SOLUCIÓN PRAC 2

DISEÑO Y USO DE BASES DE DATOS ANLÍTICAS



*Santiago Domínguez Collado.*

1. Identificación de los procesos ETL’s.

Para la identificación de los procesos ETL vamos a seguir las mismas directrices que en el ejemplo facilitado en los recursos de esta práctica.

Cabe destacar que, en este caso, no se dispondrá de una base de datos que haga el papel de staging area si no que las tablas irán todas a la misma base.

En nuestro caso identificamos los procesos ETL´s en dos bloques y utilizaremos un prefijo en el nombre para identificarlos:

* Bloque IN: procesos de carga de los datos desde las fuentes a tablas intermedias. 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 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.

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

|  |  |
| --- | --- |
| Nombre ETL | Descripción |
| IN\_COUNTRY | Carga de registros de la fuente COUNTRY.json a la tabla IN\_COUNTRY |
| IN\_MORTCAUSE | Carga de registros de la fuente REGION.json a la tabla IN\_REGION |
| IN\_REGION | Carga de registros de la fuente REGION.json a la tabla IN\_REGION |
| IN\_SARAMPION | Carga de los ficheros: Sarampion1.xml, Sarampion2.xml, Sarampion3.xml y Sarampion5.xml correspondientes a los casos, muertes, hospitalizaciones y pruebas del sarampión en los países de la OMS a la tabla IN\_SARAMPION |
| IN\_WUE\_DATA | Carga del fichero wuenic2018rev\_data\_2019-11-16.csv a la tabla IN\_WUE\_DATA. Contiene datos de cobertura de población infantil y objetivo. |
| IN\_COV\_SERIES | Carga del fichero con información cerca de la cobertura de vacunas en los países miembros de la OMS, coverage\_estimates\_series.xls a la tabla IN\_COV\_SERIES. |

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

|  |  |
| --- | --- |
| Nombre ETL | Descripción |
| TR\_DIM\_ANIO | Carga y transformación de la dimensión año (DIM\_ANIO). |
| TR\_DIM\_PAIS | Carga y transformación de la dimensión país (DIM\_PAIS). |
| TR\_DIM\_ GEOGRAFÍA | Carga y transformación de la dimensión geografía (DIM\_ GEOGRAFÍA). |
| TR\_DIM\_VACUNAS | Carga y transformación de la dimensión vacunas (DIM\_ VACUNAS). |
| TR\_DIM\_CAUSAS\_MORTALIDAD | Carga y transformación de la dimensión de causas de mortalidad (DIM\_ VACU DIM\_CAUSAS\_MORTALIDAD NAS). |

|  |  |
| --- | --- |
| Nombre ETL | Descripción |
| TR\_FACT\_COBERTURA | Carga y transformación de la tabla de hechos FACT\_COBERTURA. |
| TR\_FACT\_COBERTURA\_SARAMPION | Carga y transformación de la tabla de hechos FACT\_COBERTURA\_SARAMPION. |

1. Diseño de los procesos ETL’s.

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 ETL’s.

1. **Creación de tablas Intermedias.**

Será necesario crear previamente tablas en las que alojar la información de las fuentes.

**IN\_COUNTRY**

CREATE TABLE [dbo].[IN\_COUNTRY](

[ISO\_CODE] [varchar](50) NOT NULL,

[COUNTRY] [varchar](150) NOT NULL,

[CATEGORY] [varchar](300) NOT NULL,

[VALUE] [varchar](300) NOT NULL,

) ON [PRIMARY]

GO

**IN\_MORTCAUSE**

CREATE TABLE [dbo].[IN\_MORTCAUSE](

[ID] [numeric](5) NOT NULL,

[DISEASE] [varchar](300) NOT NULL

) ON [PRIMARY]

GO

**IN\_REGION**

CREATE TABLE [dbo].[IN\_REGION](

[CODE] [varchar](50) NOT NULL,

[REGION] [varchar](150) NOT NULL,

) ON [PRIMARY]

GO

**IN\_SARAMPION**

CREATE TABLE [dbo].[IN\_SARAMPION](

[COD] [numeric](10,0) NOT NULL,

[NOMBRE] [varchar](100) NOT NULL,

[ANIO] [numeric](15,0) NOT NULL,

[CASOS\_CONFIRMADOS] [numeric](38,0) NULL,

[MUERTES] [numeric](38,0) NULL,

[HOSPITALIZACIONES] [numeric](38,0) NULL,

[MUESTRAS\_LABORATORIO] [numeric](38,0) NULL,

) ON [PRIMARY]

GO

**IN\_WUE\_DATA**

CREATE TABLE [dbo].[IN\_WUE\_DATA](

[GROUP] [varchar](50) NOT NULL,

[SUBGROUP] [varchar](50) NOT NULL,

[NAME] [varchar](50) NOT NULL,

[YEAR] [numeric](15) NOT NULL,

[VACCINE] [varchar](50) NOT NULL,

[COVERAGE] [numeric](15, 0) NOT NULL,

[VACCINATED] [varchar](50) NOT NULL,

[TARGET] [varchar](50) NOT NULL,

[SOURCE] [varchar](200) NOT NULL,

) ON [PRIMARY]

GO

**IN\_COV\_SERIES**

CREATE TABLE [dbo].[IN\_COV\_SERIES](

[WHO\_REGION] [varchar](100) NOT NULL,

[ISO\_code] [varchar](3) NOT NULL,

[Cname] [varchar](30) NOT NULL,

[Continent] [varchar](30) NOT NULL,

[Vaccine] [varchar](300) NOT NULL,

[Year] [numeric](4, 0) NOT NULL,

[Percent\_covrage] [numeric](2, 0) NOT NULL,

[Asterisc] [varchar](1) NULL,

) ON [PRIMARY]

GO

1. **Creación del modelo multidimensional.**

En este punto veremos los scripts de creación del modelo físico multidimensional.

En la creación, además de atributos y métricas, se crearán también las restricciones definidas y que son propias del modelo multidimensional, las claves primarias de las dimensiones y las foráneas de las tablas de hechos.

1. Dimensiones

**DIM\_ANIO**

CREATE TABLE [dbo].[DIM\_ANIO](

[SK\_DIM\_ANIO] [numeric](4, 0) NOT NULL,

[DESC\_ANIO] [varchar](50) NOT NULL,

CONSTRAINT [PK\_DIM\_ANIO] PRIMARY KEY CLUSTERED

(

[SK\_DIM\_ANIO] 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\_PAIS**

CREATE TABLE [dbo].[DIM\_PAIS](

[SK\_DIM\_PAIS] [numeric](3, 0) NOT NULL,

[ISO\_CODE] [varchar](3) NOT NULL,

[NOMBRE\_PAIS] [varchar](30) NOT NULL,

[WORLD\_BANK\_INCOME] [numeric](3, 0) NOT NULL,

[WHO\_REGION\_CODE] [varchar](4) NOT NULL,

[WHO\_REGION ] [varchar](100) NOT NULL,

[ISO2\_CODE] [varchar](3) NOT NULL,

[SK\_DIM\_CAUSA] [numeric](2, 0) NOT NULL,

[LAND\_AREA] [varchar](15) NOT NULL,

[LANGUAGES\_EN\_2012] [varchar](300) NOT NULL,

CONSTRAINT [PK\_DIM\_PAIS] PRIMARY KEY CLUSTERED

(

[SK\_DIM\_PAIS] 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

ALTER TABLE [dbo].[DIM\_PAIS] WITH CHECK ADD CONSTRAINT

[FK\_DIM\_PAIS\_DIM\_CAUSA] FOREIGN KEY([SK\_DIM\_CAUSA])

REFERENCES [dbo].[DIM\_CAUSAS\_MORTALIDAD] ([SK\_DIM\_CAUSA])

GO

ALTER TABLE [dbo].[DIM\_PAIS] CHECK CONSTRAINT

[FK\_DIM\_PAIS\_DIM\_CAUSA]

GO

**DIM\_ GEOGRAFIA**

CREATE TABLE [dbo].[DIM\_GEOGRAFIA](

[SK\_DIM\_GEOGRAFIA] [numeric](3, 0) NOT NULL,

[COD\_REGION] [varchar](4) NOT NULL,

[DESC\_REGION] [varchar](30) NOT NULL,

[SK\_DIM\_PAIS] [numeric](3, 0) NOT NULL,

[DESC\_PAIS] [varchar](300) NOT NULL,

CONSTRAINT [PK\_DIM\_GEOGRAFIA] PRIMARY KEY CLUSTERED

(

[SK\_DIM\_GEOGRAFIA] 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

ALTER TABLE [dbo].[DIM\_GEOGRAFIA] WITH CHECK ADD CONSTRAINT

[FK\_DIM\_GEOGRAFIA\_DIM\_PAIS] FOREIGN KEY([SK\_DIM\_PAIS])

REFERENCES [dbo].[DIM\_PAIS] ([SK\_DIM\_PAIS])

GO

ALTER TABLE [dbo].[DIM\_GEOGRAFIA] CHECK CONSTRAINT

[FK\_DIM\_GEOGRAFIA\_DIM\_PAIS]

GO

**DIM\_VACUNA**

CREATE TABLE [dbo].[DIM\_VACUNA](

[SK\_DIM\_VACUNA] [numeric](3, 0) NOT NULL,

[TIPO\_VACUNA] [varchar](50) NOT NULL,

[NOMBRE\_VACUNA] [varchar](300) NOT NULL,

[GVAP] [varchar](1) NOT NULL,

CONSTRAINT [PK\_DIM\_VACUNA] PRIMARY KEY CLUSTERED

(

[SK\_DIM\_VACUNA] 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\_CAUSAS\_MORTALIDAD**

CREATE TABLE [dbo].[DIM\_CAUSAS\_MORTALIDAD](

[SK\_DIM\_CAUSA] [numeric](2, 0) NOT NULL,

[COD\_CAUSA] [varchar](4) NOT NULL,

[DESC\_CAUSA] [varchar](100) NOT NULL,

CONSTRAINT [PK\_DIM\_CAUSA] PRIMARY KEY CLUSTERED

(

[SK\_DIM\_CAUSA] 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

1. Tablas de hechos.

**FACT\_COBERTURA**

CREATE TABLE [dbo].[FACT\_COBERTURA](

[SK\_DIM\_ANIO] [numeric](4, 0) NOT NULL,

[SK\_DIM\_GEOGRAFIA] [numeric](3, 0) NOT NULL,

[SK\_DIM\_VACUNA] [numeric](3, 0) NOT NULL,

[COBERTURA] [numeric](2, 0) NOT NULL,

[NUM\_PV] [numeric](8, 0) NULL,

[OBJETIVO] [numeric](8, 0) NULL,

)

GO

ALTER TABLE [dbo].[FACT\_COBERTURA] WITH CHECK ADD CONSTRAINT

[FK\_ANIO] FOREIGN KEY([SK\_DIM\_ANIO])

REFERENCES [dbo].[DIM\_ANIO] ([SK\_DIM\_ANIO])

GO

ALTER TABLE [dbo].[FACT\_COBERTURA] CHECK CONSTRAINT [FK\_ANIO]

GO

ALTER TABLE [dbo].[FACT\_COBERTURA] WITH CHECK ADD CONSTRAINT

[FK\_GEOGRAFIA] FOREIGN KEY([SK\_DIM\_GEOGRAFIA])

REFERENCES [dbo].[DIM\_GEOGRAFIA] ([SK\_DIM\_GEOGRAFIA])

GO

ALTER TABLE [dbo].[FACT\_COBERTURA] CHECK CONSTRAINT [FK\_GEOGRAFIA]

GO

ALTER TABLE [dbo].[FACT\_COBERTURA] WITH CHECK ADD CONSTRAINT

[FK\_VACUNA] FOREIGN KEY([SK\_DIM\_VACUNA])

REFERENCES [dbo].[DIM\_VACUNA] ([SK\_DIM\_VACUNA])

GO

ALTER TABLE [dbo].[FACT\_COBERTURA] CHECK CONSTRAINT [FK\_VACUNA]

GO

**FACT\_COBERTURA\_SARAMPION**

CREATE TABLE [dbo].[FACT\_SARAMPION](

[SK\_DIM\_ANIO] [numeric](4, 0) NOT NULL,

[SK\_DIM\_GEOGRAFIA] [numeric](3, 0) NOT NULL,

[NUM\_CASOS] [numeric](8, 0) NULL,

[NUM\_MUERTES] [numeric](8, 0) NULL,

[NUM\_HOSPITALIZACIONES] [numeric](8, 0) NULL,

[NUM\_CASOS\_CONFIRMADOS] [numeric](8, 0) NULL,

)

GO

ALTER TABLE [dbo].[FACT\_SARAMPION] WITH CHECK ADD CONSTRAINT

[FK\_ANIO\_SARAMPION] FOREIGN KEY([SK\_DIM\_ANIO])

REFERENCES [dbo].[DIM\_ANIO] ([SK\_DIM\_ANIO])

GO

ALTER TABLE [dbo].[FACT\_SARAMPION] CHECK CONSTRAINT [FK\_ANIO\_SARAMPION]

GO

ALTER TABLE [dbo].[FACT\_SARAMPION] WITH CHECK ADD CONSTRAINT

[FK\_GEOGRAFIA\_SARAMPION] FOREIGN KEY([SK\_DIM\_GEOGRAFIA])

REFERENCES [dbo].[DIM\_GEOGRAFIA] ([SK\_DIM\_GEOGRAFIA])

GO

ALTER TABLE [dbo].[FACT\_SARAMPION] CHECK CONSTRAINT [FK\_GEOGRAFIA\_SARAMPION]

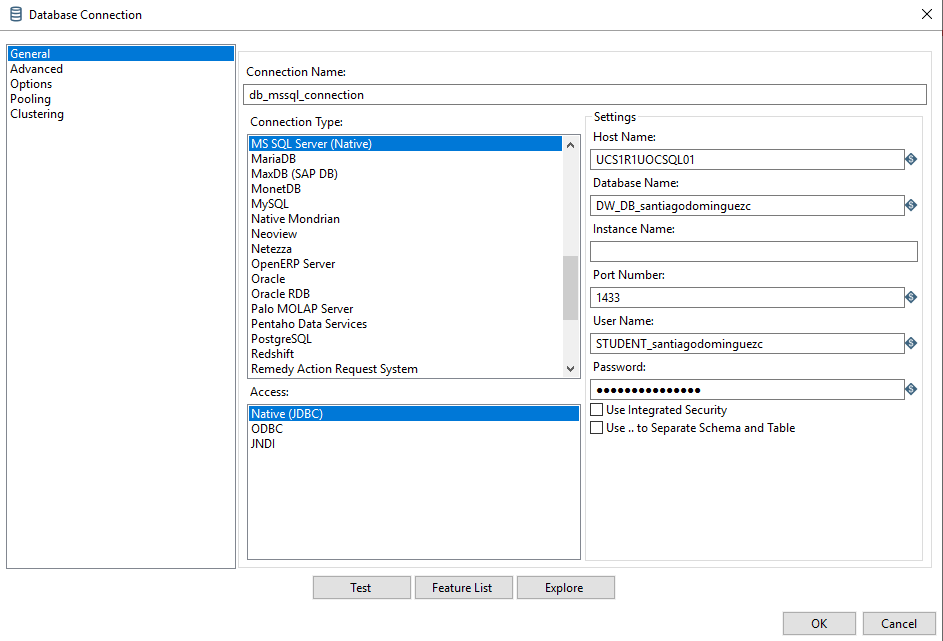
GO

1. Creación del proceso de extracción, transformación y carga (ETL).

Una vez que tenemos implementado el modelo físico del almacén, pasaremos a diseñar los procesos ETL que permitirán poblar las tablas intermedias, las tablas dimensiones y de hechos.

1. Conexión Base de Datos SQL Server

Durante todas las transacciones de la herramienta ETL será necesario configurar el acceso a la base de datos. Es por ello por lo que aquí mostramos el paso, para no repetirlo constantemente.



1. Bloque IN

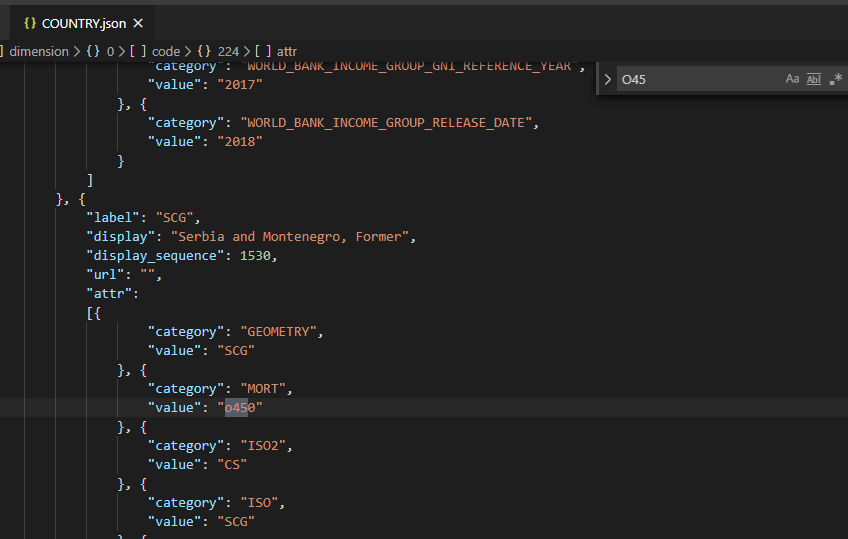
**Transformación IN\_COUNTRY**

El primer proceso por desarrollar es la carga de la fuente información sobre los países miembro de la OMS a la tabla intermedia IN\_COUNTRY. Partimos del fichero: COUNTRY.json

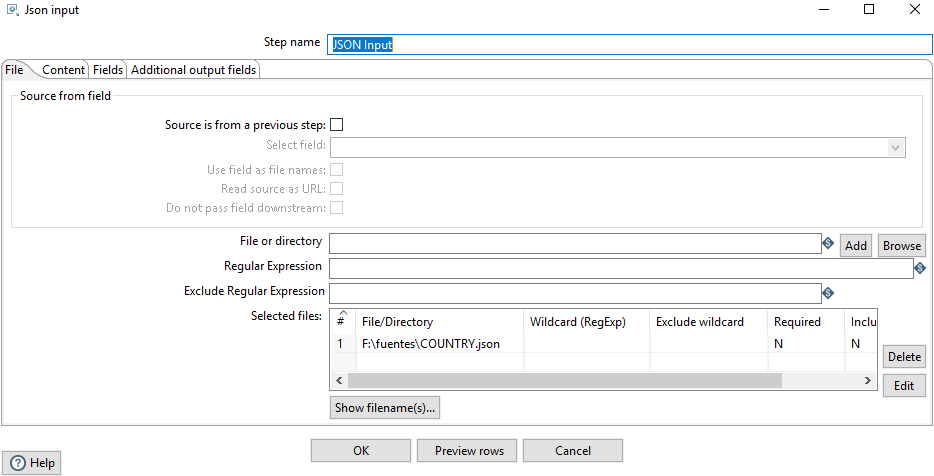
El proceso IN\_RAMA contiene cinco transformaciones: Dos lecturas de json, Selección de valores, Operación de cadena y Carga a la tabla intermedia IN\_ COUNTRY.

Previo a mostrar cada uno de los pasos en la transformación, durante el desarrollo se detecto una errata en uno de los valores a cargar, donde debería ir 0450, estaba escrito o450, lo que alteraba el tipo del atributo.

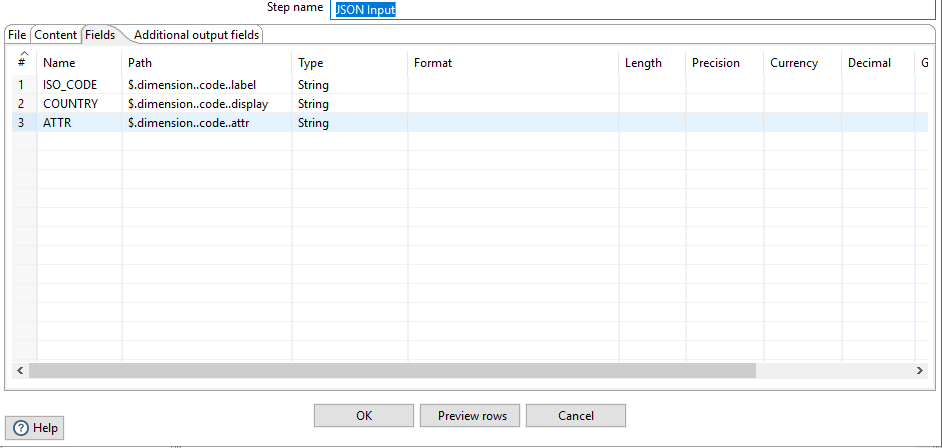
Dicho esto, lo reemplazamos haciendo uso del editor de texto.



