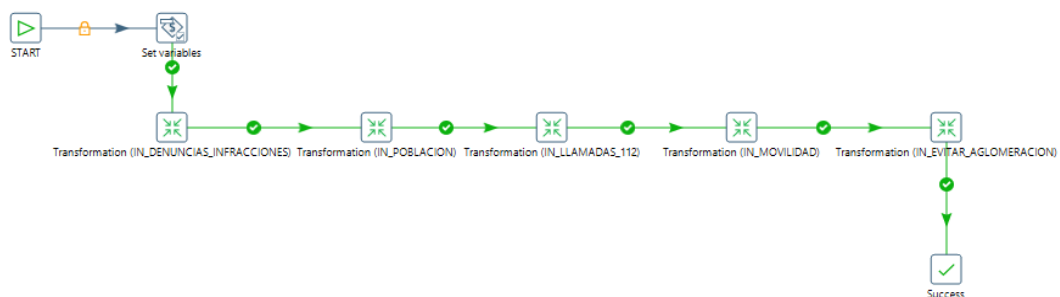
Vaiss a diseñar los trabajos (*jobs*) mediante PDI, estos van a permitir la ejecución secuencial de todos los procesos de ETL incluidos en cada bloque definido.

Cada trabajo contiene como pasos cada una de las transformaciones implementadas en el apartado anterior de diseño de ETL.

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.
- 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:

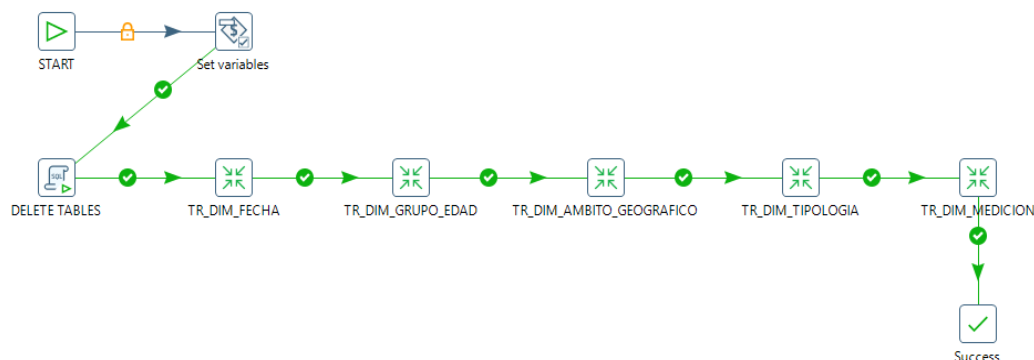
Execution Results			
Trabajo / Entrada de Trabajo	Comentario	Resultado	Razón
Trabajo: JOB_IN	Start of job execution		start
START	Start of job execution		start
START	Job execution finished	Exito	
Set variables	Start of job execution		Followed unconditional link
Set variables	Job execution finished	Exito	
Transformation (IN_DENUNCIAS_INFRACCIONES)	Start of job execution		Followed link after success
Transformation (IN_DENUNCIAS_INFRACCIONES)	Job execution finished	Exito	
Transformation (IN_POBLACION)	Start of job execution		Followed link after success
Transformation (IN_POBLACION)	Job execution finished	Exito	
Transformation (IN_LLAMADAS_112)	Start of job execution		Followed link after success
Transformation (IN_LLAMADAS_112)	Job execution finished	Exito	
Transformation (IN_MOVILIDAD)	Start of job execution		Followed link after success
Transformation (IN_MOVILIDAD)	Job execution finished	Exito	
Transformation (IN_EVITAR_AGLOMERACION)	Start of job execution		Followed link after success
Transformation (IN_EVITAR_AGLOMERACION)	Job execution finished	Exito	
Success	Start of job execution		Followed link after success
Success	Job execution finished	Exito	
Trabajo: JOB_IN	Job execution finished	Exito	finished

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

JOB_TR_DIMS

El trabajo (*job*) «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 (*job*) «JOB_TR_DIMS» es el siguiente:



Los pasos incluidos en el trabajo «JOB_TR_DIMS» 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. Aunque es importante respetar el orden de borrado, según las

relaciones definidas entre tablas, en este caso en particular, dado que no hay relaciones entre tablas de dimensión, no influirá el orden.

