



**Universitat Oberta
de Catalunya**

Máster universitario de Ciencia de Datos

Práctica 2

**Diseño y uso de bases de datos analíticas –
identificación, diseño y desarrollo de los procesos ETL.**

Autor:

Mario Ubierna San Mamés

Índice de Contenido

Índice de Contenido	3
Índice de tablas	4
Índice de ilustraciones	5
1. Introducción	6
2. Bibliografía	31

Índice de tablas

No se encuentran elementos de tabla de ilustraciones.

Índice de ilustraciones

No se encuentran elementos de tabla de ilustraciones.

1. Introducción

1.1. Presentación

A partir de la solución oficial de la primera práctica (PRA1), el estudiante debe diseñar, implementar y ejecutar los procesos de extracción, transformación y carga de los datos de las fuentes de datos proporcionadas.

Así pues, esta actividad tiene como objetivo identificar y desarrollar los procesos de carga del almacén de datos y que esta sea efectiva.

1.2. Descripción

Si nos centramos en los subobjetivos, esta segunda parte del caso práctico consiste en lo siguiente:

- Identificar los procesos de extracción, transformación y carga de datos (ETL) hacia el almacén de datos.
- Diseñar y desarrollar los procesos ETL mediante las herramientas de diseño proporcionadas.
- Implementar con los trabajos (*jobs*) los procesos ETL para que su carga planificada sea efectiva.

Además del documento con la solución de la PRA2 que se debe entregar, también se tendrá en consideración la implementación sobre la máquina virtual proporcionada en el curso.

En resumen, el documento de la solución de la PRA2 debe incluir los siguientes aspectos:

- Descripción de todas las acciones que se han realizado.

- Capturas de pantalla que muestren todas las partes significativas del ETL, sus características y su correspondiente explicación.
- Capturas de pantalla que demuestren la correcta ejecución de la ETL y el tiempo de ejecución.
- Capturas de pantalla que demuestren la correcta carga de los datos (cargados en la base de datos).

2. Identificación de los procesos ETL

A la hora de diseñar los procesos de carga de una base de datos analítica no hay una única estrategia. Es habitual estructurar los procesos ETL sobre la base de las entidades de datos que se deben actualizar, ya que existen diferencias conceptuales en la actualización de una dimensión con respecto a la de una tabla de hechos. La división del proceso de carga inicial en diferentes bloques de actualización facilitará el diseño de un orden de ejecución y la gestión de las dependencias. Cada uno de estos bloques de actualización se dividirá en las correspondientes etapas de extracción, transformación y carga.

Se identifican los dos bloques siguientes:

- **Bloque IN:** procesos de carga de los datos desde las fuentes a las 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 cargar los datos desde las tablas intermedias hasta nuestro almacén, según el modelo multidimensional diseñado. Así pues, son diferentes los procesos ETL de transformación para cargar las dimensiones de aquellos que se realizan para cargar las tablas de hechos. Estos procesos se distinguen con el prefijo «TR_» en el nombre.

2.1. Bloque IN

Respecto al bloque In, el cual nos va a permitir almacenar la información en el staging area, tenemos los siguientes procesos:

Nombre ETL	Descripción	Orígenes de los datos	Tabla de destino (stage)
------------	-------------	-----------------------	--------------------------

IN_ DENUNCIAS_ INFRACCIONE S	Carga de los datos correspondientes a las estadísticas sobre los expedientes incoados por el artículo 36.6 LOPSC de desobediencia durante el estado de emergencia sanitaria COVID-19 en la comunidad de Euskadi.	ACUMULADO- DENUNCIAS- INFRACCIONES.xlsx	STG_Denuncias _Infracciones
IN_POBLACIO N	Carga los datos respectivos a las cifras de la población española.	población_9687bsc .csv	STG_Poblacion
IN_MOVILIDA D	Movilidad de la población durante el estado de alarma.	35167bsc.csv	STG_Movilidad
IN_AGLOMER ACION	Porcentaje de la población que evitaba las aglomeraciones con motivo del coronavirus, por grupo de edad y provincia.	statistic_id1104235 _covid19_- poblacion-que- evitabalas- aglomeraciones- segunedad-en- espana-2020.xlsx	STG_Evitar_Aglo meracion
IN_LLAMADA S_112	Llamadas al 112 por ámbito geográfico y tipología (accidentes de tráfico, civismo, incendios, asistencia sanitaria, seguridad...)	rows.xml	SGT_Llamadas1 12

Tabla 1 - Procesos ETL Bloque IN.

2.2. Bloque TR

Respecto al bloque TR tenemos tanto los procesos para dotar de datos a las dimensiones como a los hechos.

2.2.1. Dimensiones

Los procesos ETL que se encargan de añadir la información a las dimensiones son los siguientes:

Nombre del ETL	Descripción	Tabla de origen	Tabla de destino (dimensión)
TR_DIM_FECHA	Carga y transformación de la dimensión temporal.		
TR_DIM_AMBITO_GEOGRAFICO	Carga y transformación de la dimensión con los datos de los ámbitos geográficos.		
TR_DIM_GRUPO_EDAD	Carga y transformación de la dimensión con los datos de los grupo de edad.		
TR_DIM_MEDICION	Carga y transformación de la dimensión con los datos de las mediciones.		
TR_DIM_TIPOLOGIA	Carga y transformación de la dimensión con los datos de la tipología.		