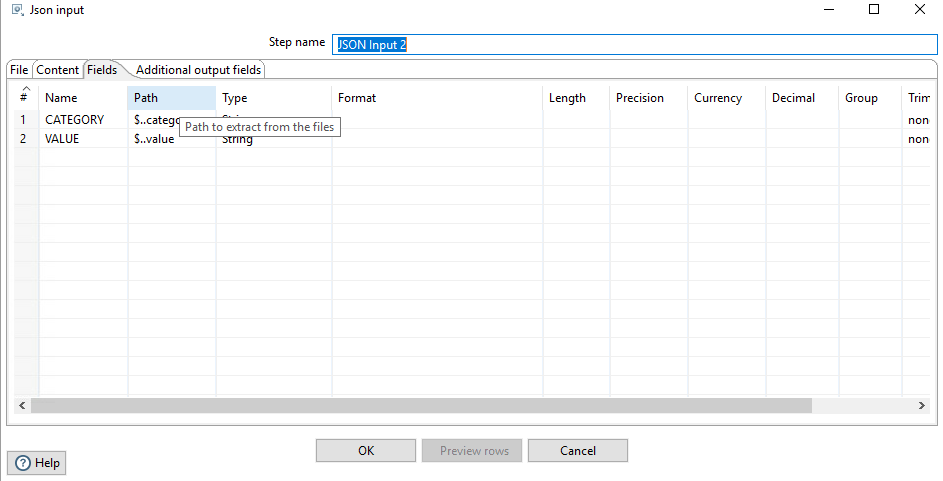
Una vez tratado el fichero procedemos a cargar el json con el step Input Json.



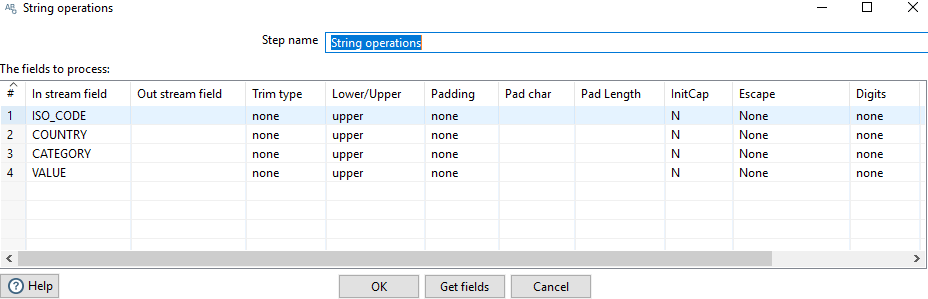
Basándonos en los ejemplos que facilita la propia herramienta podemos especificar el path de cada variable para que el step extraiga los datos como queramos.



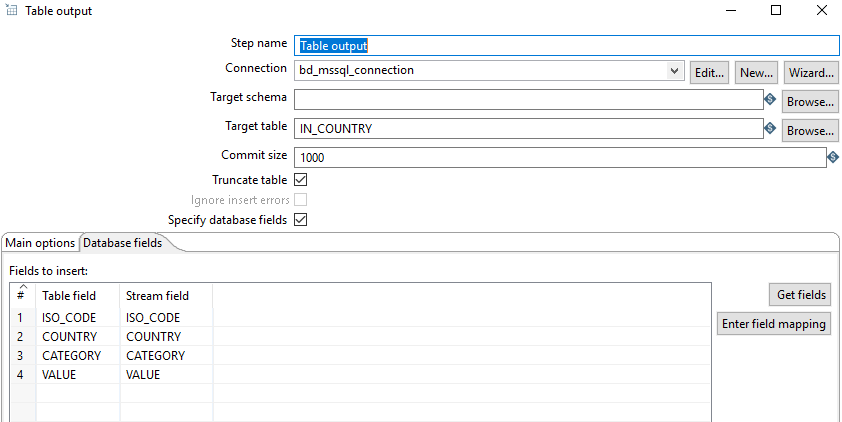
En este caso el json en cuestión requiere otro step del mismo tipo, esto ocurre cuando los ficheros tienen a tributos en mayo profundidad.



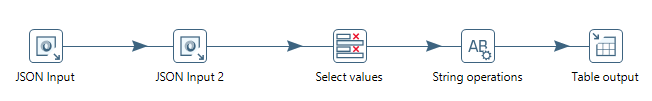
Ponemos todos los valores en mayúscula para evitar problemas posteriores al relacionar tablas.



Por último, cargamos los valores en la tabla previamente creada en el servidor.



La transformación completa es la siguiente:



**Transformación IN\_MORTCAUSE**

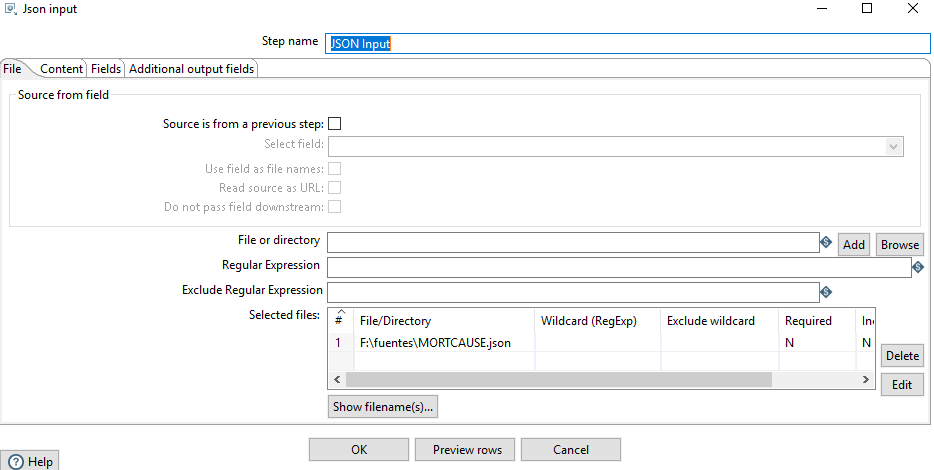
Nuevamente extraemos y cargamos la información desde un json.

La transformación consta de tres pasos: Lectura de fichero, Operación de cadea y Carga en la tabla intermedia IN\_MORTCAUSE.

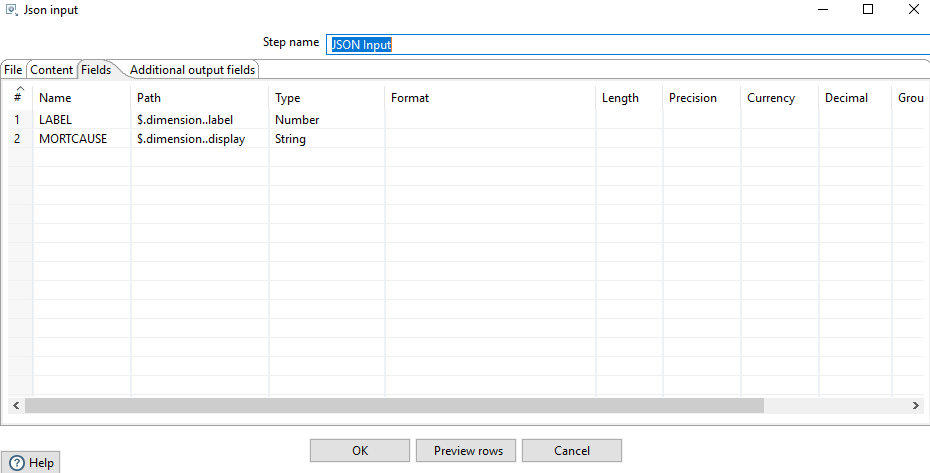
Previamente ha sido necesario modificar el primer registro del JSON pues no aportaba información y no era del tipo correcto. El registro con los valores a 0 será eliminado mediante una query desde el propio MSSQL.



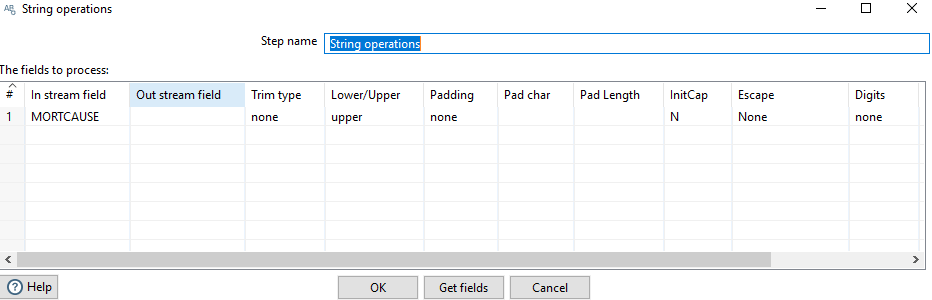
Una vez tratado el fichero procedemos a cargar el json con el step Input Json.

****

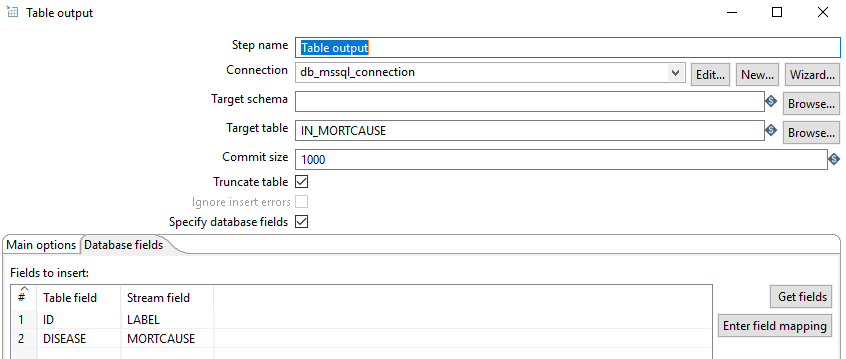
Especificamos el path de cada variable para que el step extraiga los datos como queramos.

****

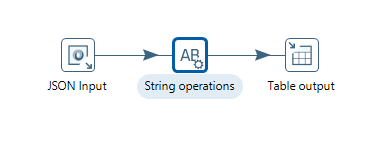
Transformamos el único atributo de tipo cadena poniéndolo en mayúsculas.

****

Por último, cargamos los valores en la tabla previamente creada en el servidor.



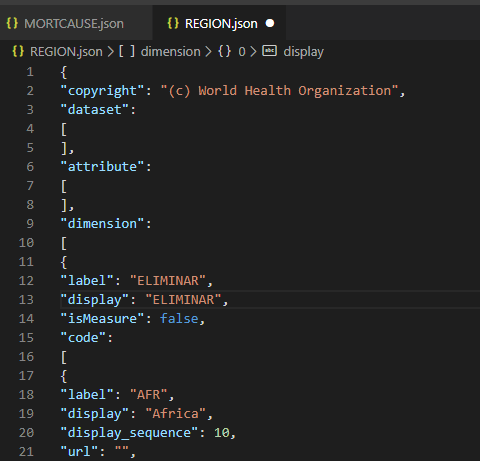
La transformación completa es la siguiente:

****

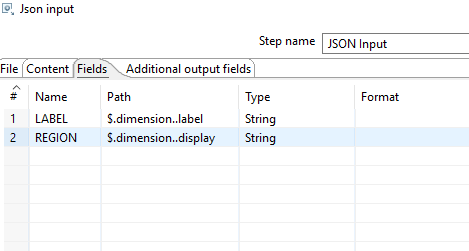
**Transformación IN\_REGION**

Extrae la información del archivo json con información sobre las regiones de la OMS y carga los en la tabla intermedia IN\_REGION.

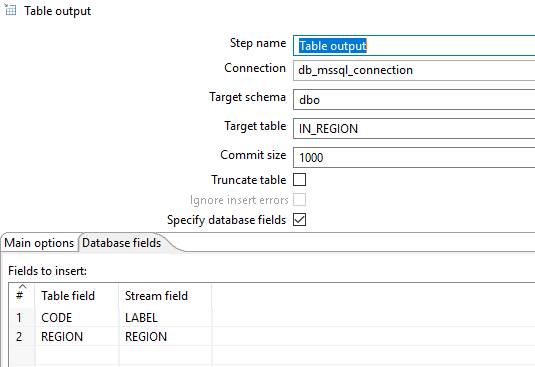
Previamente se ha modificado el primer registro del caso anterior por las mismas razones. Se eliminará posteriormente con una query desde el propio MSSQL.



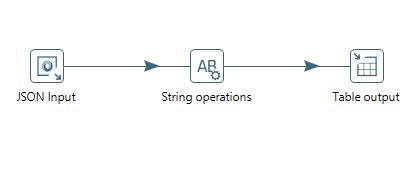
Una vez tratado el fichero procedemos a cargar el json con el step Input Json. Especificamos el path o la ruta de sus atributos.

****

Por último, cargamos los valores en la tabla previamente creada en el servidor.

****

La transformación completa es la siguiente:

****

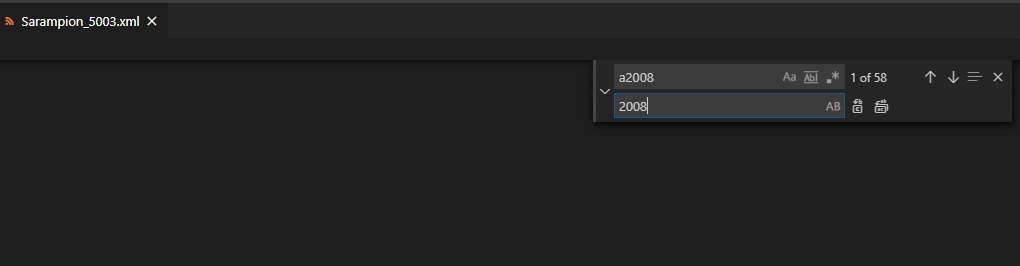
**Transformación IN\_SARAMPION**

La siguiente transformación es la más compleja de todas. Consiste en cargar simultáneamente 4 ficheros xml en la misma tabla, de manera que cada uno de ellos quede reducido a una columna.

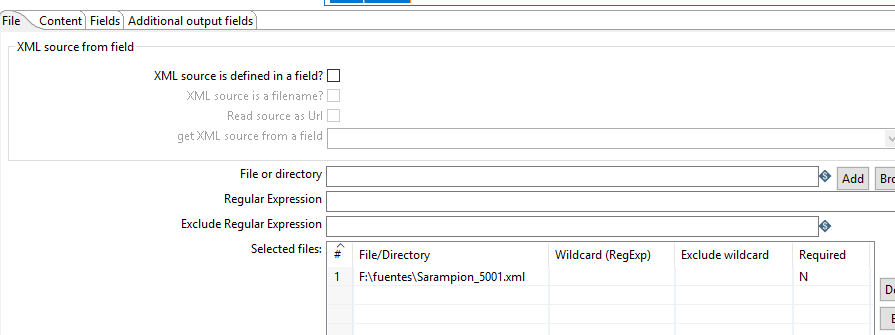
* El primer fichero se corresponde con los casos de sarampión.
* El segundo fichero se corresponde con las muertes por sarampión.
* El tercer fichero se corresponde con las hospitalizaciones por sarampión.
* El cuarto fichero se corresponde con los casos confirmados en laboratorio de sarampión.

La transformación tiene 17 pasos: 4 lecturas de ficheros, 4 operaciones de cadena, cuatro normalizadores de fila, 3 joins, 1 selección de valores y la carga en la tabla intermedia IN\_Sarampion.

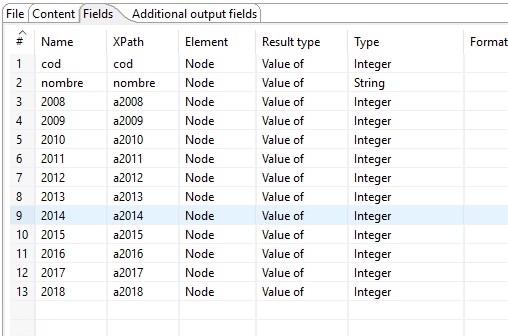
Previo a esto, los ficheros expresan los años con una letra delante, utilizamos el editor de texto para removerla.



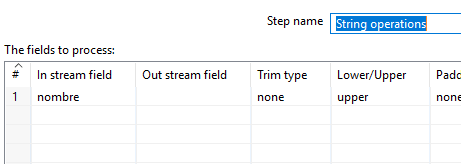
Comenzamos con la lectura del fichero, las cuatro lecturas son idénticas salvo por la ruta de la fuente. De manera que este paso solo se mostrará una vez.

****

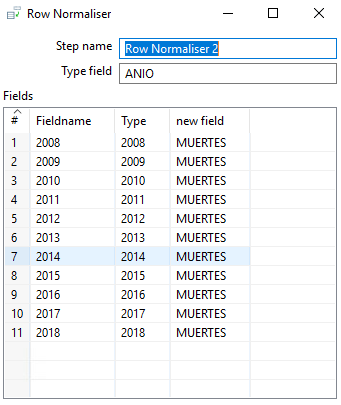
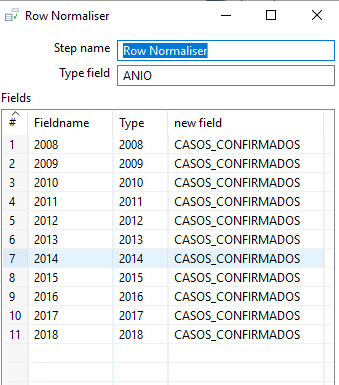
En la lectura del fichero es necesario especificar la ruta y el tipo de los atributos. En este caso es mucho más intuitivo que con los archivos json.

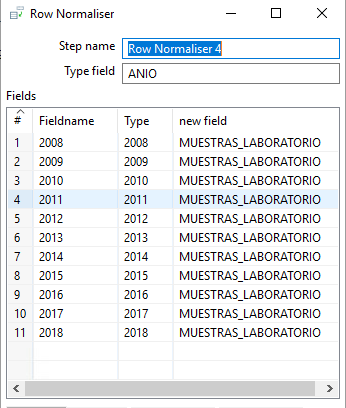
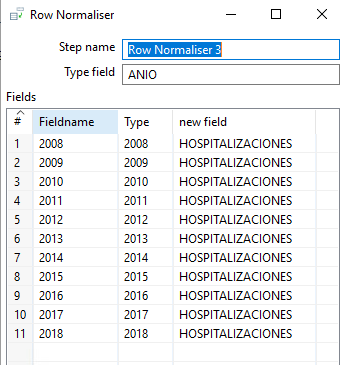
****

Ponemos en mayúscula el único valor en cada fichero que es de tipo cadena.

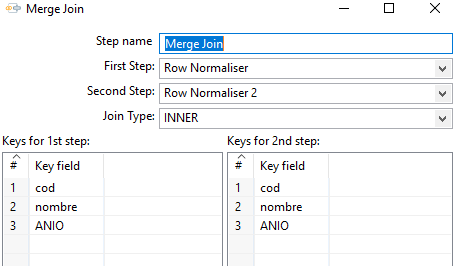
****

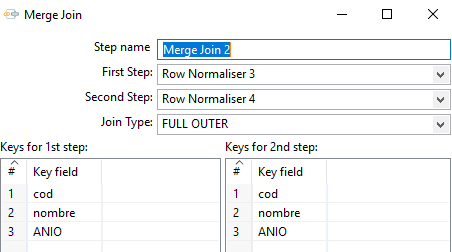
Normalización: en este caso queremos sintetizar todos los campos de los años en uno solo para que posteriormente la inserción en la tabla FACT\_SARAMPION sea mucho más sencilla. De este modo cada fichero remplaza los campos de sus años por uno nuevo.

****

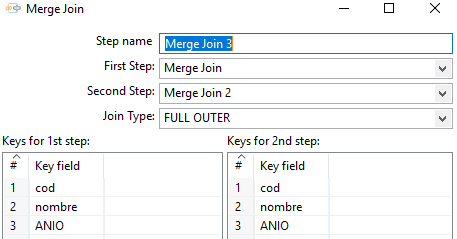
****

Utilizamos el paso “Merge Join” para unir los datos de los ficheros dos a dos en una misma entidad.