- 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:

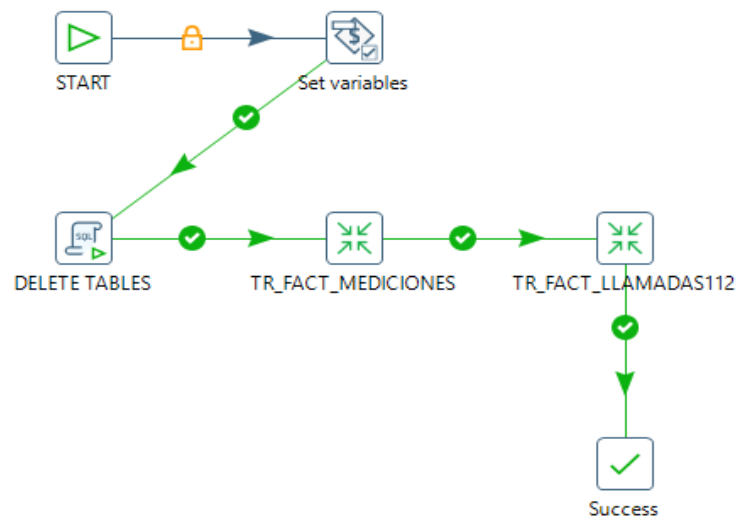
Execution Results			
Trabajo / Entrada de Trabajo	Comentario	Resultado	Razón
▼ JOB_TR_DIMS			
Trabajo: JOB_TR_DIMS	Start of job execution		start
START	Start of job execution		start
START	Job execution finished	Exito	
Set variables	Start of job execution		Followed unconditional link
Set variables	Job execution finished	Exito	
DELETE TABLES	Start of job execution		Followed link after success
DELETE TABLES	Job execution finished	Exito	
TR_DIM_FECHA	Start of job execution		Followed link after success
TR_DIM_FECHA	Job execution finished	Exito	
TR_DIM_GRUPO_EDAD	Start of job execution		Followed link after success
TR_DIM_GRUPO_EDAD	Job execution finished	Exito	
TR_DIM_AMBITO_GEOGRAFICO	Start of job execution		Followed link after success
TR_DIM_AMBITO_GEOGRAFICO	Job execution finished	Exito	
TR_DIM_TIPOLOGIA	Start of job execution		Followed link after success
TR_DIM_TIPOLOGIA	Job execution finished	Exito	
TR_DIM_MEDICION	Start of job execution		Followed link after success
TR_DIM_MEDICION	Job execution finished	Exito	
Success	Start of job execution		Followed link after success
Success	Job execution finished	Exito	
Trabajo: JOB_TR_DIMS	Job execution finished	Exito	finished

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

JOB_TR_FACTS

El trabajo (*job*) «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.

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



Los pasos incluidos en el trabajo «JOB_TR_FACTS» son:

- Inicio del *job*.
- Carga de variables de entorno.
- Eliminación 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

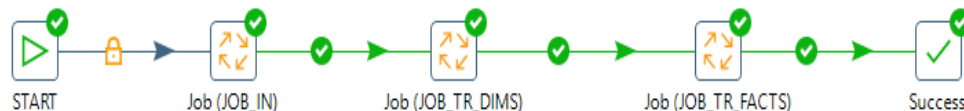
Trabajo / Entrada de Trabajo	Comentario	Resultado	Razón
Trabajo: JOB_TR_FACTS	Start of job execution		start
START	Start of job execution		start
START	Job execution finished	Exito	
Set variables	Start of job execution		Followed unconditional link
Set variables	Job execution finished	Exito	
DELETE TABLES	Start of job execution		Followed link after success
DELETE TABLES	Job execution finished	Exito	
TR_FACT_MEDICIONES	Start of job execution		Followed link after success
TR_FACT_MEDICIONES	Job execution finished	Exito	
TR_FACT_LLAMADAS112	Start of job execution		Followed link after success
TR_FACT_LLAMADAS112	Job execution finished	Exito	
Success	Start of job execution		Followed link after success
Success	Job execution finished	Exito	
Trabajo: JOB_TR_FACTS	Job execution finished	Exito	finished

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

JOB_CARGA_DW

El trabajo (*job*) «JOB_CARGA_DW» orquesta todos los trabajos anteriores en un único proceso.

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



Los pasos incluidos en el trabajo «JOB_CARGA_DW» son:

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

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

Execution Results			
Trabajo / Entrada de Trabajo	Comentario	Resultado	Razón
Trabajo: JOB_CARGA_DW	Start of job execution		start
START	Start of job execution		start
START	Job execution finished	Exito	
Job (JOB_IN)	Start of job execution		Followed unconditional link
Job (JOB_IN)	Job execution finished	Exito	
Job (JOB_TR_DIMS)	Start of job execution		Followed link after success
Job (JOB_TR_DIMS)	Job execution finished	Exito	
Job (JOB_TR_FACTS)	Start of job execution		Followed link after success
Job (JOB_TR_FACTS)	Job execution finished	Exito	
Success	Start of job execution		Followed link after success
Success	Job execution finished	Exito	
Trabajo: JOB_CARGA_DW	Job execution finished	Exito	finished

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

El tiempo total de la carga inicial del *data warehouse* es de aproximadamente un minuto y medio.

2020/12/11 14:57:13 - Spoon - Iniciando trabajo...
2020/12/11 14:58:42 - Spoon - Trabajo ha terminado.