Tabla 2 - Procesos ETL Bloque TR Dimensiones.

2.2.2. Hechos

Respecto a los hechos tenemos los siguientes procesos de carga:

Nombre del ETL	Descripción	Tabla de origen
TR_FACT_LLAMADAS112	Carga y transformación de la tabla de hechos Fact_Llamadas112.	STG_Llamadas112
TR_FACT_MEDICIONES	Carga y transformación de la tabla de hechos Fact_Mediciones	

Tabla 3 - Procesos ETL Bloque TR Hechos.

3. Diseño y desarrollo de los procesos ETL

En este apartado, se deben diseñar los procesos de carga identificados en el punto anterior con la herramienta de diseño proporcionada. En este caso es Pentho Data Integration (PDI).

3.1. Creación de tablas

El primer paso para la implementación de los procesos ETL consiste en la creación de las tablas. Esto se llevará a cabo una única vez, mediante *scripts*, sobre la base de datos proporcionada (en nuestro caso: SQL Server). Se deberán crear las tablas intermedias y las tablas del modelo dimensional de la solución oficial, es decir, las dimensiones y las tablas de hechos. Para hacerlo, deben utilizarse los *scripts* facilitados junto a la solución de la PRA1.

3.1.1. Tablas del área intermedia (*staging area*)

Lo primero que vamos a hacer es la creación de las tablas intermedias:

Tabla intermedia STG_Denuncias_Infracciones

```

USE [DB_marioum]
GO

/***** Object: Table [dbo].[STG_Denuncias_Infracciones]
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
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_ppll] [float] NULL,
    [detenidos_ppll] [float] NULL,
    [denuncias_ppll] [float] NULL,
    [vehic_intercept_ppll] [float] NULL,
    [fecha] [datetime] NULL
) ON [PRIMARY]
GO

```

Ilustración 1 - STG_Denuncias_Infracciones.

Tabla intermedia STG_Poblacion

```

USE [DB_marioum]
GO

/***** Object: Table [dbo].[STG_Denuncias_Infracciones]
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
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

```

Ilustración 2 - STG_Poblacion.

Tabla intermedia STG_Llamadas112

```

USE [DB_marioum]
GO

/***** Object: Table [dbo].[STG_Denuncias_Infracciones]
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[STG_Llamadas112](
    [anio] [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]

```

*Ilustración 3 - STG_Llamadas112.***Tabla intermedia STG_Movilidad**

```

USE [DB_marioum]
GO

/***** Object: Table [dbo].[STG_Denuncias_Infracciones]
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

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

```

Ilustración 4 - STG_Movilidad.

Tabla intermedia STG_Evitar_Aglomeracion

```

USE [DB_marioum]
GO

/***** Object: Table [dbo].[STG_Denuncias_Infracciones]
SET ANSI_NULLS ON
GO

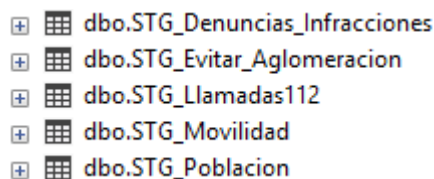
SET QUOTED_IDENTIFIER ON
GO

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

```

Ilustración 5 - STG_Evitar_Aglomeracion.

Comprobamos que todas las tabla intermedias se han creado correctamente:



```

+  dbo.STG_Denuncias_Infracciones
+  dbo.STG_Evitar_Aglomeracion
+  dbo.STG_Llamadas112
+  dbo.STG_Movilidad
+  dbo.STG_Poblacion

```

*Ilustración 6 - Tablas de staging area.***3.1.2. Tablas de las dimensiones**

Lo segundo que debemos de hacer es la creación de las tablas de dimensiones:

Tabla dimensión DIM_Ambito_Geografico

```

USE [DB_marioum]
GO

/***** Object: Table [dbo].[STG_Denuncias_Infracciones] *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

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

```

Ilustración 7 - DIM_Ambito_Geografico.

Tabla dimensión DIM_Fecha

```
USE [DB_marioum]
GO

/***** Object: Table [dbo].[STG_Denuncias_Infracciones] *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[DIM_Fecha](
    [pk_fecha] [int] NOT NULL,
    [anyo] [int] NOT NULL,
    [mes] [int] NOT NULL,
    [dia] [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
```

Ilustración 8 - DIM_Fecha.

Tabla dimensión DIM_Grupo_Edad

```
USE [DB_marioum]
GO

/***** Object: Table [dbo].[STG_Denuncias_Infracciones] *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

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
```

Ilustración 9 - DIM_Grupo_Edad.

Tabla dimensión DIM_Medicion

```
USE [DB_marioum]
GO

/***** Object: Table [dbo].[STG_Denuncias_Infracciones] *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[DIM_Medicion](
    [pk_medicion] [int] NOT NULL,
    [nombre] [varchar](100) NOT NULL,
    [unidad_medida] [varchar](20) NOT 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
```

Ilustración 10 - DIM_Medicion.