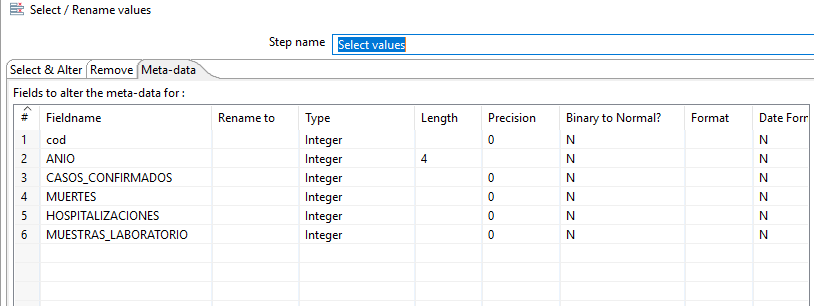
****

****

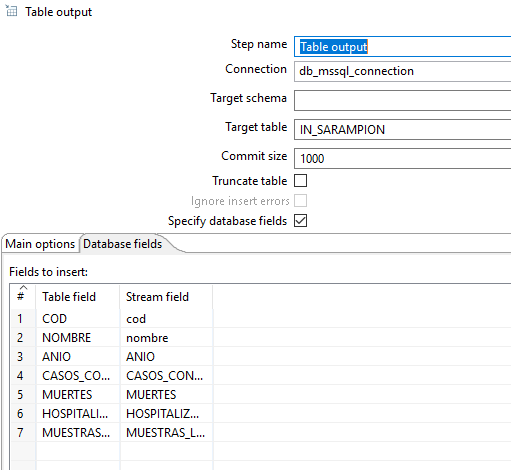
El último join se encarga de juntar definitivamente todos los atributos. Quedando un archivo con dos atributos comunes a los cuatro ficheros iniciales y cuatro nuevos campos, uno por cada normalización hecha

****

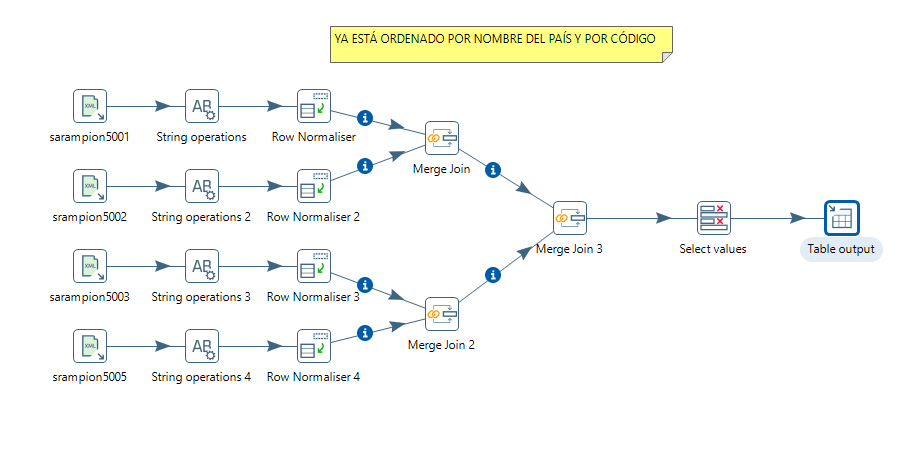
Utilizamos el paso selección de valores para asegurarnos que cada atributo es del tipo deseado.

****

Por último, cargamos los valores en la tabla previamente creada en el servidor.

****

La transformación queda así:

****

Es importante señalar que los datos estaban ya ordenados en los ficheros.

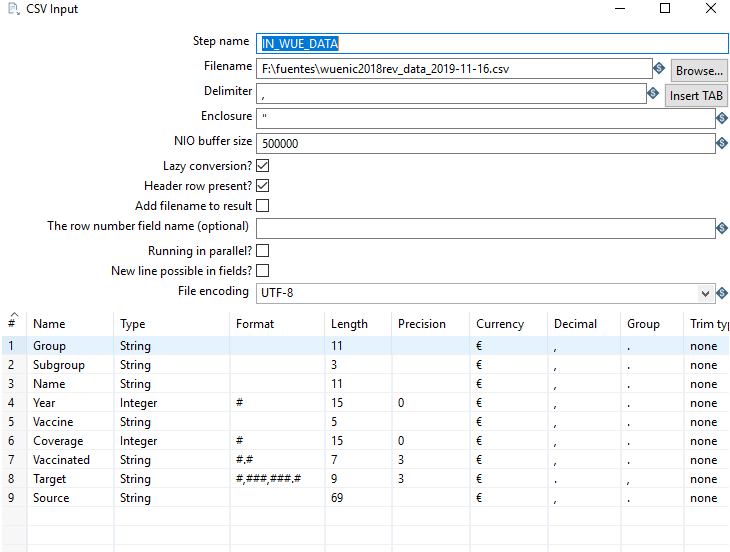
También hay que mencionar que algunas de estas transformaciones podrían realizarse en la fase siguiente de los procesos ETL en lugar de en esta sin ningún inconveniente.

**Transformación IN\_WUE\_DATA**

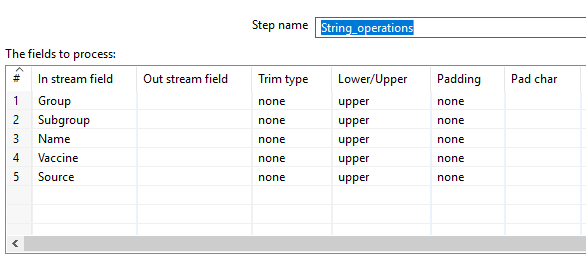
La siguiente transformación carga el contenido de un csv en la tabla intermedia IN\_WUE\_DATA.

Consta de cuatro pasos: Lectura del fichero, Operación de cadena, Ordenación de filas y Carga en la tabla.

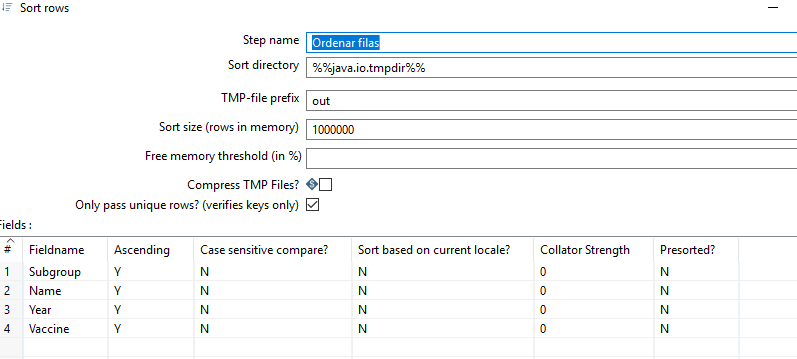
Cargamos el fichero csv, delimitador: “;”. Todos los valores son los que vienen por defecto.

****

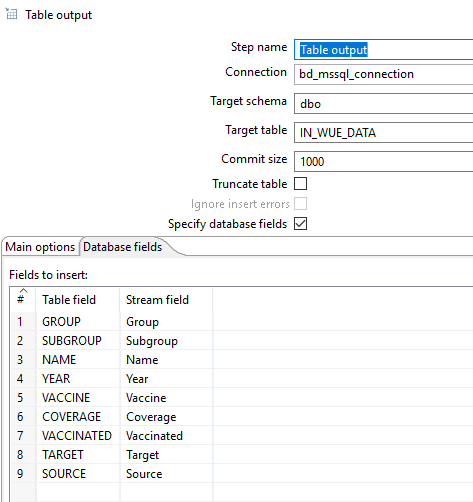
Ponemos todos los atributos de tipo cadena en mayúscula:

****

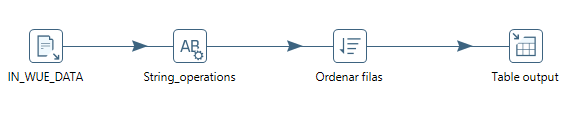
Ordenamos los registros de la siguiente manera:

****

Por último, cargamos los datos en la tabla previamente creada en el servidor.

****

La transacción queda finalmente:

****

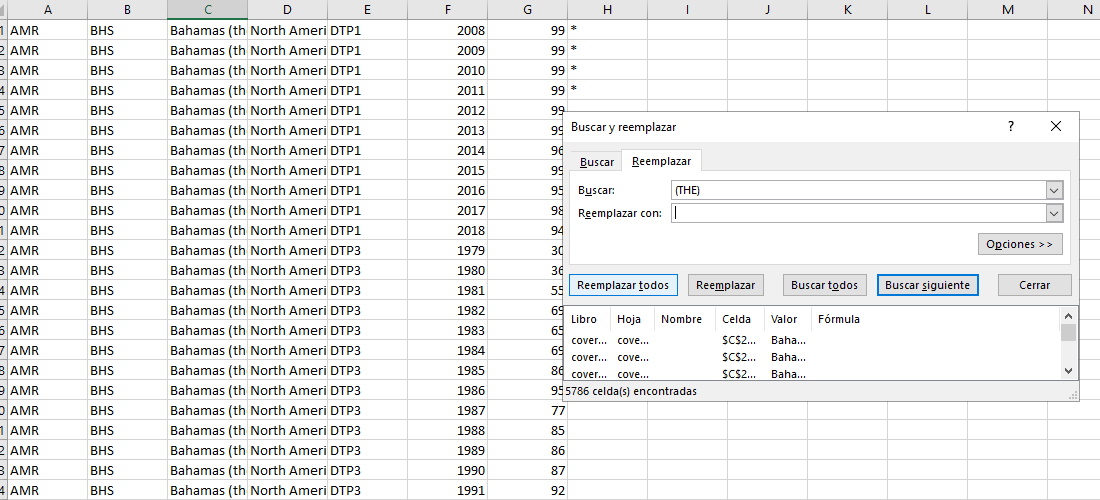
**Transformación IN\_COV\_SERIES**

En la siguiente transformación se carga en la tabla IN\_OCV\_SERIES el contenido del fichero coverage\_estimated\_series.xls. Dicho fichero recoge los datos historificados de cobertura de vacunación de los países de la OMS.

La transformación consiste en 5 pasos: Lectura del fichero, Operación de cadena, Ordenación de filas, Filtrado de filas y carga en la Tabla.

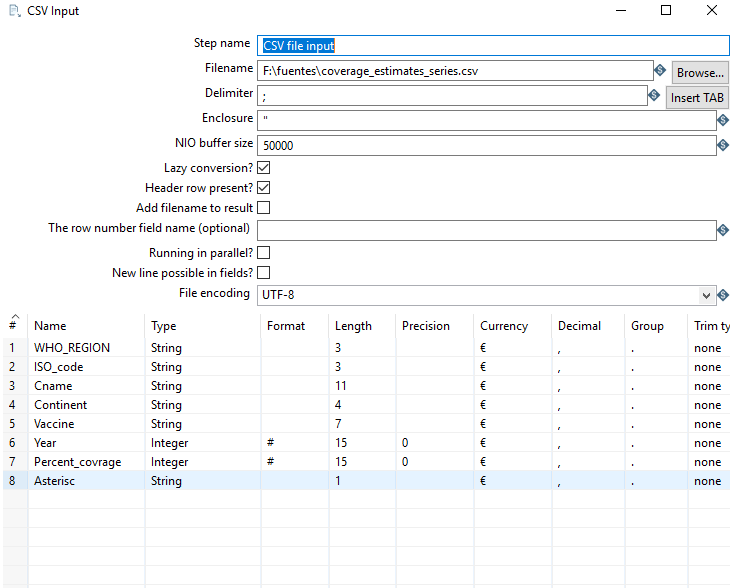
Previo a la transformación es necesario remover algunos valores de la columna de los países. Resulta que todos los países que empiezan por THE tienen dicha palabra al final del registro y entre paréntesis, mientras que en los países ya cargados del fichero COUNTRY.json esta palabra ha sido omitida.

Es por ello que debemos eliminar dicho artículo para que no haya campos vacíos que no deberían cuando relacionemos las tablas.

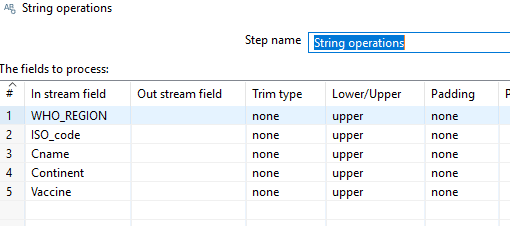


A algunos de los alumnos nos ha surgido un raro error en el step para leer archivos xls que provoca que el paso no lea el contenido pese a ejecutarse correctamente. Es por ello por lo que he decido guardar el archivo como csv.

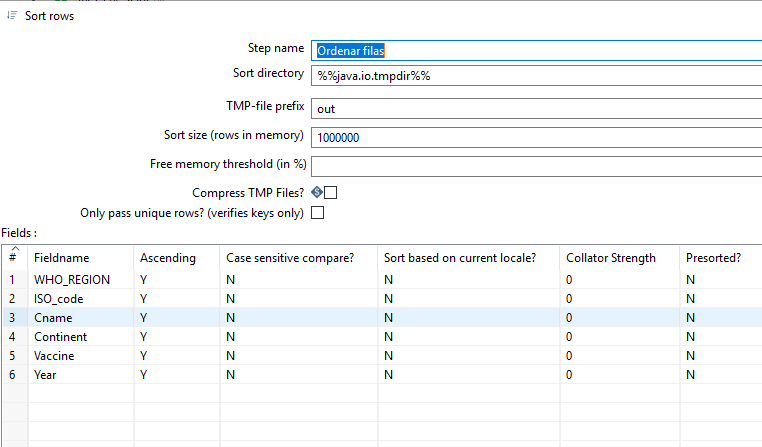
Cargamos el fichero csv, delimitador: “;”.

****

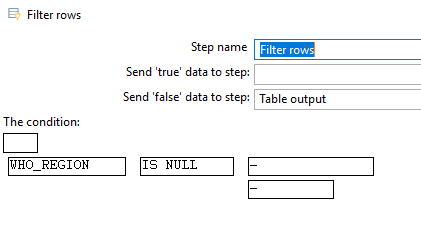
Ponemos en mayúsculas todas las cadenas.

****

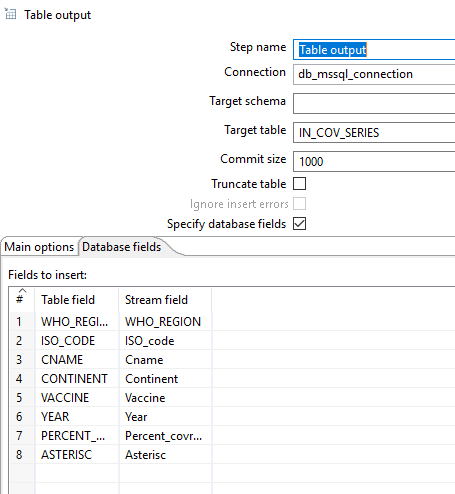
Ordenamos los registros de la siguiente manera:

****

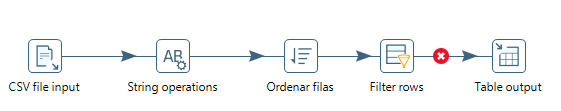
Tras ordenar los registros aparecieron dos filas completamente vacías, es por ello que filtramos los registros descartando las filas cuando la columna WHO\_REGION, la cual no debería ser nula nunca, lo es.

****

Por último, cargamos el fichero en la tabla intermedia IN\_COV\_SERIES

****

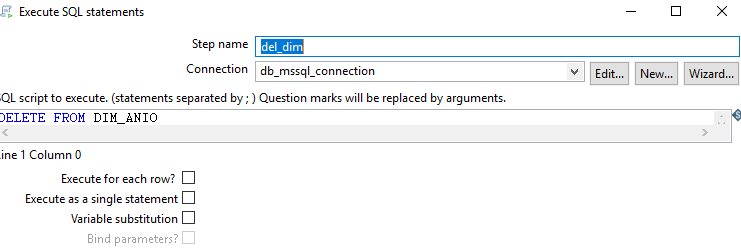
La transformación queda con este aspecto:

****

1. Bloque IN
2. Transformaciones Bloque TR\_DIM

Este bloque, contiene las transformaciones para la carga inicial de las dimensiones al almacén desde las tablas intermedias IN\_. Además de la carga inicial de las dimensiones del modelo multidimensional, se tendrá en cuenta que se puedan ejecutar las transformaciones las veces que sean necesarias, por esto se incluirá como primer paso de las transformaciones del bloque TR\_DIM un borrado de las tablas de dimensiones. Lo mismo se hará en las TR\_FACT.

A continuación, se muestra dicho paso para la dimensión DIM\_ANIO, la query mostrada será la misma para todas las tablas cambiando el valor correspondiente a la tabla.

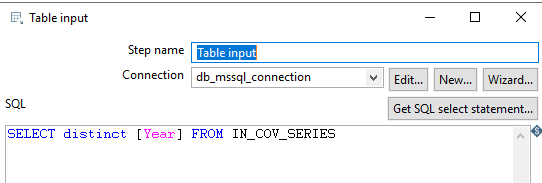


**TR\_DIM\_ANIO**

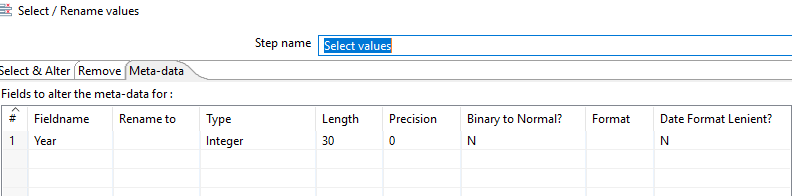
A continuación, se muestra la carga de la tabla DIM\_ANIO.

La transformación consta de cinco pasos: Carga de la tabla desde nuestra bd, Selección de valores, Mapeador de valores, Ordenación de filas y carga en tabla.

Cargamos la columna año de la tabla IN\_COV\_SERIES:



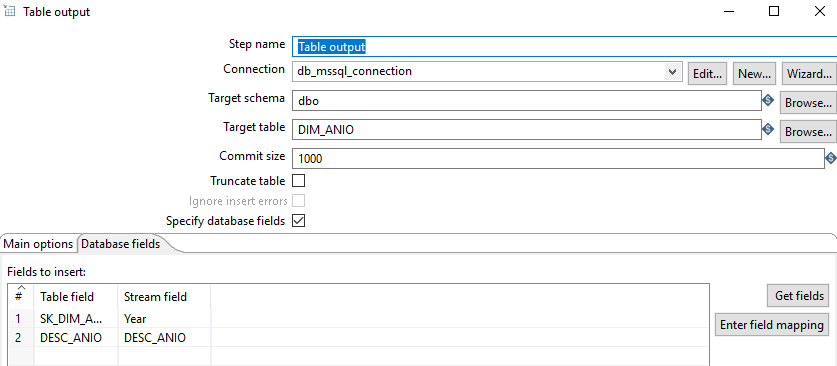
Utilizamos el paso selección de valores para combrobar el tipo y que la carga ha sido buena en cuanto a los meta-datos se refiere.



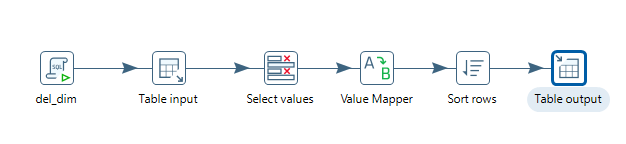
Con el mapeador de valores generamos una nueva columna con información complementaria a cada año.



Ordenamos los valores por año de menor a mayor y cargamos los datos en la tabla DIM\_ANIO.



La transformación es la siguiente:



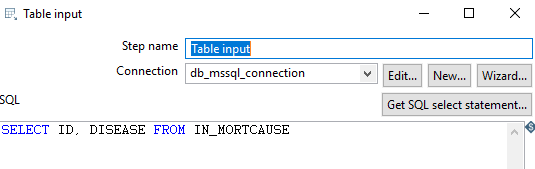
El resultado de la ejecución es:

**TR\_DIM\_CAUSAS\_MORTALIDAD**

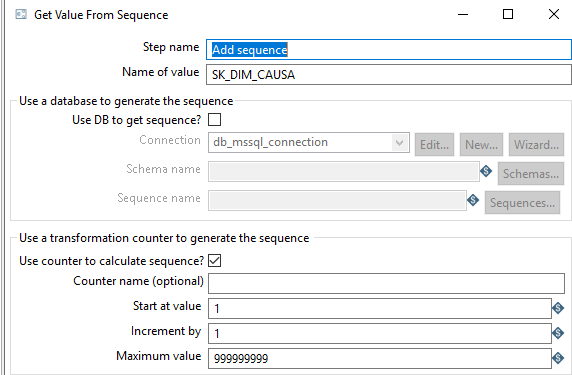
En la siguiente transformación cargaremos la información deseada en la tabla DIM\_CAUSAS\_MORTALIDAD.

Consta de tres pasos: Lectura de tabla, Añadir secuencia y Carga de Tabla.