Tabla dimensión DIM_Tipologia

```
USE [DB_marioum]
GO

/***** Object: Table [dbo].[STG_Denuncias_Infracciones] *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[DIM_Tipologia](
    [pk_tipologia] [int] NOT NULL,
    [nombre] [varchar](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
```

Ilustración 11 - DIM_Tipologia.

Comprobamos que todas las tablas de dimensiones se han creado correctamente:

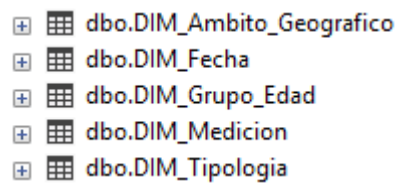


Ilustración 12 - Tablas de dimensiones.

3.1.3. Tablas de hechos

Finalmente creamos las diferentes tablas de los hechos:

Tabla hecho FACT_Llamadas112

```
USE [DB_marioum]
GO

/***** Object: Table [dbo].[STG_Denuncias_Infracciones] *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[FACT_Llamadas112](
    [pk_fk_fecha] [int] NOT NULL,
    [pk_fk_ambito_geografico] [int] NOT NULL,
    [pk_fk_tipologia] [int] NOT NULL,
    [llamadas] [int] NULL,
    CONSTRAINT [PK_FACT_Llamadas112] 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
```

Ilustración 13 - FACT_Llamadas112.

Tabla hecho FACT_Mediciones

```
USE [DB_marioum]
GO

/***** Object: Table [dbo].[STG_Denuncias_Infracciones] *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

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_FACT_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
```

Ilustración 14 - FACT_Mediciones.

Realizamos los alter table de las tablas de hechos:

```
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
```

Ilustración 15 - Alter table hechos.

Comprobamos que se han creado todas las tablas correspondientes:

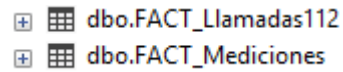


Ilustración 16 - Tablas de hechos.

3.2. Bloque IN

En este bloque se van a realizar las transformaciones para que la información en forma bruta se pase a las tablas intermedias, y luego haremos uso de éstas para crear las transformaciones de dimensiones y hechos.

3.2.1. Definición de variables de entorno

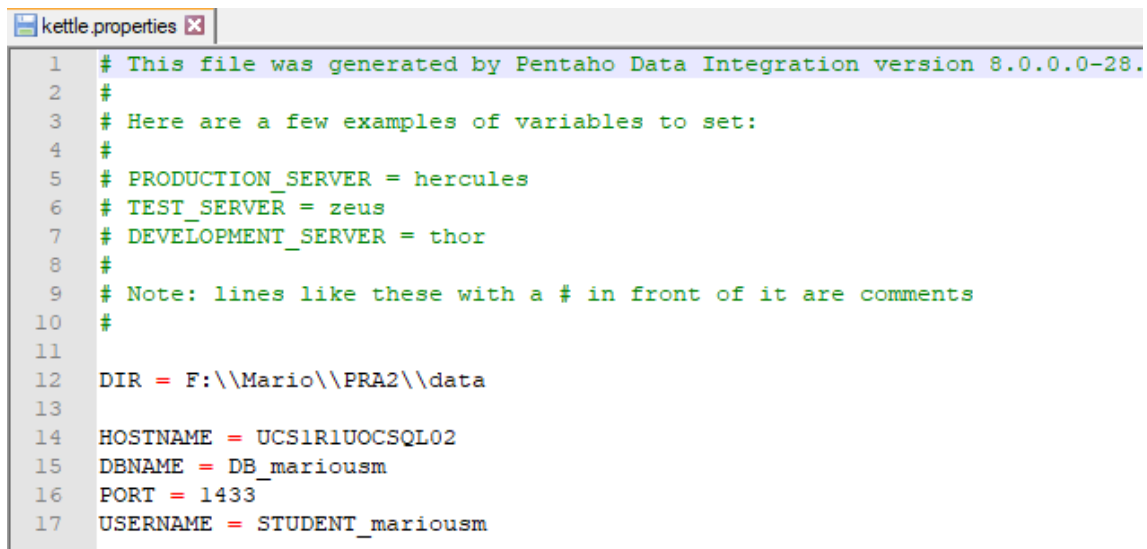
Es una buena práctica utilizar variables de entorno para así poder evitar errores en el definiciones futuras. Para ello accedemos a *kettle.properties* y definimos las siguientes variables:

Para el origen en el que se encuentran todos los archivos definimos la variables DIR_ENT:

- Nombre: DIR
- Valor: F:\Mario\PRA2\data

Para la cadena de conexión a la base de datos vamos a usar:

- Nombre: HOSTNAME
- Valor: UCS1R1UOCSQL02
- Nombre: DBNAME
- Valor: DB_mariouism
- Nombre: PORT
- Valor: 1433
- Nombre: USERNAME
- Valor: STUDENT_mariouism



```
kettle.properties
1 # This file was generated by Pentaho Data Integration version 8.0.0.0-28.
2 #
3 # Here are a few examples of variables to set:
4 #
5 # PRODUCTION_SERVER = hercules
6 # TEST_SERVER = zeus
7 # DEVELOPMENT_SERVER = thor
8 #
9 # Note: lines like these with a # in front of it are comments
10 #
11
12 DIR = F:\\Mario\\PRA2\\data
13
14 HOSTNAME = UCS1R1UOCSQL02
15 DBNAME = DB_mariousm
16 PORT = 1433
17 USERNAME = STUDENT_mariousm
```

Ilustración 17 - Variables de entorno.