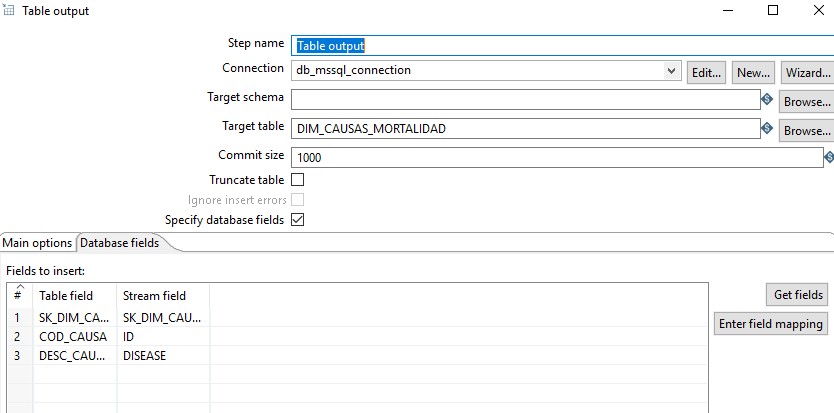
Extraemos los dos registros de la tabla IN\_MORTCAUSE.



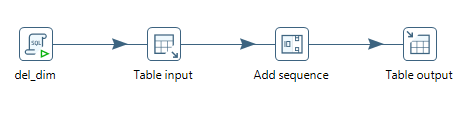
Añadimos como clave primaria una columna con una secuencia de dígitos.



Por último, cargamos la tabla de la base de datos.



La transformación final es la siguiente:

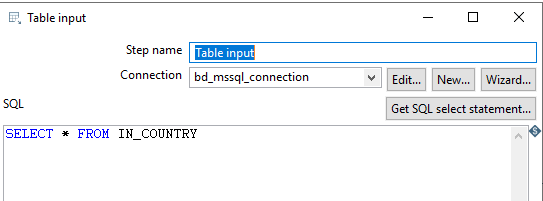


**TR\_DIM\_PAIS**

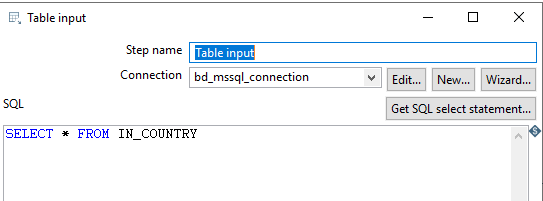
En la siguiente transformación cargaremos la información deseada en la tabla DIM\_PAIS. Esta transformación tiene como peculiaridad que es necesario cargar datos que se corresponden con otros registros de otras tablas.

Consta de seis pasos: Lectura de tabla, Desnormalización de fila, Búsqueda en la base de datos, Ordenación de fila, Añadir secuencia y Carga de Tabla.

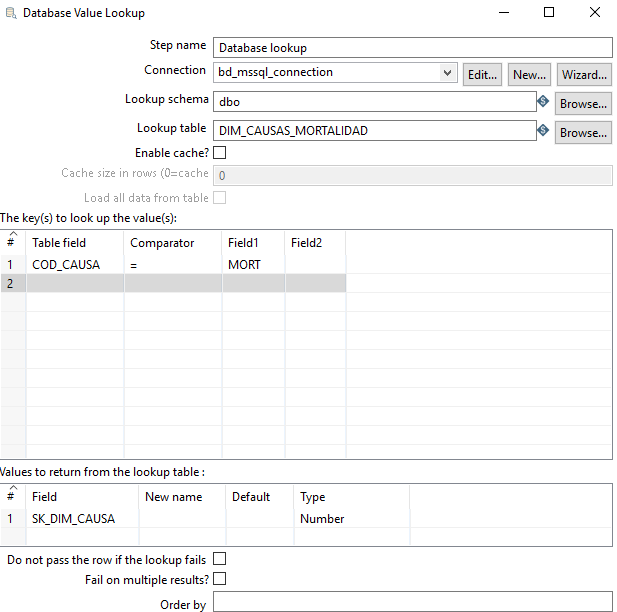
Cargamos todos los valores de IN\_COUNTRY.



En el paso de desnormalización lo que hacemos es extraer todos los campos deseados que estaban en las columnas de category y value en la anterior tabla.

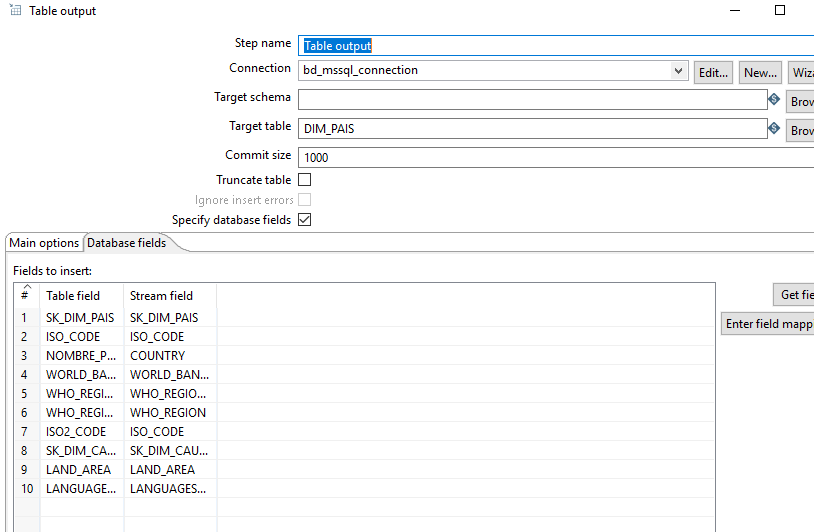


Nos interesa la clave de la tabla de causas de mortalidad, para ello usamos la columna MORT y la comparamos con una de la tabla para que nos devuelva el valor que queremos, SK\_MID\_CAUSA.

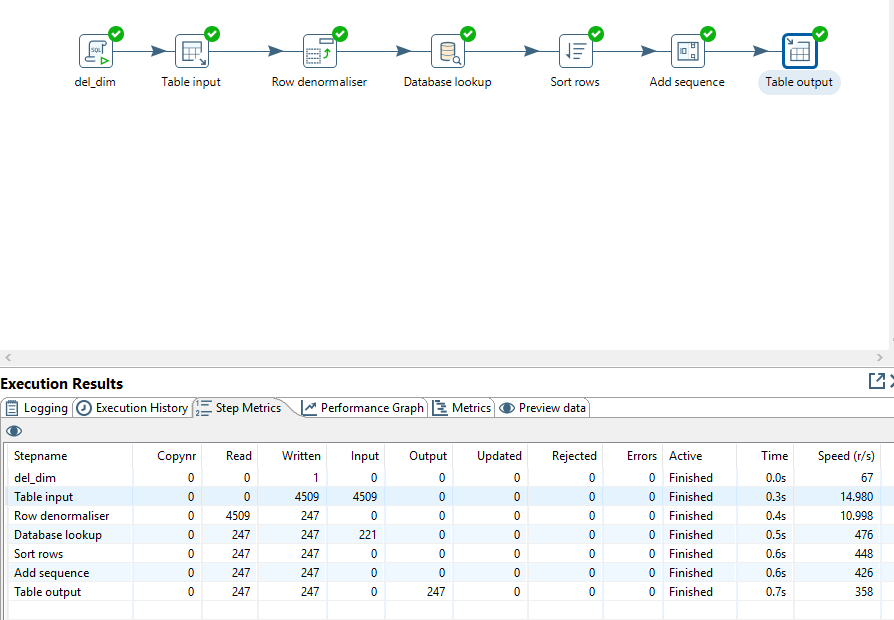


Ordenamos la tabla por orden ascendente según el atributo ISO\_CODE y añadimos una columna con una secuencia numérica que será la clave de esta tabla.

Por último, cargamos la tabla en nuestra base de datos:



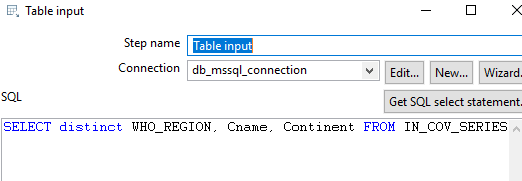
La transformación final y el resultado de sue ejecución es el siguiente:



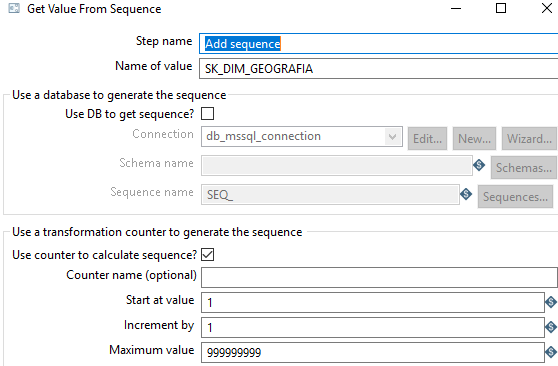
**TR\_DIM\_ GEOGRAFÍA**

La siguiente transformación conta de 6 pasos: Entrada de tabla, Orden de filas, Añadir secuencia, Búsqueda en base de datos, Selección de valores y Carga de Tabla.

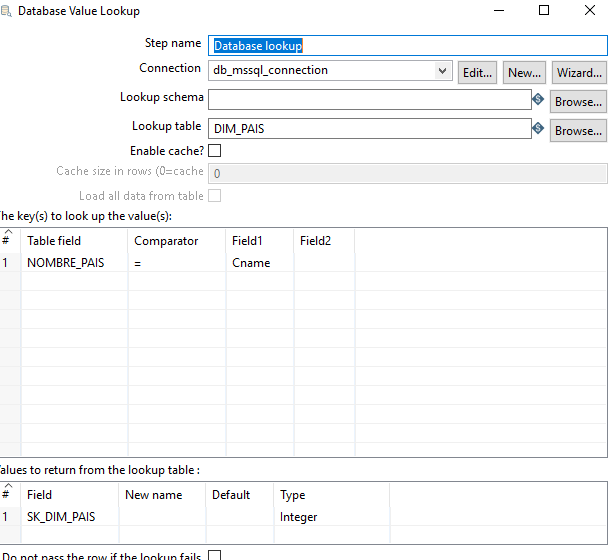
Extraemos cuatro valores de la tabla IN\_COV\_SERIES



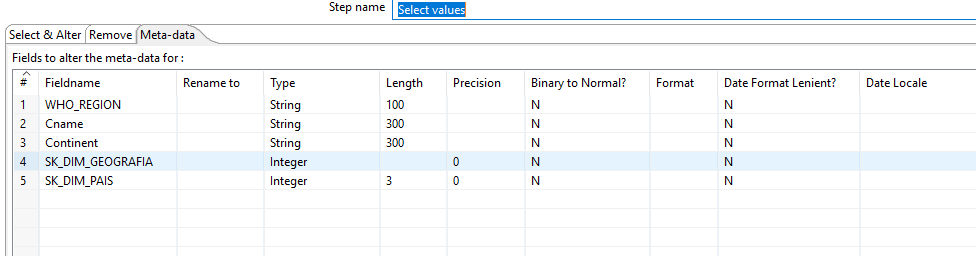
Ordenamos las filas por WHO\_REGION y Cname.

Añadimos clave primaria.

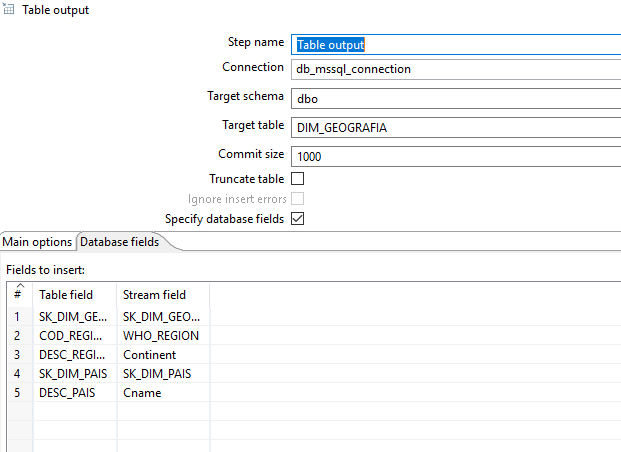
Con el nombre del país queremos conseguir la clave primaria correspondiente a ese país de la tabla DIM \_PAIS.



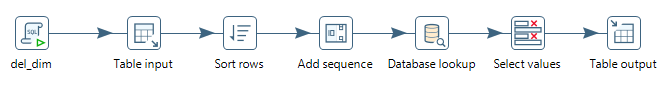
Hacemos una comprobación de los metadatos con la selección de valores:



Y cargamos en la tabla.



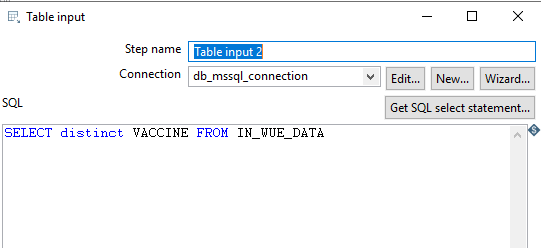
La transformación final tiene este aspecto:



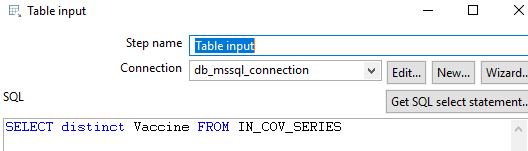
**TR\_DIM\_VACUNA**

La siguiente transformación consta de trece pasos: 2 lecturas de tabla, 3 Ordenaciones de fila, fusión ordenada, 2 Mapeados de valores, Operaciones de cadena, Filtrado de filas, Ordenación de filas, Filas únicas, Añadir secuencia, Salida en tabla.

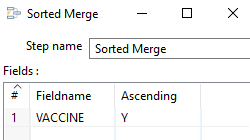
Cargamos los distintos tipos de vacuna de la tabla IN\_WUE\_DATA.



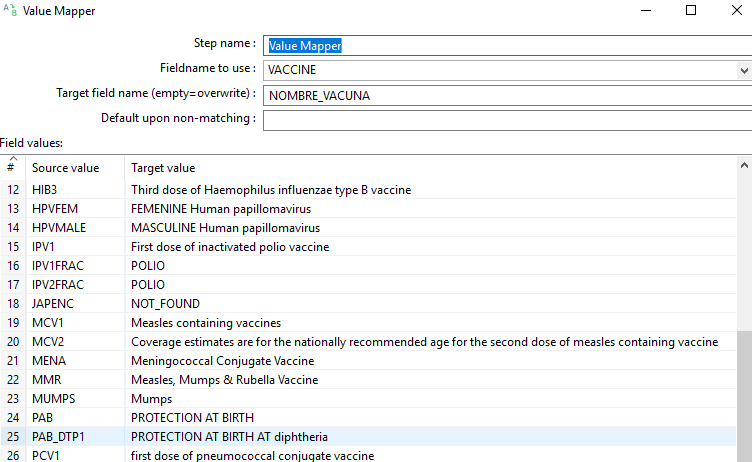
Hacemos lo mismo con la tabla IN\_COV\_SERIES.



Ordenamos ambas tablas y las unimos con el paso “sorted merge”

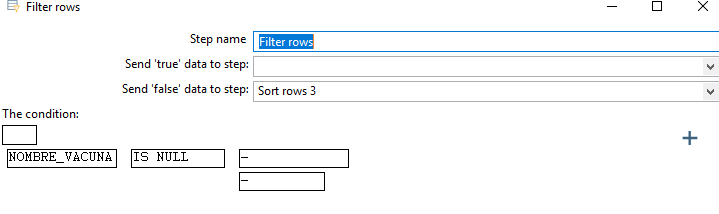


Tras mucho indagar en internet ha sido posible generar un campo con el nombre de cada vacuna.



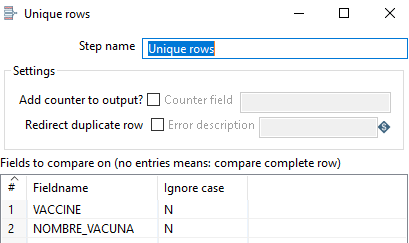
Con operaciones de cadena ponemos en mayúscula ambas columnas.

Filtramos por si hubiera algún valor nulo:

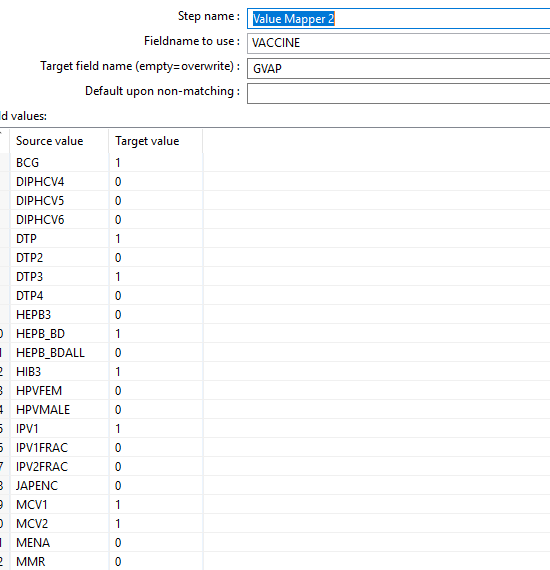


Reordenamos por si el filtrado hubiera desordenado los datos.

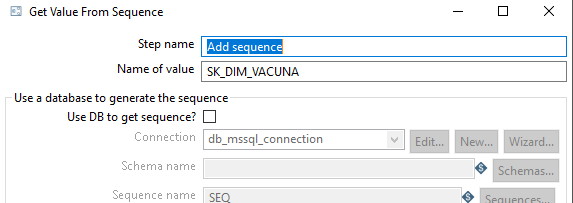
Eliminamos las filas repetidas con el paso filas únicas:



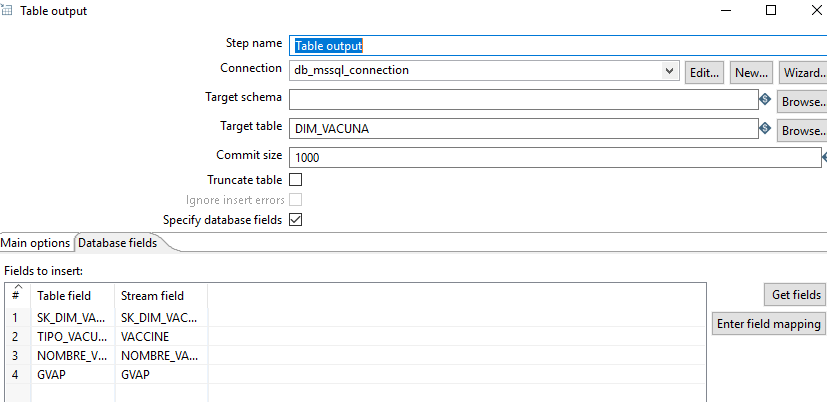
Generamos una columna para las vacunas del GVAP con el último mapeador de valores.



Añadimos la clave primaria como en anteriores ocasiones.



Por último, cargamos la última tabla de las dimensiones.



1. Transformaciones Bloque TR\_FACT

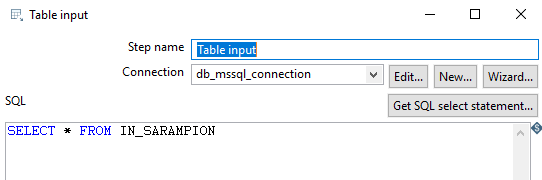
En este apartado, una vez cargadas en la base de datos tanto las fuentes como las dimensiones, el siguiente paso consiste en desarrollar las transacciones para poblar nuestras tablas de hechos.

**Transformación TR\_FACT\_SARAMPION**

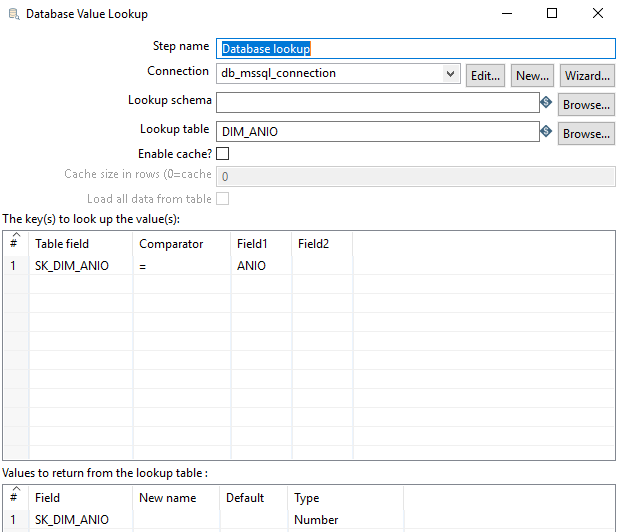
Mediante esta transformación poblaremos la tabla de hechos FACT \_SARAMPION según el modelo multidimensional definido y que nos permitirá realizar un análisis evolutivo de la cobertura de inmunización de la enfermedad del sarampión por año, país.