3.2.2. Conexión base de datos SQL Server

El siguiente paso es crear la conexión a la base de datos que va a ser usada tanto por las transformaciones como por los jobs que se realicen en esta práctica.

Para ello creamos la nueva conexión y establecemos los valores definidos en las variables de entorno:

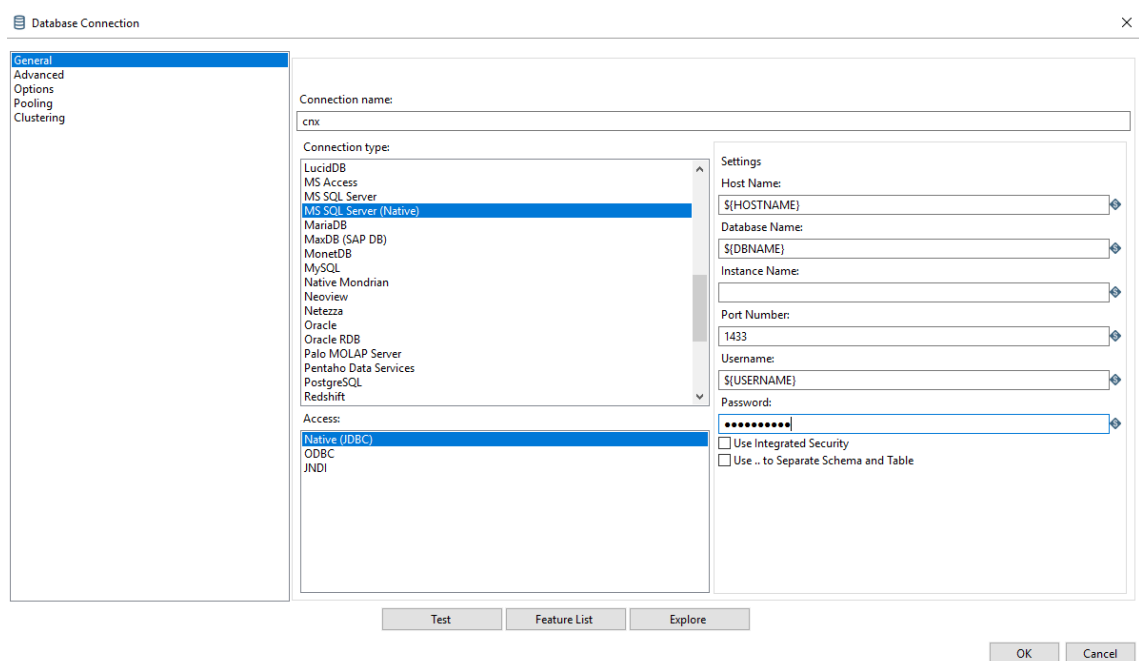


Ilustración 18 - Conexión a la base de datos.

3.2.3. Transformación IN_DENUNCIAS_INFRACCIONES

Una vez que ya hemos definido las variables de entorno y la conexión podemos proceder a realizar todas las transformaciones y trabajos.

La primera transformación que vamos a realizar se llama “IN_DENUNCIAS_INFRACCIONES”, su objetivo es leer todos los datos del archivo “ACUMULADO-DENUNCIAS-INFRACCIONES.xlsx” en la tabla intermedia “STG_Denuncias_Infracciones”.

En este caso no hemos hecho ninguna modificación en el Excel original, por lo que la transformación nos queda de la siguiente forma:

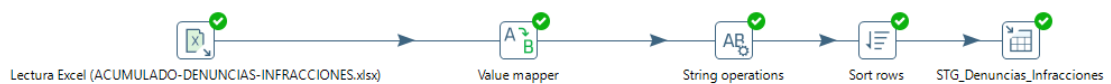


Ilustración 19 - IN_DENUNCIAS_INFRACCIONES.

Ahora vamos a explicar paso a paso lo que hemos hecho:

Lectura del Excel

Lo primero de todo es leer el fichero Excel que se nos proporciona, y para ello usamos el componente “Microsoft Excel Input”, una vez hecho eso escribimos el nombre del paso, le indicamos el fichero que va a utilizar, y le indicamos que el formato del fichero Excel es la XLSX:

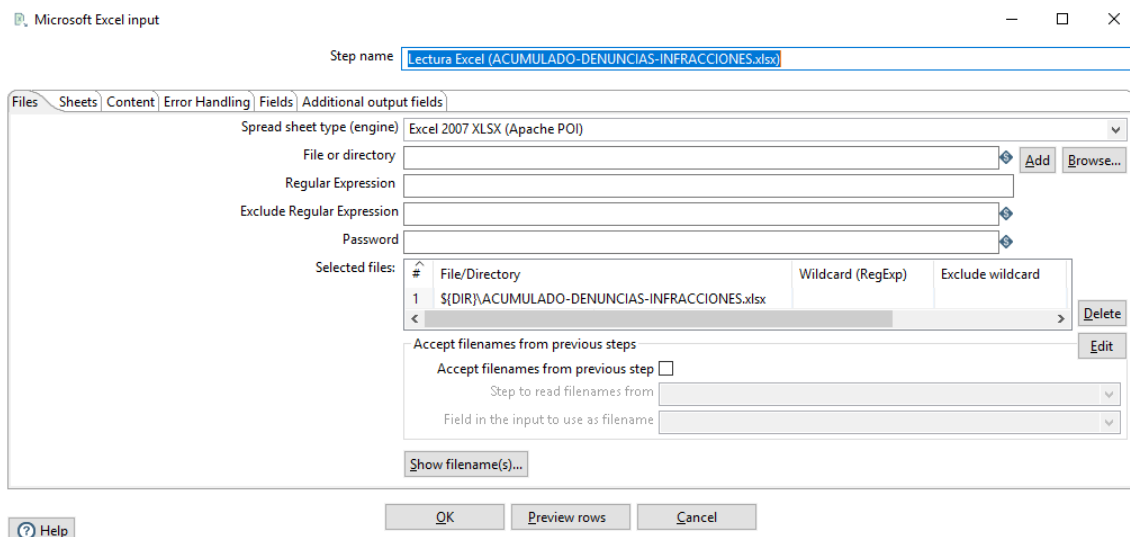


Ilustración 20 - Lectura IN_DENUNCIAS_INFRACCIONES.

Una vez hecho eso, le indicamos qué hoja tiene que leer y desde qué fila y columna, en nuestro caso la hoja “Datos_tratados” y la fila 5 columna 0:

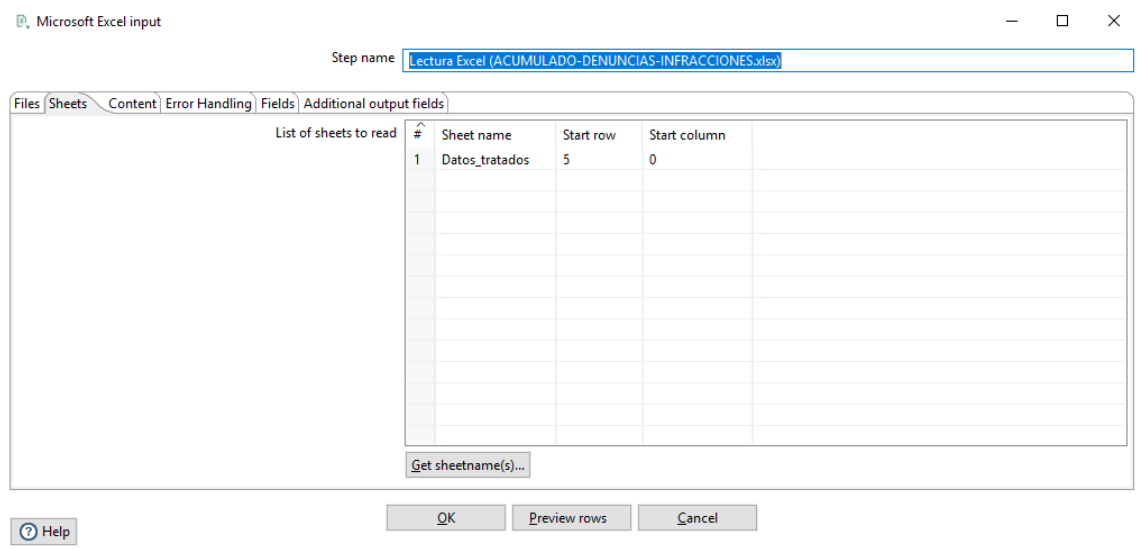


Ilustración 21 - Lectura IN_DENUNCIAS_INFRACCIONES.

Posteriormente obtenemos los campos leídos en la pestaña “Field”:

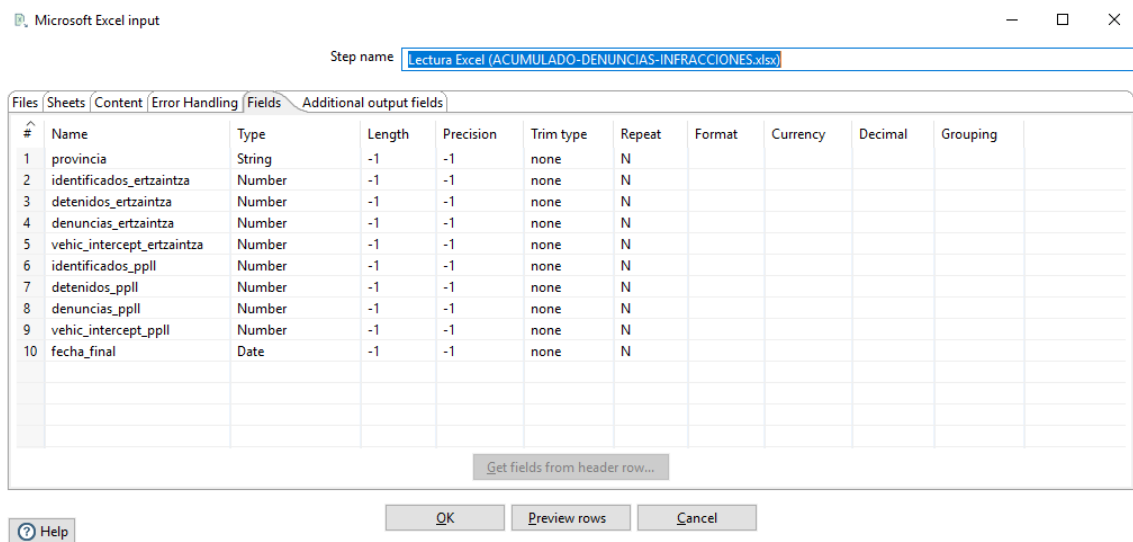


Ilustración 22 - Lectura IN_DENUNCIAS_INFRACCIONES.

Mapeo

Una vez leídos los datos vemos que las provincias están escritas en euskera, por lo que para homogeneizar los datos hemos decidido convertirlas al castellano. Por lo tanto, hacemos la traducción tal y como vemos en la siguiente captura:

[illegible]

Ilustración 23 - Mapeo Valores IN DENUNCIAS INFRACCIONES.

Normalización

Posteriormente hacemos una normalización de los campos que son de tipo “string”, ya que en éstos vamos a convertir los valores a mayúscula y sin espacios, tal y como vemos en la siguiente ilustración:

[illegible]

Ilustración 24 - Normalización Strings IN DENUNCIAS INFRACCIONES.

Ordenación

Posteriormente, ordenamos todos los campos de forma ascendente:

Sort rows

Step name: Sort rows

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

TMP-file prefix: out

Sort size (rows in memory): 1000000

Free memory threshold (in %):

Compress TMP Files? ☒

Only pass unique rows? (verifies keys only) ☐

Fields:

#	Fieldname	Ascending	Case sensitive compare?	Sort based on current locale?	Collator Strength	Presorted?
1	provincia	Y	N	N	0	N
2	identificados_ertzaintza	Y	N	N	0	N
3	detenidos_ertzaintza	Y	N	N	0	N
4	denuncias_ertzaintza	Y	N	N	0	N
5	vehic_intercept_ertzaintza	Y	N	N	0	N
6	identificados_ppil	Y	N	N	0	N
7	detenidos_ppil	Y	N	N	0	N
8	denuncias_ppil	Y	N	N	0	N
9	vehic_intercept_ppil	Y	N	N	0	N
10	fecha_final	Y	N	N	0	N

? Help OK Cancel Get Fields

Ilustración 25 - Ordenación IN_DENUNCIAS_INFRACCIONES.

Guardado

Finalmente, introducimos todos los valores en la base de datos, es decir, en la tabla intermedia STG_Denuncias_Infracciones, indicamos que haga un truncate de la tabla y con la conexión definida guardamos los valores en la tabla correspondiente:

Table output

Step name: **STG_Denuncias_Infracciones**

Connection: **cnx** [Edit...] [New...] [Wizard...]

Target schema: **dbo** [Browse...]

Target table: **STG_Denuncias_Infracciones** [Browse...]

Commit size: **1000**

Truncate table: ☒

Ignore insert errors: ☐

Specify database fields: ☒

Main options Database fields

Partition data over tables: ☐
Partitioning field: []

Partition data per month: ☒
Partition data per day: ☐

Use batch update for inserts: ☒

Is the name of the table defined in a field?: ☐
Field that contains name of table: []
Store the tablename field: ☒

Return auto-generated key: ☐
Name of auto-generated key field: []

[?] Help [OK] [Cancel] [SQL]

Ilustración 26 - Guardado IN_DENUNCIAS_INFRACCIONES.

Al ejecutar la anterior transformación obtenemos las siguiente métricas:

Execution Results

Logging [x] Execution History [x] Step Metrics [x] Performance Graph [x] Metrics [x] Preview data [x]

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)	input/output
1	Lectura Excel (ACUMULADO-DENUNCIAS-INFRACCIONES.xlsx)	0	0	219	219	0	0	0	0	Finished	0.2s	894	-
2	Value mapper	0	219	219	0	0	0	0	0	Finished	0.3s	826	-
3	String operations	0	219	219	0	0	0	0	0	Finished	0.3s	782	-
4	Sort rows	0	219	219	0	0	0	0	0	Finished	0.3s	702	-
5	STG_Denuncias_Infracciones	0	219	219	0	219	0	0	0	Finished	0.3s	637	-

Ilustración 27 - Métricas IN_DENUNCIAS_INFRACCIONES.

Observamos que tenemos 219 registros leídos y en nuestra base de datos se han almacenado también 219 registros, por lo que la información es correcta.

3.2.4. Transformación IN_POBLACION

La segunda transformación que vamos a realizar se llama “IN_POBLACION”, su objetivo es leer todos los datos del archivo “poblacion_9687bsc.csv” y almacenarlos en la tabla intermedia “STG_Poblacion”.

En este caso no hemos hecho ninguna modificación al fichero original, por lo que la transformación nos queda de la siguiente forma:



Ilustración 28 - IN_POBLACION.

Lectura CSV

Lo primero que debemos de hacer es cargar la información que se nos proporciona a partir del fichero CSV correspondiente. Por lo tanto, lo primero escribimos el nombre del paso, indicamos el fichero y el delimitador del CSV, en nuestro caso “;”:

CSV file input

Step name

Lectura CSV (poblacion_9687bsc.csv)

Filename

\${DIR}\poblacion_9687bsc.csv

Browse...

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		\$,	.	none
2	Provincias	String		25		\$,	.	none
3	Sexo	String		11		\$,	.	none
4	Periodo	String		18		\$,	.	none
5	Total	Integer	###,###,##	9	3	\$,	.	none

Ilustración 29 - Lectura IN_POBLACION.

Cabe destacar que hemos tenido que modificar el tipo del campo “Total” ya que lo reconocía como decimal cuando realmente es un entero.

Split

Luego separamos el campo “Provincias”, para así obtener tanto el código como la provincia correspondiente, para ello indicamos que el campo que queremos separar es “Provincias”, y luego en el grid establecemos los nuevos campos:

Step name:

Field to split:

Delimiter:

Enclosure:

Fields

#	New field	ID	Remove ID?	Type	Length	Precision	Format	Group	Decimal	Currency	Nullif	Default	Trim typ
1	provincia_codigo		N	String	2						99		both
2	provincia_nombre		N	String	100						NA		both

Ilustración 30 - Separación Campos IN_POBLACION.

Mapeo

Una vez hecho el paso anterior tenemos que mapear valores, esto se debe a que los nombres de las provincias no son del todo correctos (aparecen en gallego, euskera, catalán y valenciano). Además, al hacer la separación algunos nombres de provincias compuestas han desaparecido, es por ello que necesitamos de este paso para solventar los problemas:

Step name:

Fieldname to use:

Target field name (empty=overwrite):

Default upon non-matching:

Field values:

#	Source value	Target value
1	Alicante/Alacant	Alicante
2	Araba/Álava	Álava
3	Balears,	Baleares
4	Bizkaia	Vizcaya
5	Castellón/Castelló	Castellón
6	Coruña,	La Coruña
7	Gipuzkoa	Guipúzcoa
8	Girona	Gerona
9	Lleida	Lérida
10	Palmas,	Las Palmas
11	Rioja,	La Rioja
12	Santa	Santa Cruz de Tenerife
13	Valencia/València	Valencia
14	Ciudad	Ciudad Real
15	Ourense	Orense

Ilustración 31 - Mapeo Valores IN_POBLACION.

Normalización

Antes de almacenar los datos en la base de datos, vamos a normalizar los strings para que todos estén en mayúsculas y no tengan espacios ni al principio ni al final:

[illegible]

Ilustración 32 - Normalización Strings IN_POBLACION.

Guardado

Finalmente, guardamos los datos en la tabla “STG_Poblacion”, indicando que haga un truncate de la tabla y asociamos los campos:

Table output

Step name:

Connection: Edit... New... Wizard...

Target schema: Browse...

Target table: Browse...

Commit size:

Truncate table: ☒

Ignore insert errors: ☐

Specify database fields: ☒

Main options **Database fields**

Partition data over tables: ☐

Partitioning field:

Partition data per month: ☒

Partition data per day: ☐

Use batch update for inserts: ☒

Is the name of the table defined in a field? ☐

Field that contains name of table:

Store the tablename field: ☒

Return auto-generated key: ☐

Name of auto-generated key field:

Help OK Cancel SQL

Ilustración 33 - Guardado IN POBLACION.

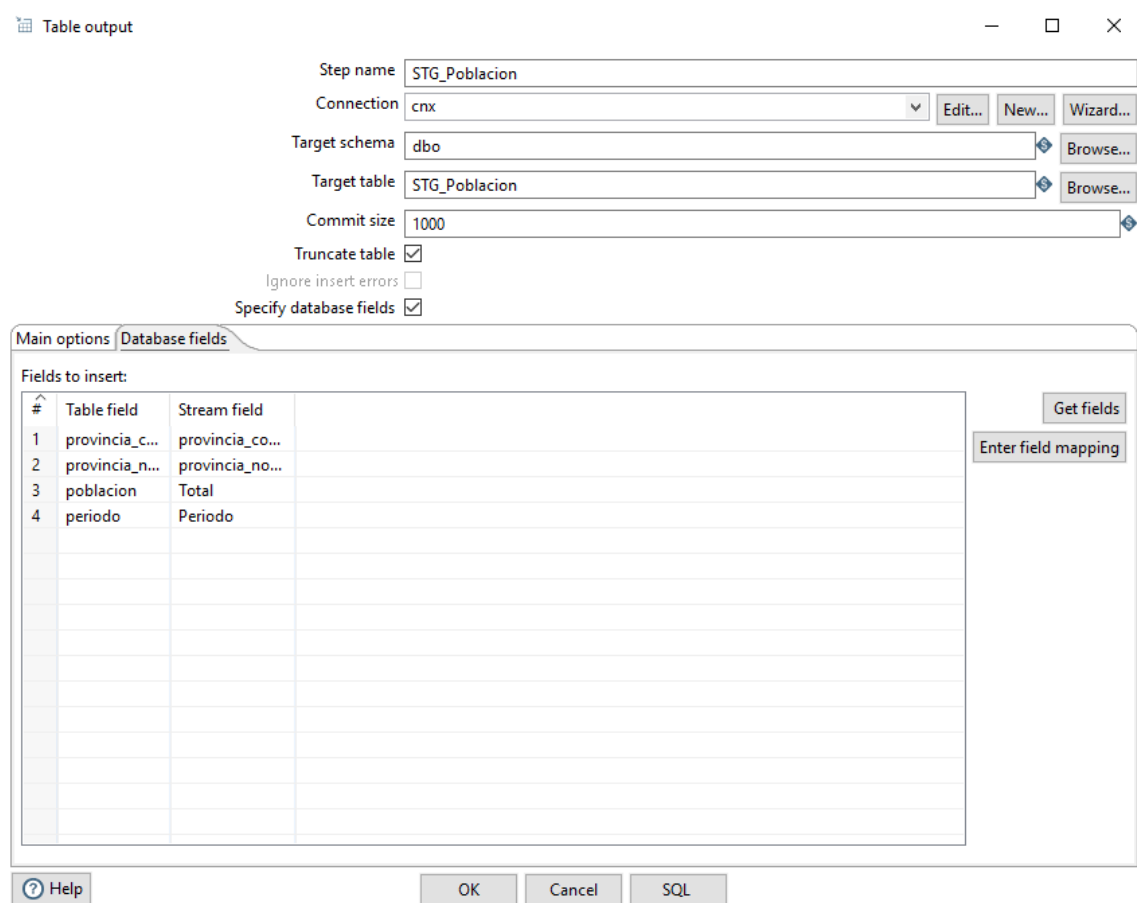


Ilustración 34 - Guardado IN_POBLACION.

Al ejecutar la anterior transformación obtenemos las siguiente métricas:

Execution Results

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)	input/output
1	Lectura CSV (poblacion_9687bsc.csv)	0	0	52	53	0	0	0	0	Finished	0.0s	1,325	-
2	Split fields	0	52	52	0	0	0	0	0	Finished	0.1s	426	-
3	Value mapper	0	52	52	0	0	0	0	0	Finished	0.1s	406	-
4	String operations	0	52	52	0	0	0	0	0	Finished	0.1s	397	-
5	STG_Poblacion	0	52	52	0	52	0	0	0	Finished	0.2s	292	-

Ilustración 35 - Métricas IN_POBLACION.

Observamos que tenemos 53 registros (52 registros + 1 cabecera) y se han almacenado 52 registros, por lo que la información es correcta.

3.2.5. Transformación IN_MOVILIDAD

La tercer transformación que vamos a realizar se llama “IN_MOVILIDAD”, su objetivo es leer todos los datos del archivo “35167bsc.csv” y guardarlos en la tabla intermedia “STG_Movilidad”.

En este caso no hemos hecho ninguna modificación en el fichero CSV original, por lo que la transformación nos queda de la siguiente manera:

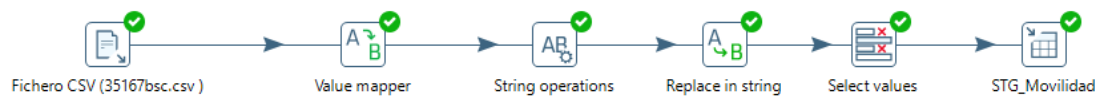


Ilustración 36 - IN_MOVILIDAD.

Ahora vamos a explicar paso a paso lo que hemos hecho:

Lectura CSV

Lo primero que tenemos que hacer es leer el fichero CSV que se nos proporciona, luego escribimos el nombre del paso, indicamos el fichero y el delimitador del CSV, en nuestro caso “,”:

CSV file input

Step name:

Filename:

Delimiter:

Enclosure:

NIO buffer size:

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	Decimal	Group	Trim type
1	Zonas de movilidad	String		27		\$,	.	both
2	Periodo	Timestamp	dd/MM/yyyy			\$,	.	both
3	Total	String	#,##	15	0	\$,	.	both

Ilustración 37 - Lectura IN_MOVILIDAD.

Cabe destacar que el atributo “Total” hemos indicado que sea de tipo string, ya que si considerábamos que fuera numérico a la hora de introducirlo en la base de datos no guardaba los decimales.

4. Bibliografía

a