La transformación tiene 5 pasos: Carga de tabla, 2 Búsquedas en base de datos y Salida en tabla.

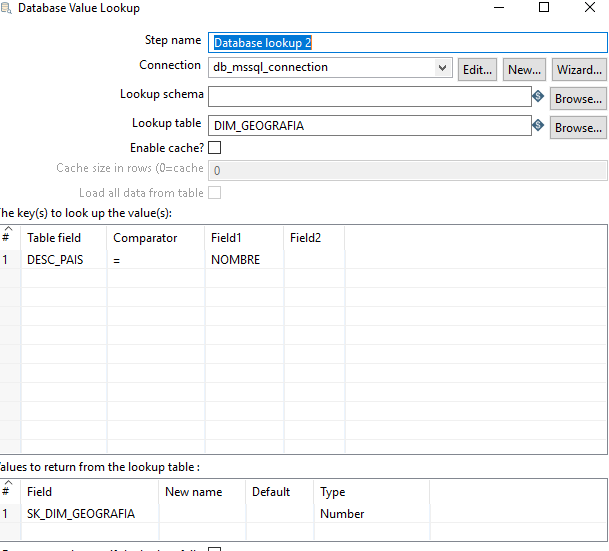
Comenzamos cargando todos los atributos de la tabla IN\_SRAMPION.



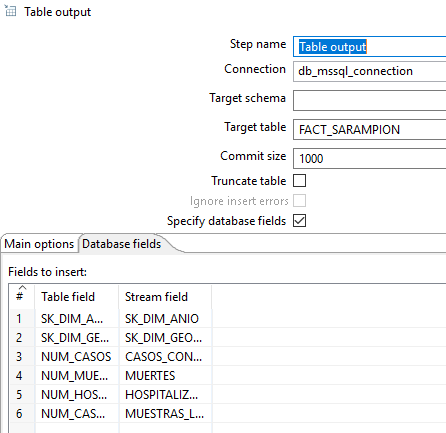
Extraemos, usando el atributo año, la clave primaria SK\_DIM\_ANIO.



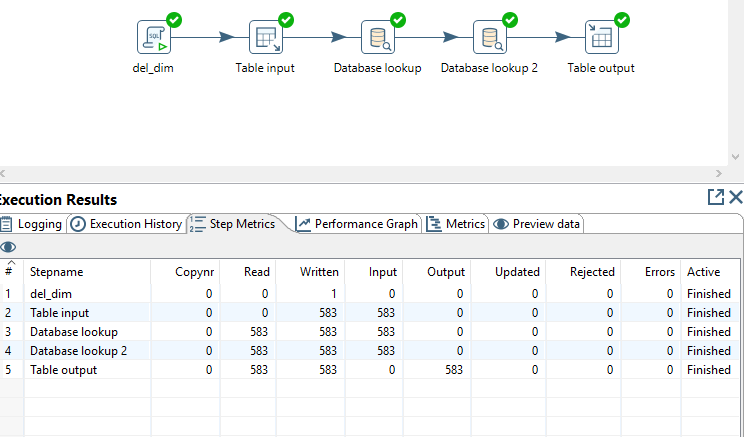
Repetimos el proceso, esta vez utilizando el nombre del país para extraer la clave de la tabla DIM\_GOGRAFIA.



Por, último, cargamos todas las columnas obtenidas con sus correspondientes en la tabla FACT\_SARAMPION.



La transformación final y el resultado de la ejecución tienen este aspecto.



**Transformación TR\_FACT\_COBERTURA**

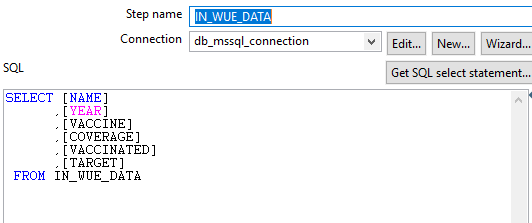
Mediante esta transformación poblaremos la tabla de hechos FACT\_COBERTURA según el modelo multidimensional definido y que nos permitirá realizar un análisis evolutivo de la cobertura de inmunización por año, país y vacuna.

Esta transformación, a pesar de ser una sola, actúa con dos flujos de trabajo de forma paralela.

Flujo 1:

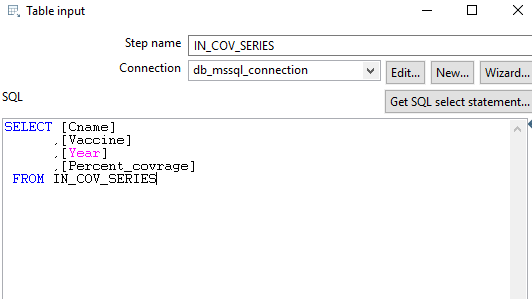
La transformación tiene 5 pasos: Carga de tabla, 3 Búsquedas en base de datos y Salida en tabla.

Cargamos los atributos necesarios de la tabla IN\_WUE\_DATA.



Flujo2:

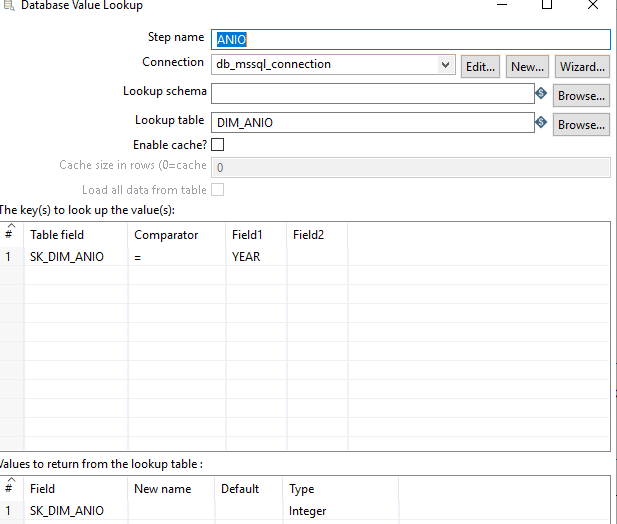
Hacemos lo mismo con la tabla IN\_COV\_SERIES-



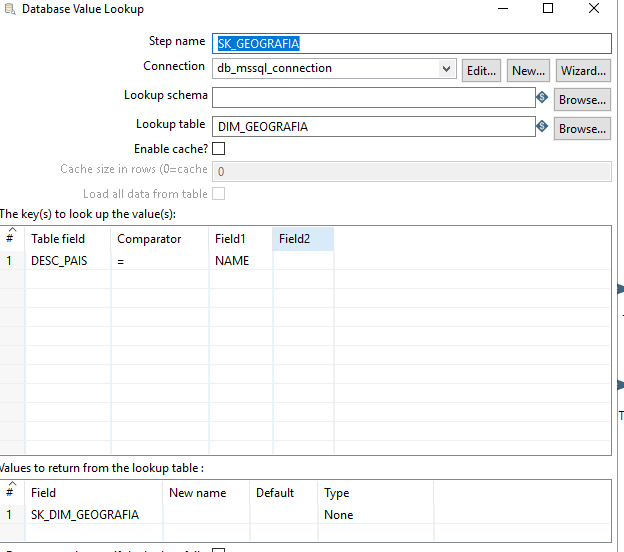
Con el atributo año, name y vacuna cargamos las calves principales de las tablas DIM\_ANIO, DIM\_GEOGRAFIA y DIM\_VACUNA, respectivamente.

El proceso es el mismo en ambos flujos.

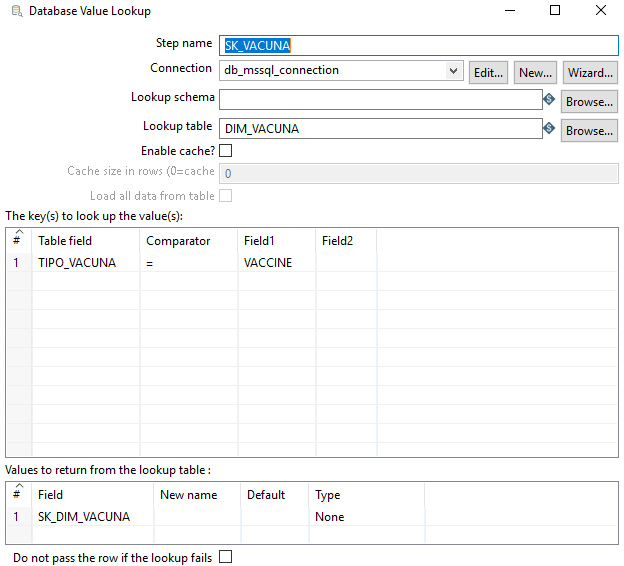
Obtenemos SK\_DIM\_ANIO:



Obtenemos SK\_DIM\_GEOGRAFIA:

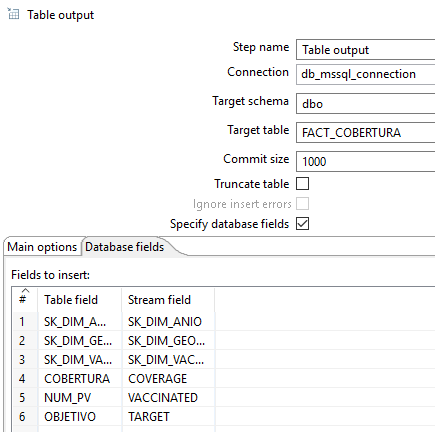


Obtenemos SK\_DIM\_VACUNA.

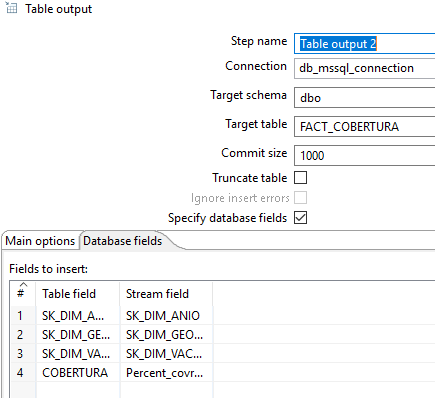


Cargamos los datos en la tabla FACT\_COBERTURA.

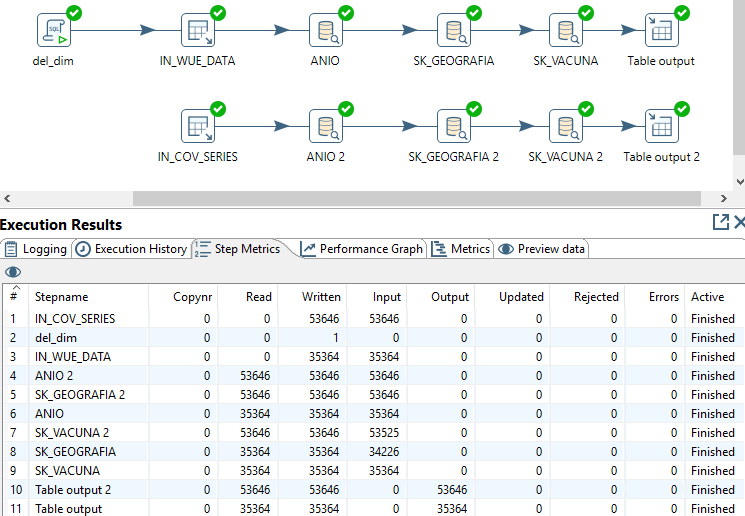
Flujo 1:



Flujo 2:



El resultado final de la ejecución es el siguiente:

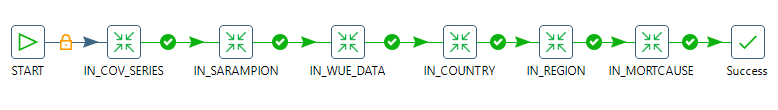


La razón por la que se ha implementado este diseño de dos flujos es porque hay dos atributos provenientes de la fuente IN\_COV\_SERIES que no existen y esta es la manera más sencilla de cargarlos todos los datos en la tabla FACT\_COBERTURA.

1. Implementación de trabajos con procesos ETL´s.

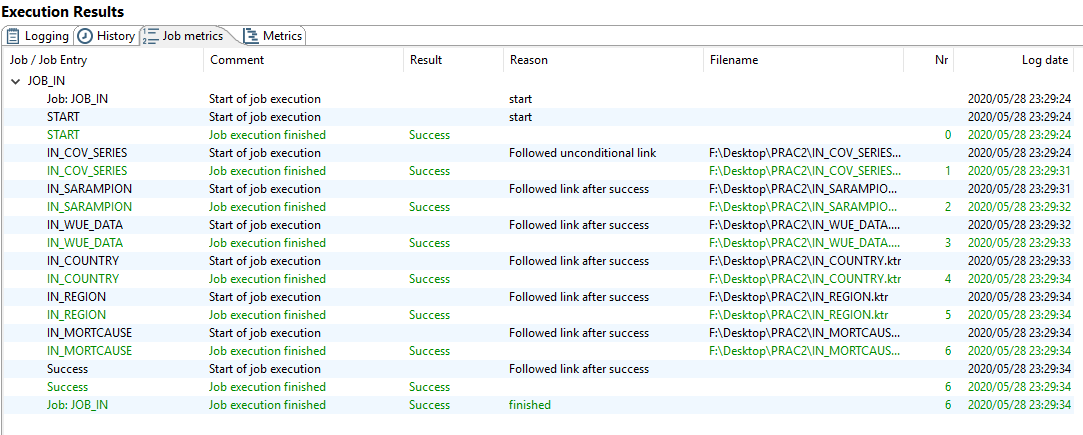
Vamos a diseñar los trabajos (jobs) mediante PDI que van a permitir la ejecución secuencial de todos los procesos ETL’s incluidos en cada bloque definido. Cada trabajo contiene como pasos cada una de las transformaciones implementadas en el apartado anterior de Diseño de ETL’s.

**JOB\_IN**

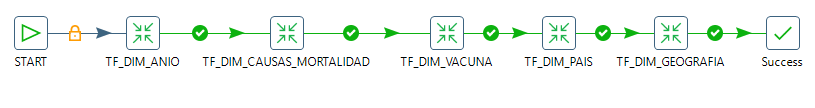


* Start: Componente General de diseño de trabajos, que marca el Inicio del mismo.
* IN\_COV\_SERIES: Ejecución de transformación IN\_COV\_SERIES.
* IN\_SARAMPION: Ejecución de transformación IN\_SEGR\_N.
* IN\_EGR\_C16\_17: Ejecución de transformación IN\_ SARAMPION.
* IN\_WUE\_DATE: Ejecución de transformación IN\_WUE\_DATA.
* IN\_COUNTRY: Ejecución de transformación IN\_COUNTRY.
* IN\_REGION: Ejecución de transformación IN\_REGION.
* IN\_MORTCAUSE: Ejecución de transformación IN\_MORTCAUSE.
* Success: Componente General de diseño de trabajos, que marca la Finalización del trabajo.

Resultados:



**JOB\_TR\_DIM**

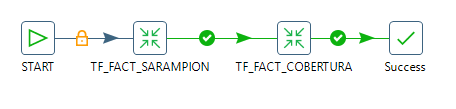
****

* Start: Componente General de diseño de trabajos, que marca el Inicio del mismo.
* TF\_DIM\_ANIO: Ejecución de transformación TF\_DIM\_ANIO.
* TF\_DIM\_CAUSAS\_MORTALIDAD Ejecución de transformación TF\_DIM\_CAUSAS \_MORTALIDAD.
* TF\_DIM\_VACUNA: Ejecución de transformación TF\_DIM\_VACUNA.
* TF\_DIM\_PAIS: Ejecución de transformación ITF\_DIM\_PAIS.
* TF\_DIM\_GEOGRAFIA: Ejecución de transformación TF\_DIM\_GEOGRAFIA.
* Success: Componente General de diseño de trabajos, que marca la Finalización del trabajo.

No es posible re-ejecutar el trabajo para plasmar los resultados en esta memoria debido a que las dimensiones no se pueden vaciar a causa de las dependencias que hay entre ellas y con los hechos.

No obstante, en los siguientes apartados se verán los datos como prueba de que han sido cargados correctamente.

**JOM\_TF\_FACT**

****

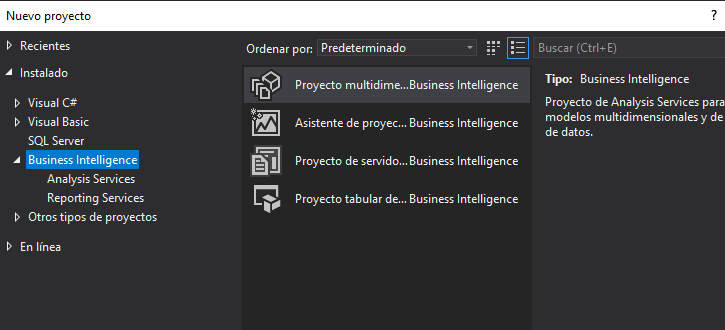
* Start: Componente General de diseño de trabajos, que marca el Inicio del mismo.
* TF\_FACT\_SARAMPION: Ejecución de transformación TF\_FACT\_SARAMPION.
* TF\_FACT\_COBERTURA Ejecución de transformación TF\_FACT\_COBERTURA.
* Success: Componente General de diseño de trabajos, que marca la Finalización del trabajo.

No es posible re-ejecutar el trabajo para plasmar los resultados en esta memoria debido a que los hechos no se pueden vaciar a causa de las dependencias que hay entre ellos y con las dimensiones.

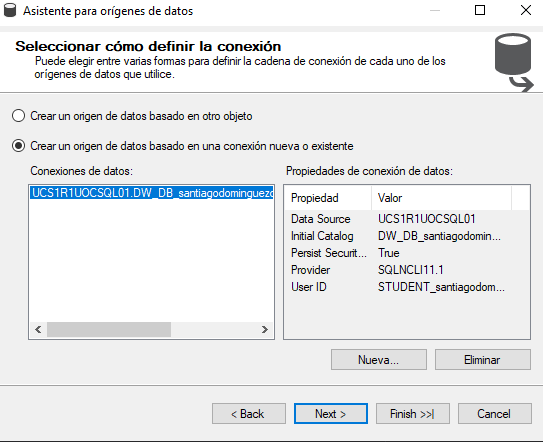
No obstante, en los siguientes apartados se verán los datos como prueba de que han sido cargados correctamente.

1. Creación de modelo OLAP

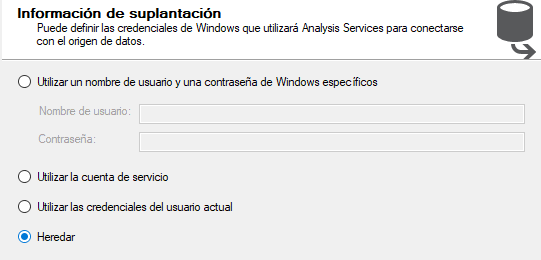
Creamos el proyecto del tipo “proyecto multidimensional y de minería de datos” que tanto permite crear cubos como proyectos de minería de datos.

****

Conectamos nuestro proyecto la base de datos creando un nuevo origen de datos.

****

Como no tenemos información de qué usuario va a acceder a cada parte de nuestro cubo, seleccionamos la opción “Heredar”

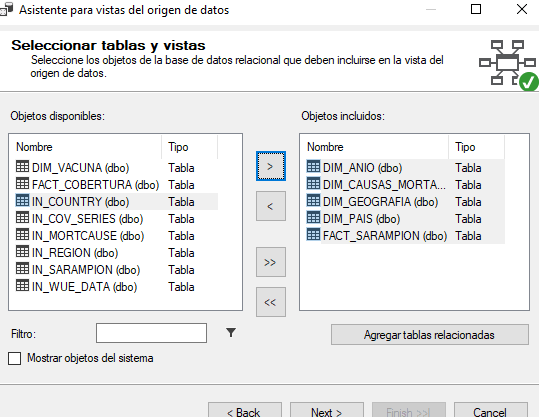
****

1. **VISTAS DEL ORIGEN DE DATOS.**

Crearemos dos vistas del origen de datos, una por cada diseño lógico diseñado en la primera parte de la práctica. En el explorador de soluciones >vistas> click derecho > nueva vista.

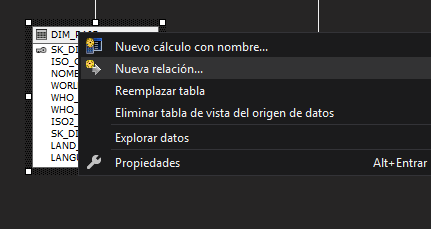
**VISTA 1.**

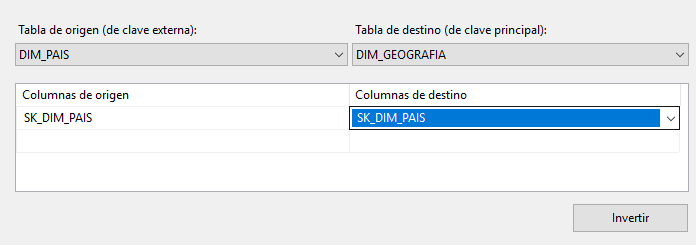
Esta vista recoge el diseño lógico de la implementación de FACT\_SARAMPION.

****

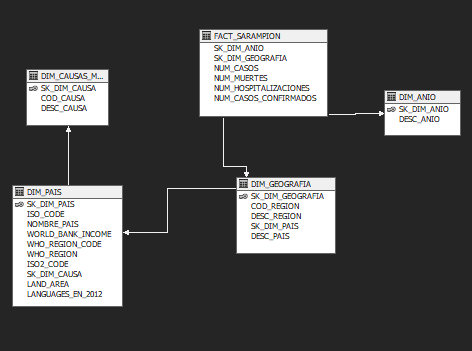
Es frecuente que no se carguen las relaciones entre tablas correctamente, de modo que creamos nosotros las que faltan.

Por ejemplo, a continuación, se muestra para el caso entre las tablas DIM\_PAIS Y DIM\_GEOGRAFIA.

****

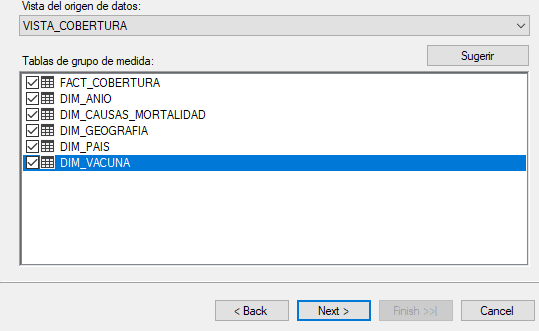
****

El resultado de la vista de datos es el siguiente:

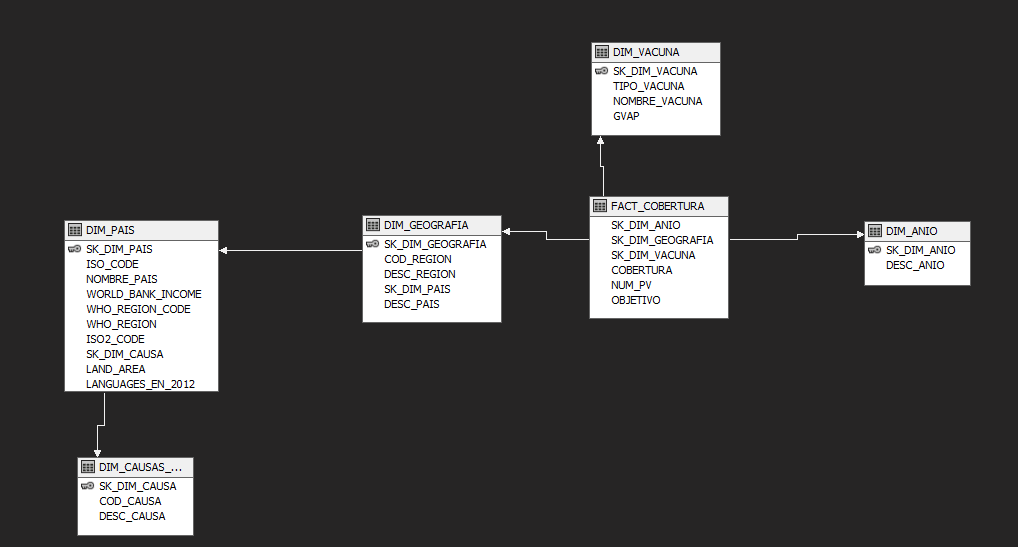
****

**VISTA 2.**

Esta vista recoge el diseño lógico de la implementación de FACT\_COBERTURA.

****

Arreglamos las relaciones que no se hayan cargado, posteriormente este el diseño resultante:

****

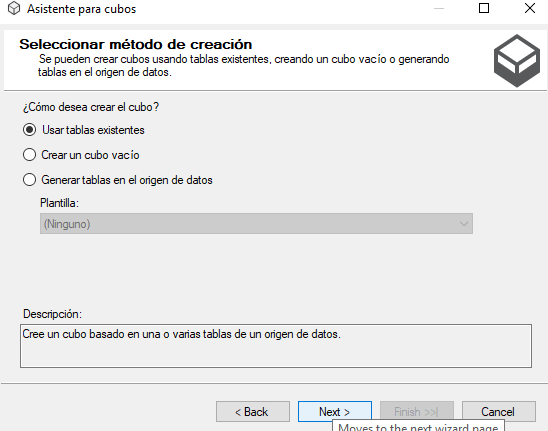
1. **CUBOS.**

Ahora ya estamos preparados para crear los cubos con los modelos multidimensionales correspondientes. Vamos a crear 2 cubos, uno para cada vista de origen de datos.

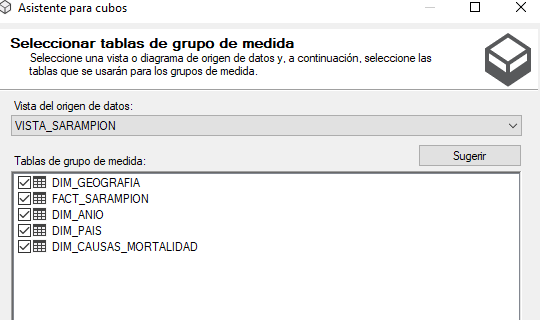
**CUBO 1**

Empezamos con el primer cubo, creamos un nuevo cubo desde el menú contextual de cubos para que aparezca el asistente para cubos.

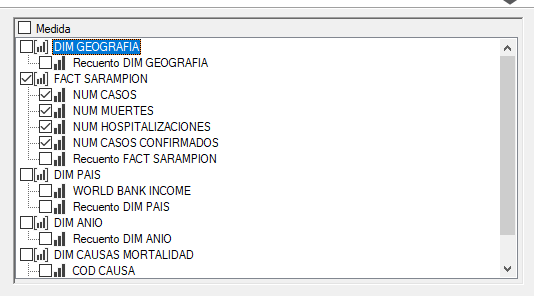
Seleccionamos la opción de “Usar tablas existentes” para generar el cubo.

****

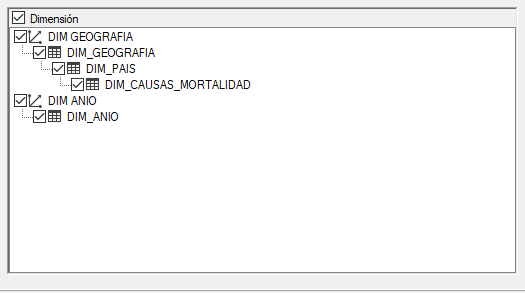
El primer cubo utilizará las tablas de la primera vista generada.

****

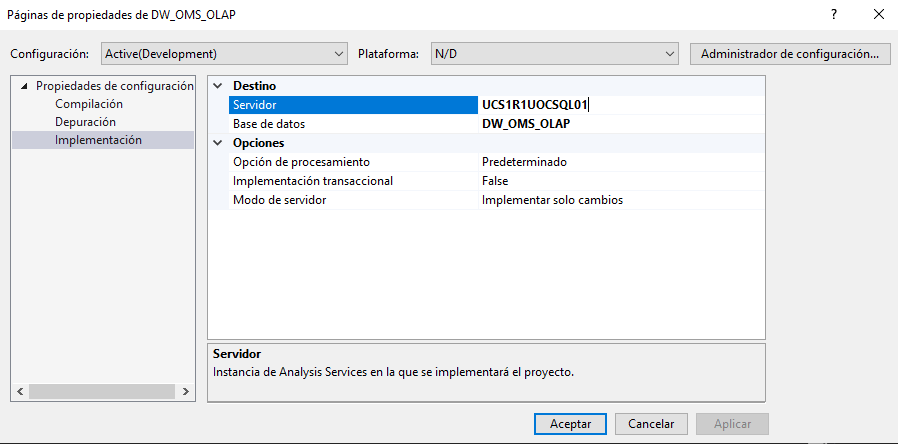
Establecemos los atributos de la taba FACT\_SARAMPION como únicas medidas.

****

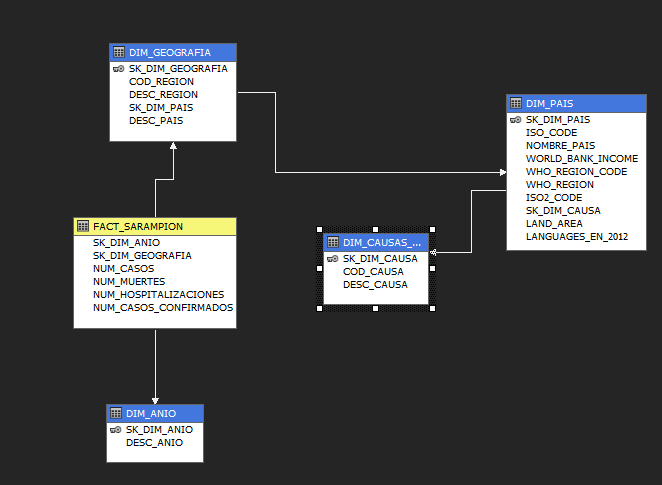
Y seleccionamos todas las dimensiones en el paso para elegir las mismas.

****

Para poder compilar el cubo es necesario cambiar el servidor en el que se implementará, por defecto es localhost. Se cambia en propiedades del proyecto >implementación.

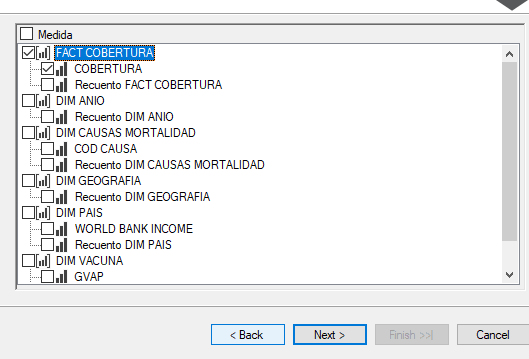
****

Finalmente, el diagrama del cubo es el siguiente.

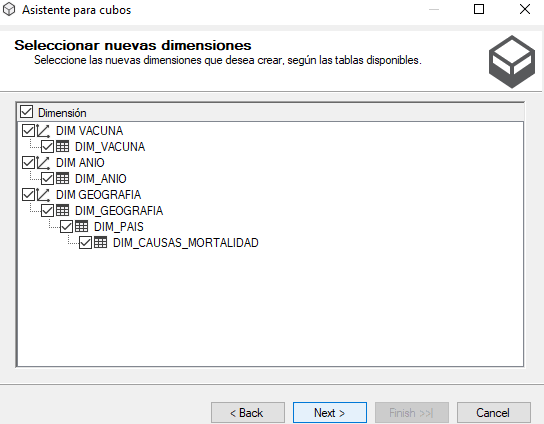
****

**CUBO 2**

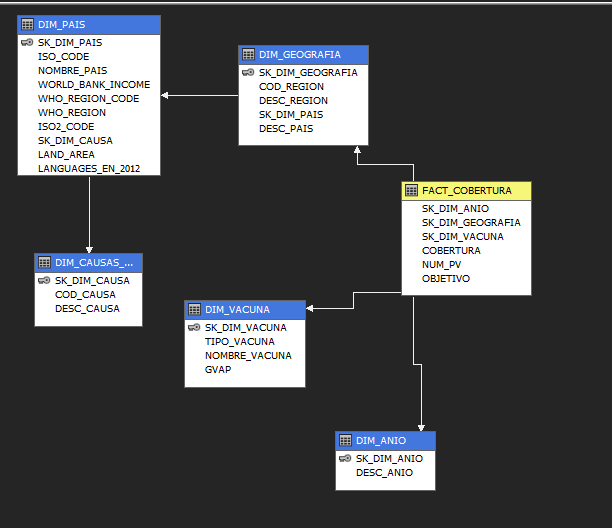
Repetimos el proceso para crear el segundo cubo, el cual está basado en la segunda vista. Seleccionamos cobertura como única métrica para el mismo.

****

De nuevo, seleccionamos todas las dimensiones disponibles.

****

Finalmente, el diagrama del cubo es el siguiente.

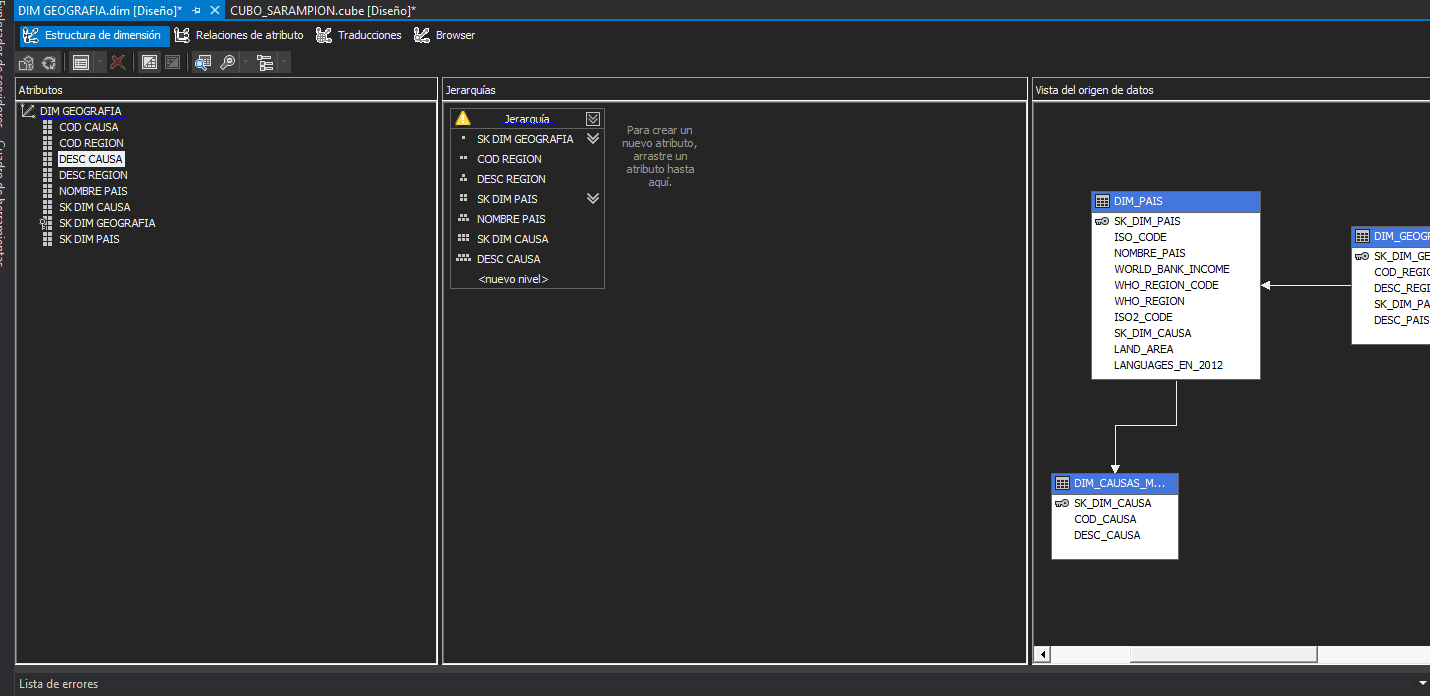
****

1. **JERARQUÍAS Y CUBOS.**

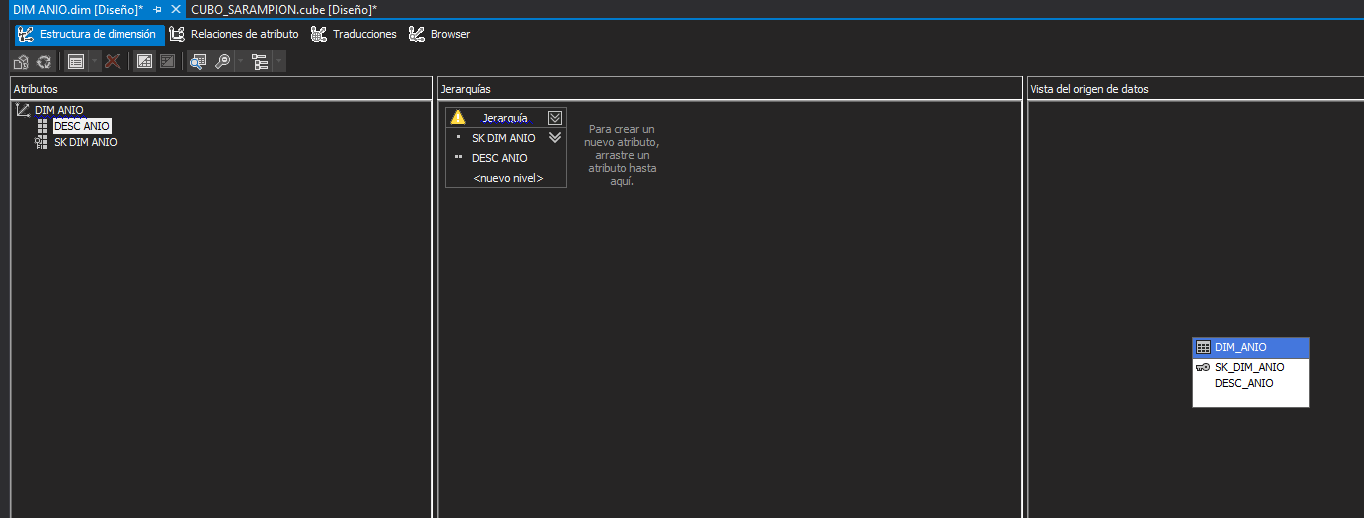
A fin de poder utilizar todos los atributos que deseamos de nuestras dimensiones en las consultas, es necesario implementarlos y generar una jerarquía.

El proceso es sencillo. Pulsando en las dimensiones se abrirán tres paneles contiguos, en el de la izquierda deberemos arrastrar los atributos de las dimensiones que queremos usar en las consultas. Y luego, arrastrando de nuevo estos atributos al panel del centro se definen las jerarquías.

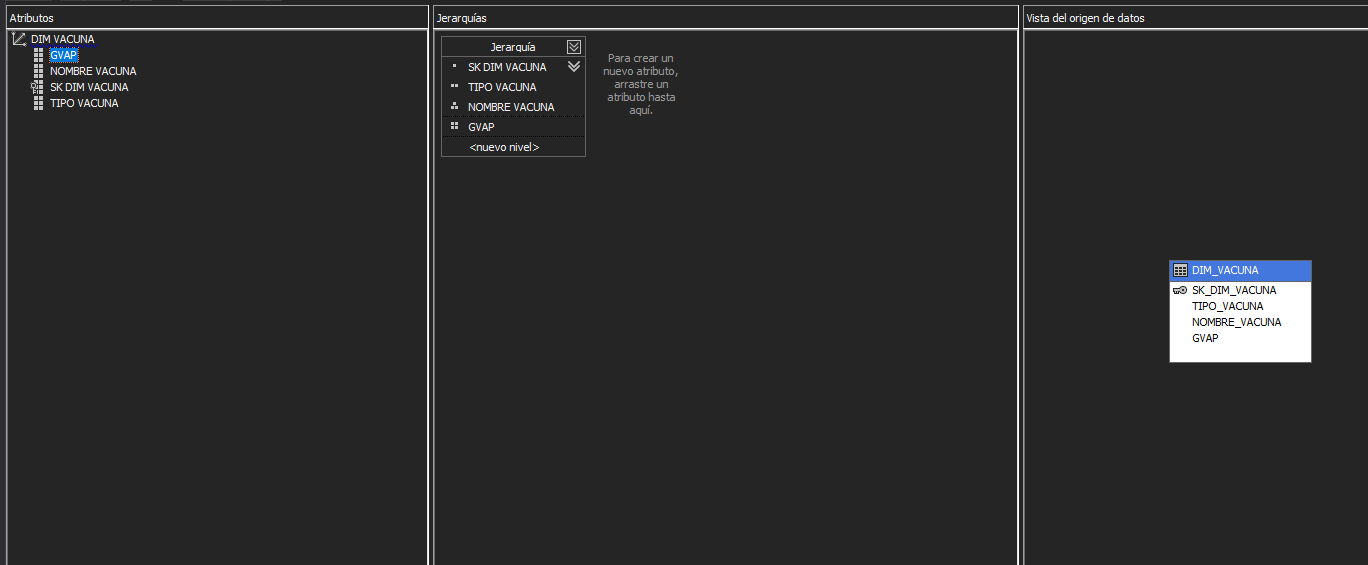
**DIM\_GEOGRAFÍA**

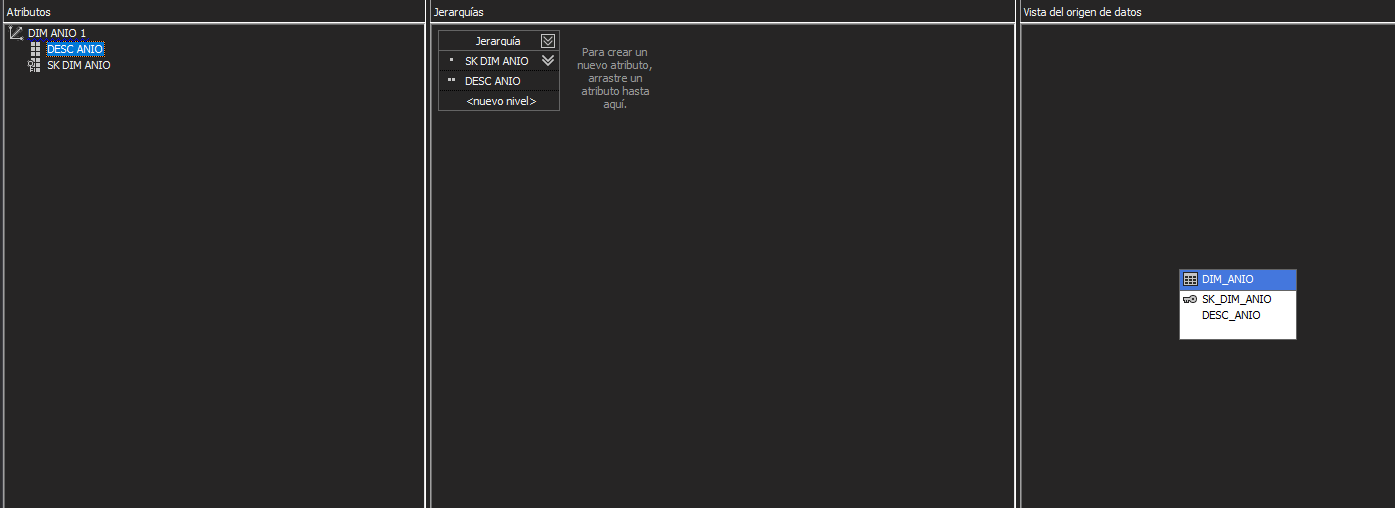
****

**DIM\_ANIO**

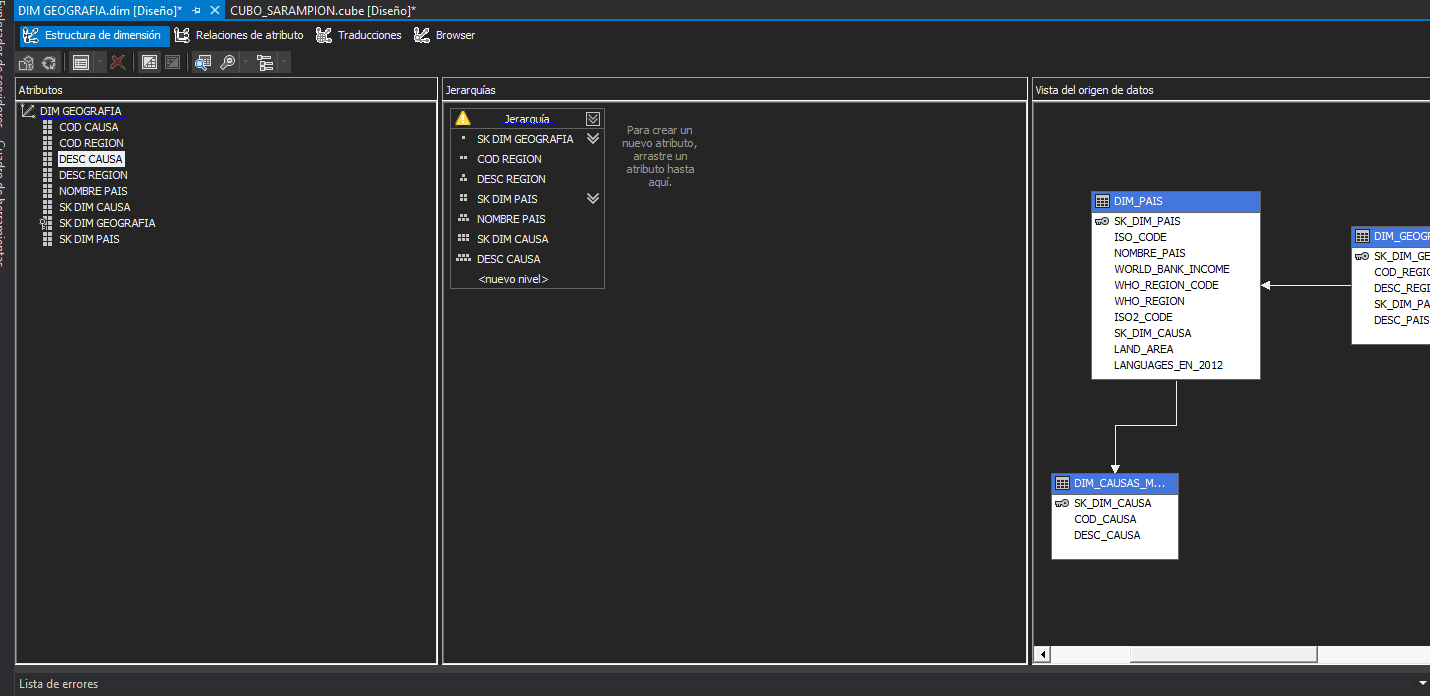
****

**DIM VACUNA**

****

**DIM ANIO 1**

**DIM\_GEOGRAFÍA 1**

****

Tras este paso nuestros cubos ya están listos para la fase de explotación de estos, solo es necesario pulsar en la opción de “implementar”, disponible en el menú desplegable de “Compilar”.

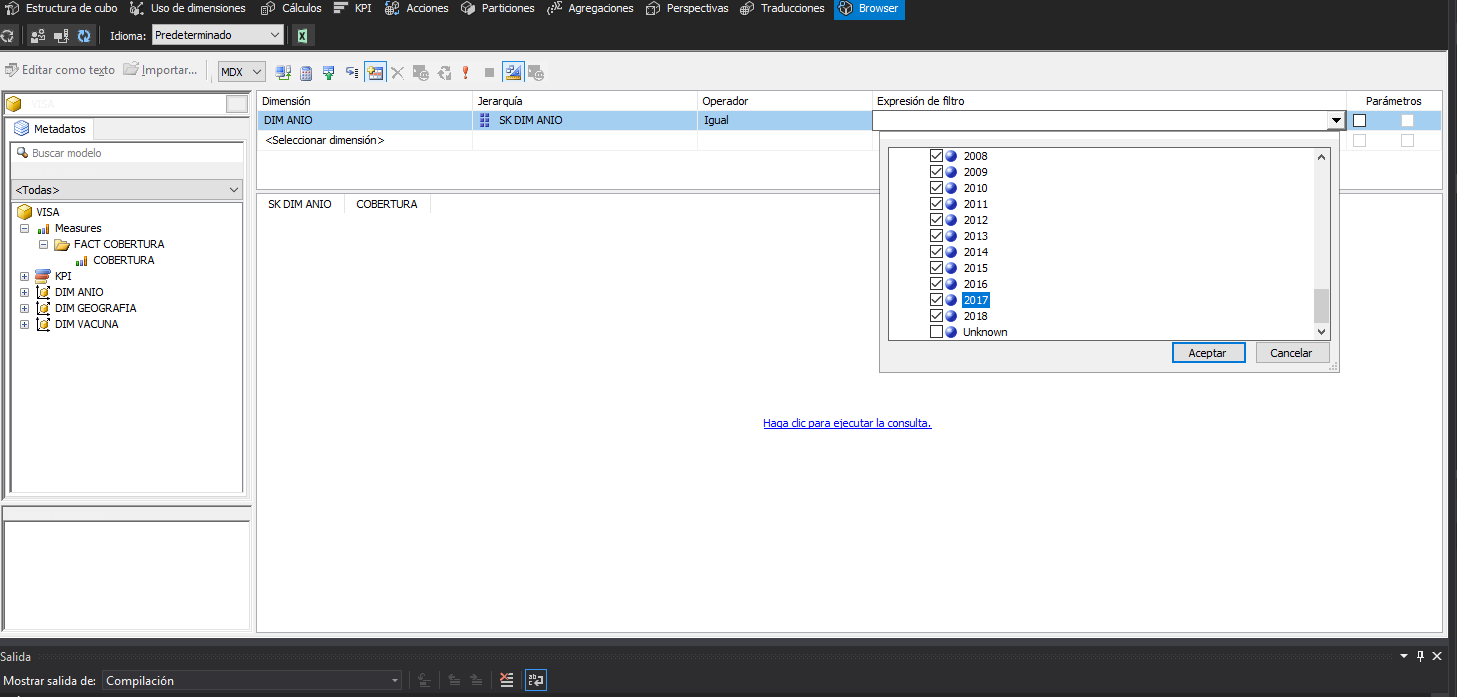
1. **EXPLOTACIÓN DE LOS CUBOS.**

En este apartado se muestran los resultados de las consultas que dan respuesta a las preguntas del enunciado de la práctica.

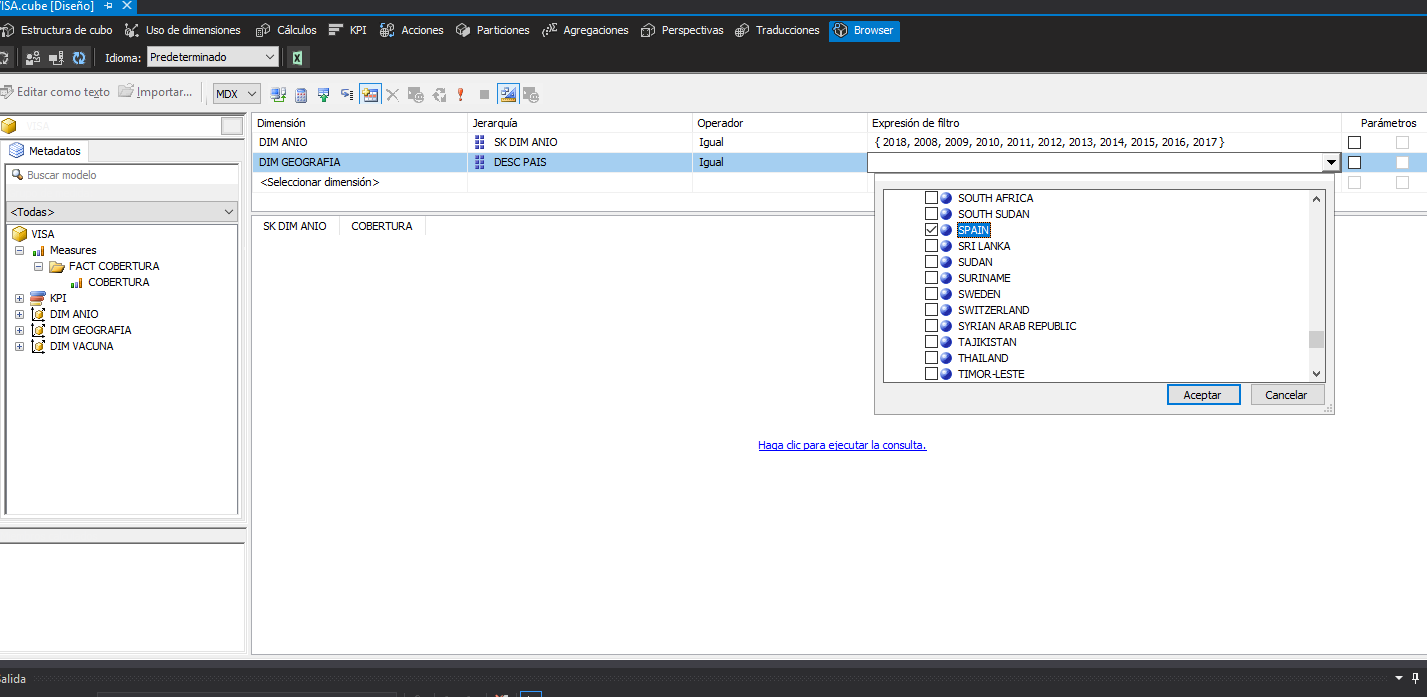
Las consultas se hacen desde el apartado Browser de los cubos.

* **Análisis de la tendencia de los diez últimos años de la cobertura de inmunización de España.**

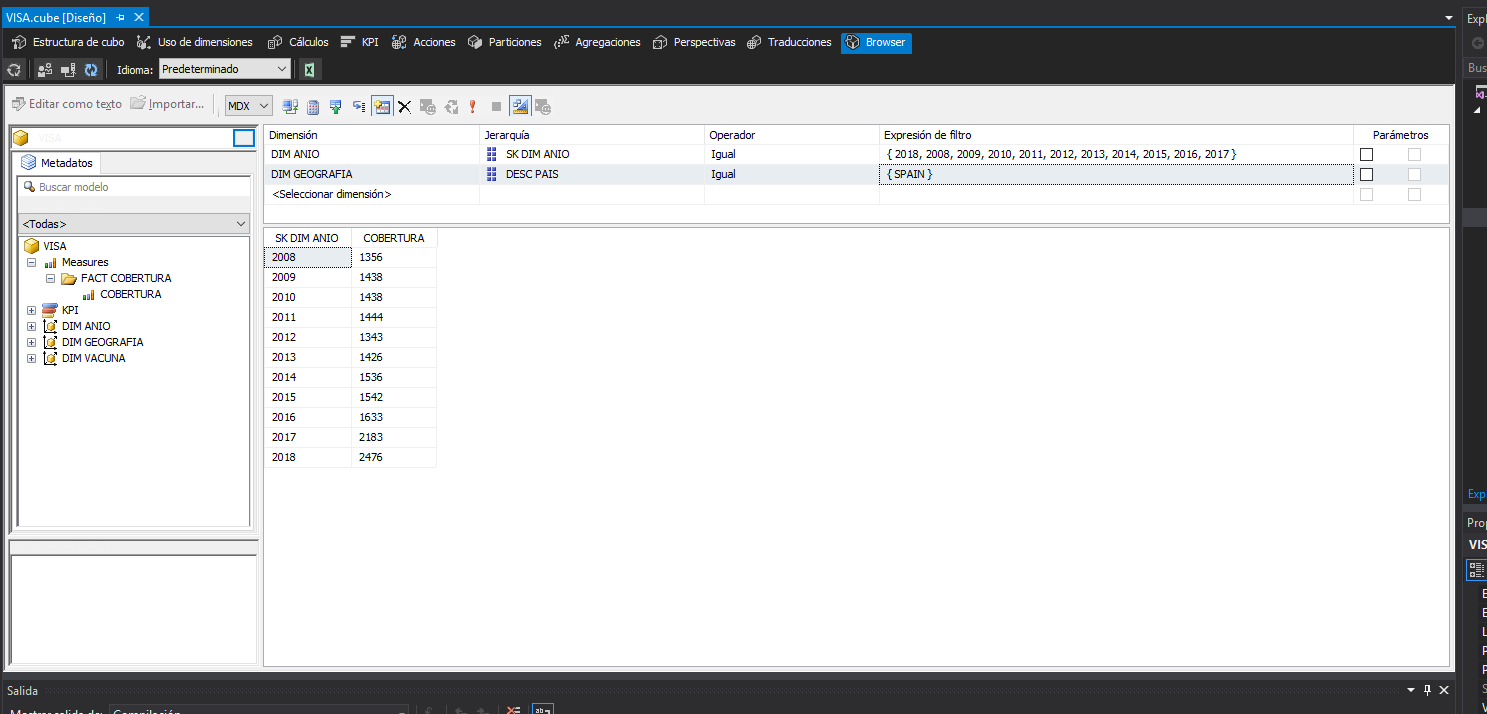
Lo primero es establecer un filtro con solo los diez últimos años.

****

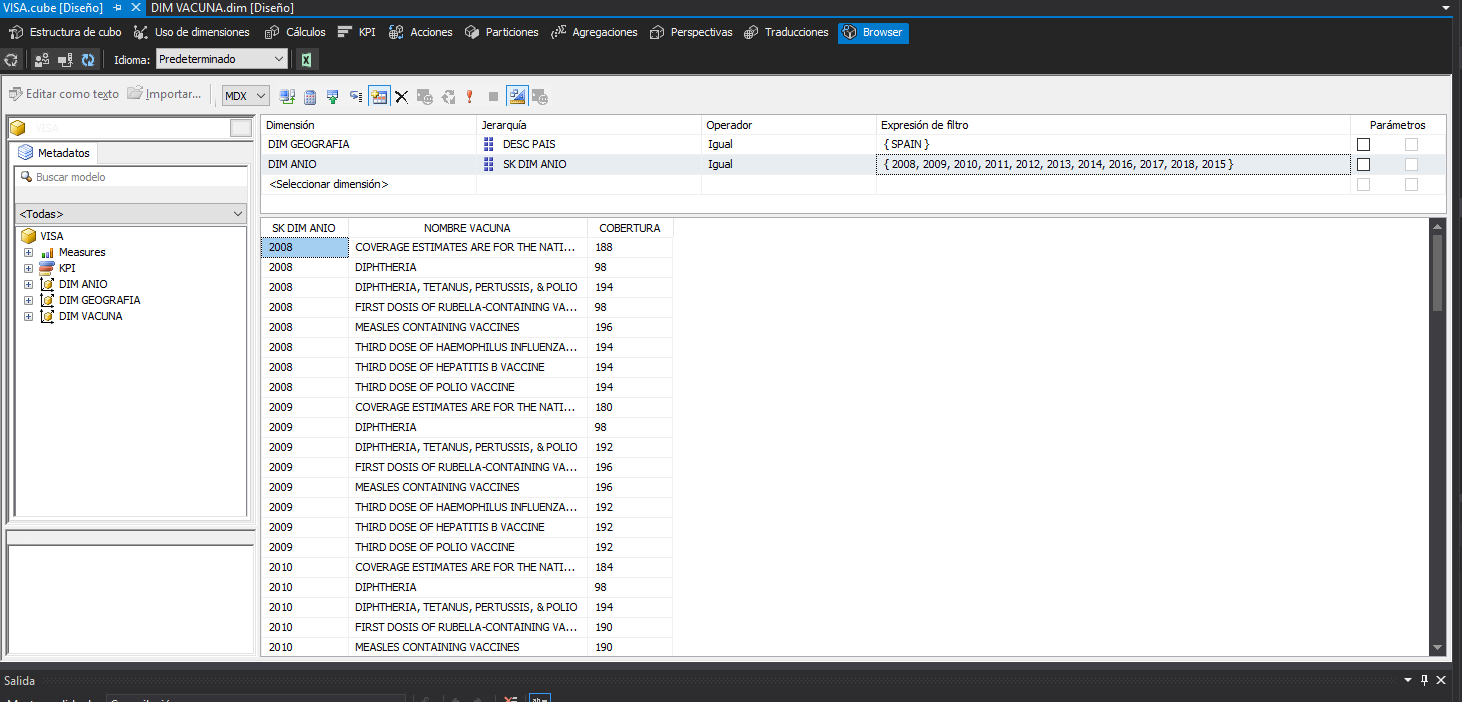
Y otro con España como único país admitido

****

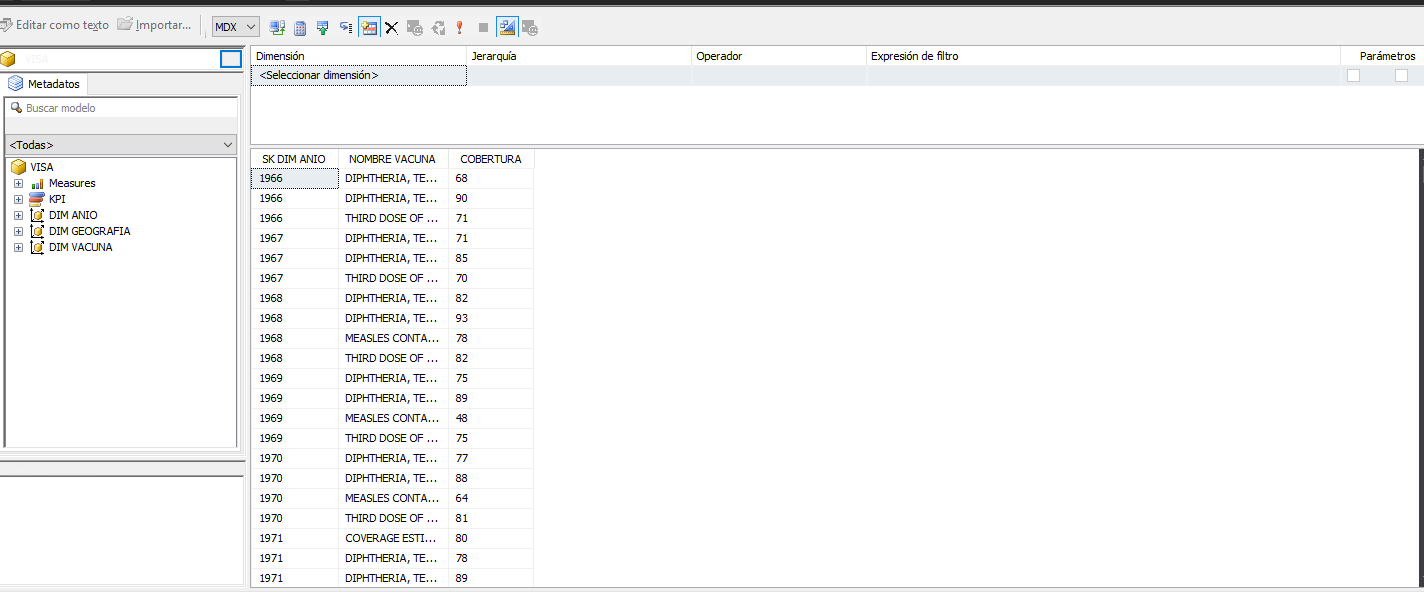
El resultado es el siguiente:

****

Si queremos ver el resultado por vacuna solo hay que añadir dicho campo a la consulta.

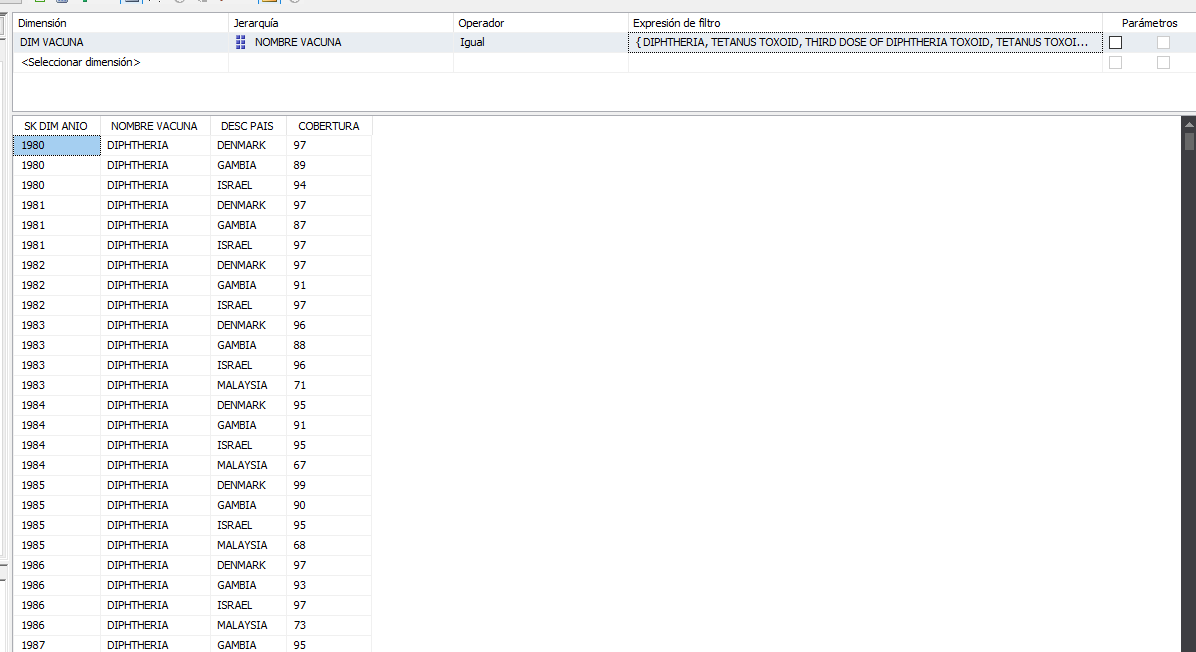
****

**- Análisis evolutivo de la tendencia de la cobertura de inmunización.**

****

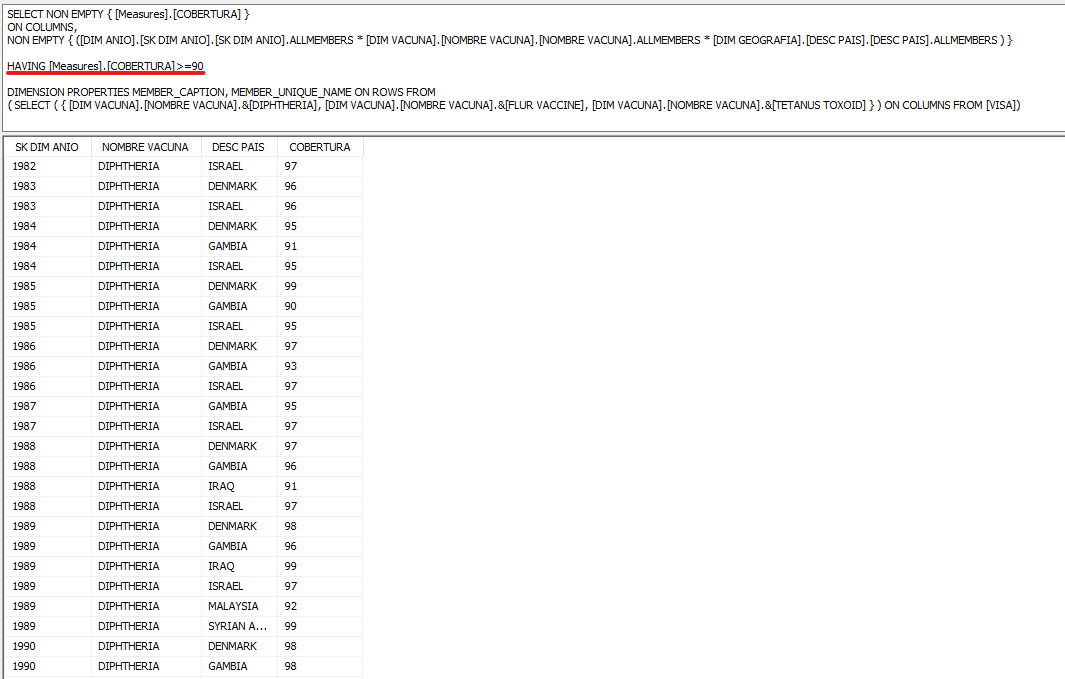
* **Análisis de los países miembros que consiguen el cumplimiento en 2018 del objetivo 1 para la meta 2015 propuesta por el GVAP (Plan de Acción Mundial sobre Vacunas).**

Para el cumplimiento de dicho objetivo es necesario seleccionar las tres vacunas DTP.

****

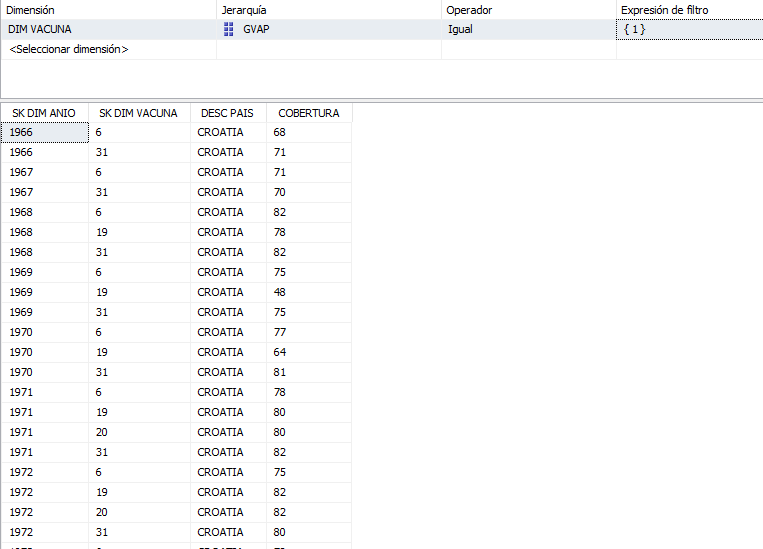
Una de las desventajas de la herramienta de consultas de Visual Studio es que no permite filtrar por medida con la misma sencillez.

Como en este caso tenemos que mostrar solo los casos en los que la cobertura sea mayor a 90%, es necesario acceder al modo diseño y añadir la cláusula “HAVING”. Indicando que solo deseamos seleccionar las coberturas superiores o iguales a 90.

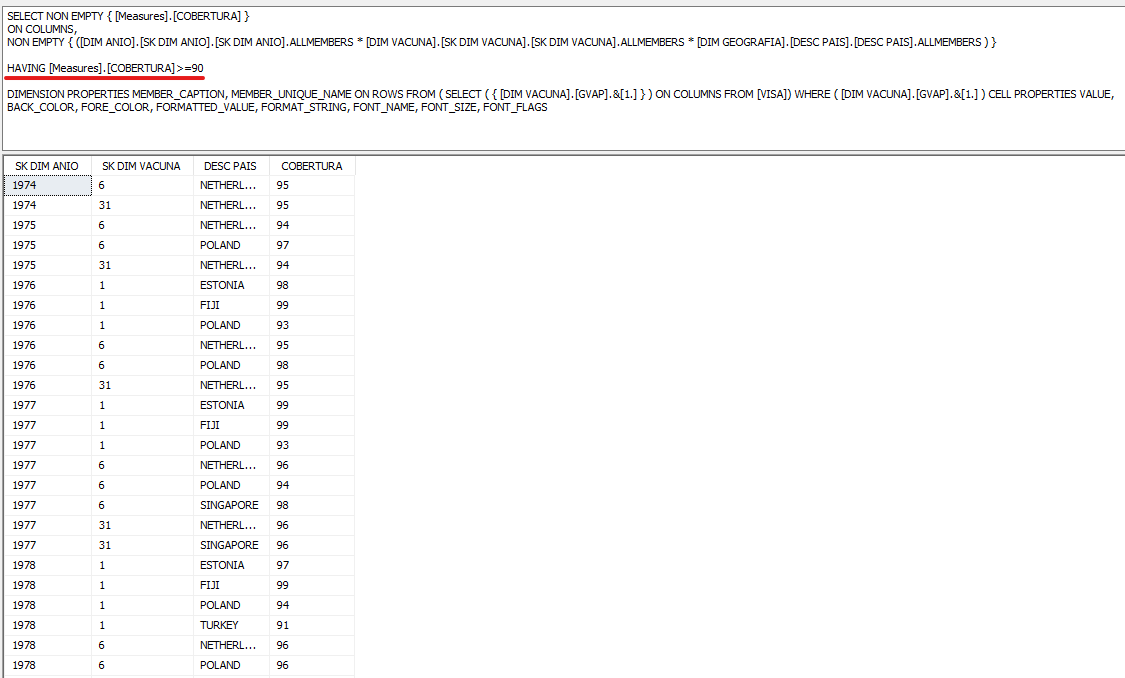
****

* **Ranking de países de la OMS (Organización Mundial de la Salud) que llegan al cumplimiento del objetivo 1 para la meta 2020 propuesta por el GVAP.**

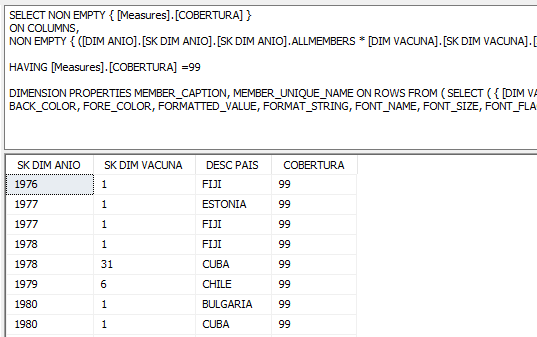
Lo primero es filtrar por vacunas del GVAP.

****

De acuerdo con el objetivo del 2020, dichas vacunas deben superar el 90% de cobertura. Añadimos la cláusula “HAVING”.

****

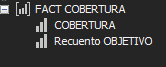
Si se desea ver los países “top“, basta con modificar la cláusula “HAVING” y ponerla a 99%:



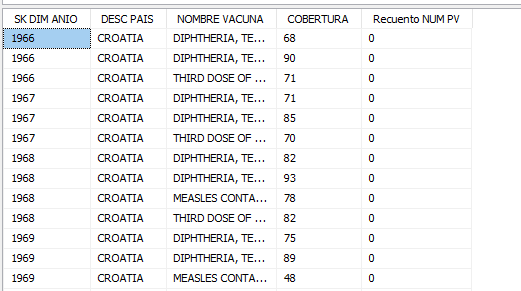
* **Ranking de países con mayor porcentaje de población objetivo vacunada.**

Esta consulta es igual que las anteriores, solo requiere una ligera modificación, si recordamos el apartado de ETL, los datos de la población objetivo tenían el atributo “NUM\_PV” como null. Utilizaremos esto en los filtros.

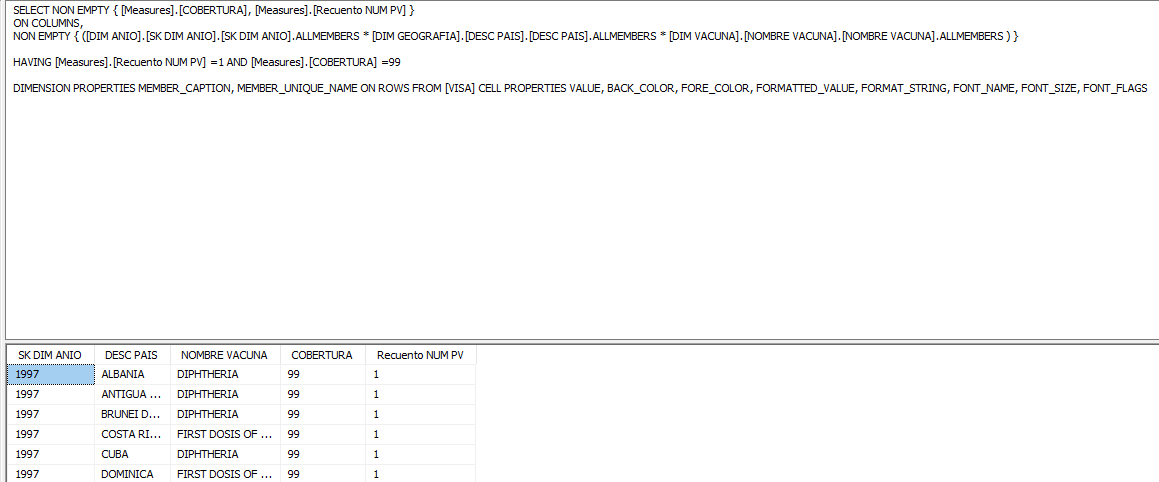
En el diseño del cubo, pulsamos “Nueva medida de columna” haciendo click derecho en el campo.



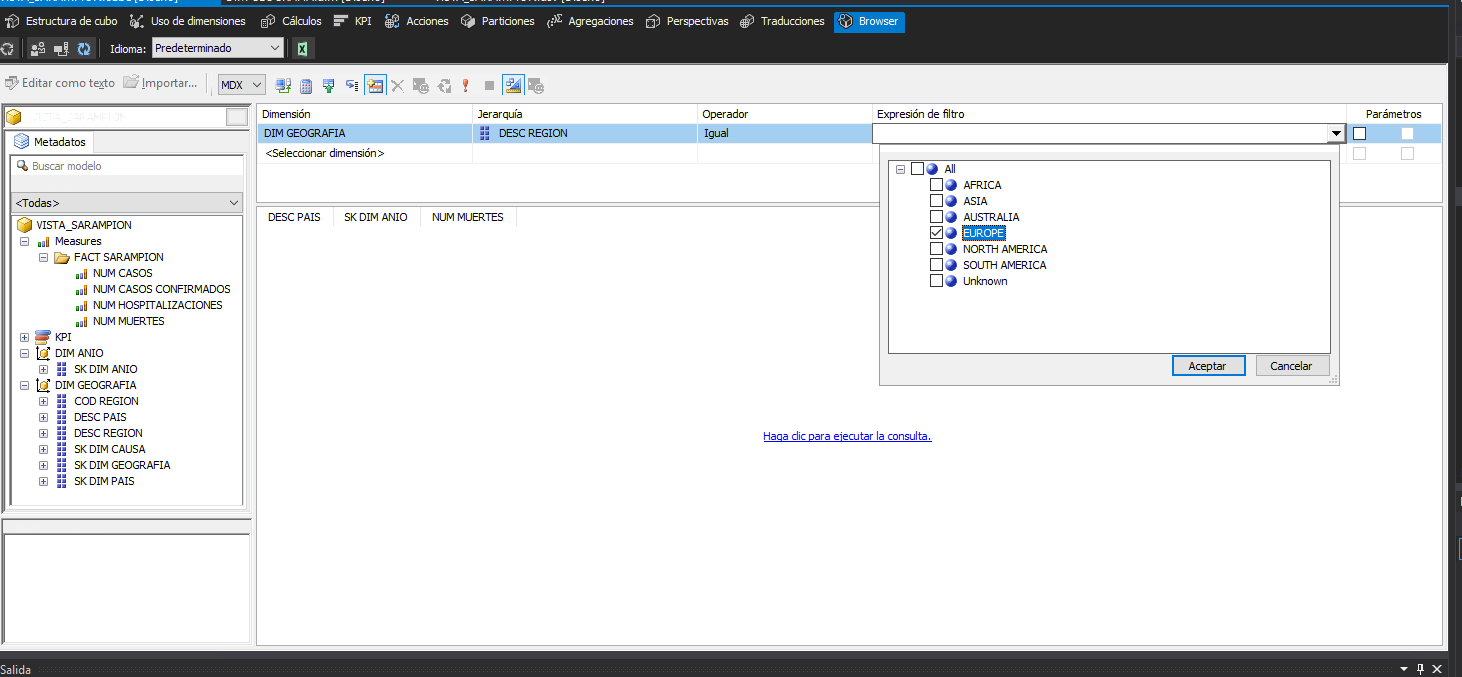
Realizamos la consulta. Cuando este valor no es nulo, su valor es 1.



Filtramos con la cláusula HAVING.

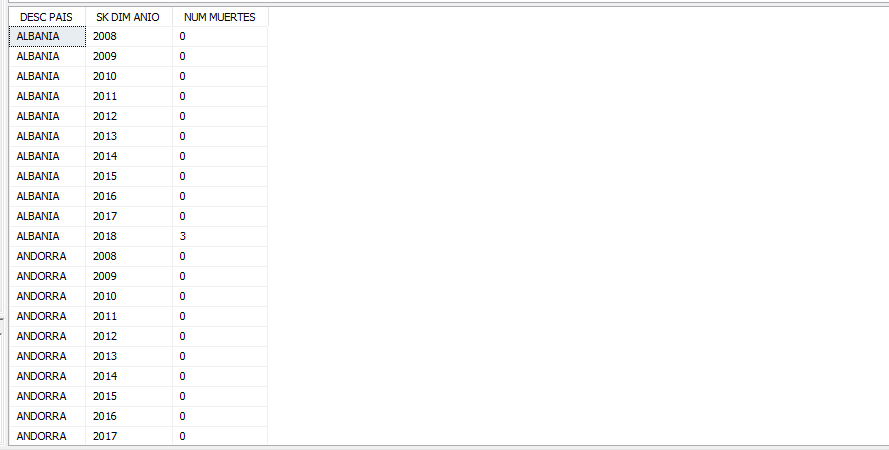


* **Evolución de los casos de muerte por sarampión en los países de Europa.**



En este caso, basta con implementar un filtro de la dimensión geografía, en concreto el campo DESC\_REGION que solo acepte información de la región de Europa.

Para poder mostrar una evolución de los casos de muerte, es necesario arrastrar a la consulta los campos muerte, año y país.



**- Planteamiento de otros análisis que puedan enriquecer el sistema con otras herramientas o visualizaciones que se estimen oportunas.**

Un último análisis interesante podría ser una comparación entre países de distintos continentes respecto a la cobertura de la enfermedad del sarampión.

Por ejemplo, una comparativa entre Uzbekistán y España en 2016:

