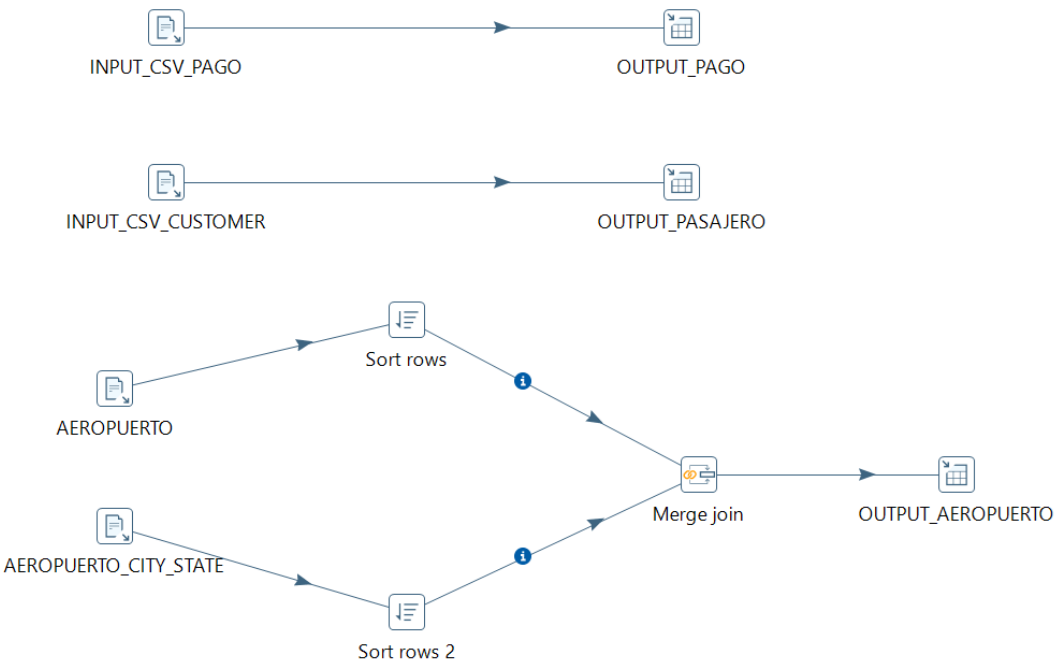
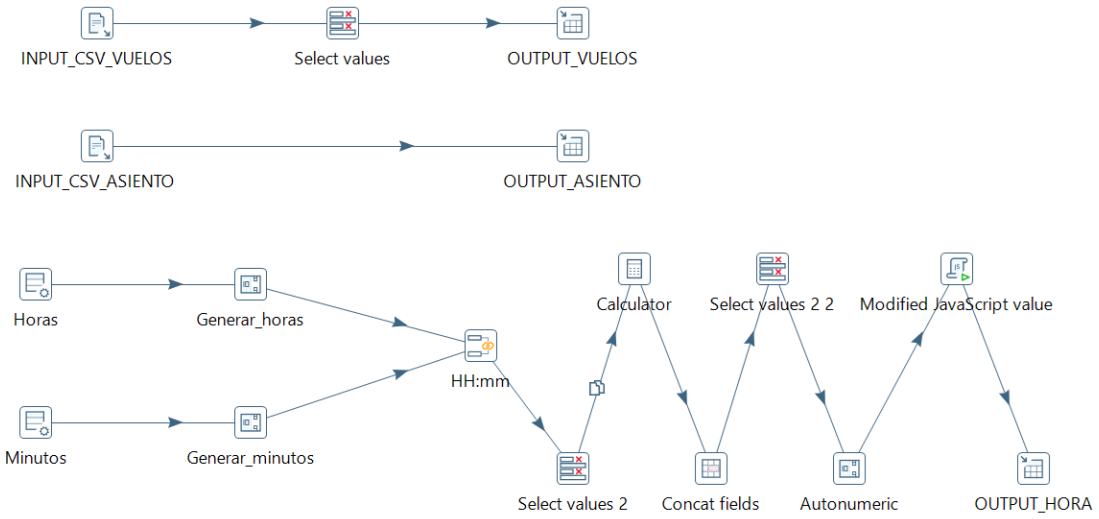
Task 2: Data transformation

Cristian Torres Ortega, Pablo Gamarro Lozano

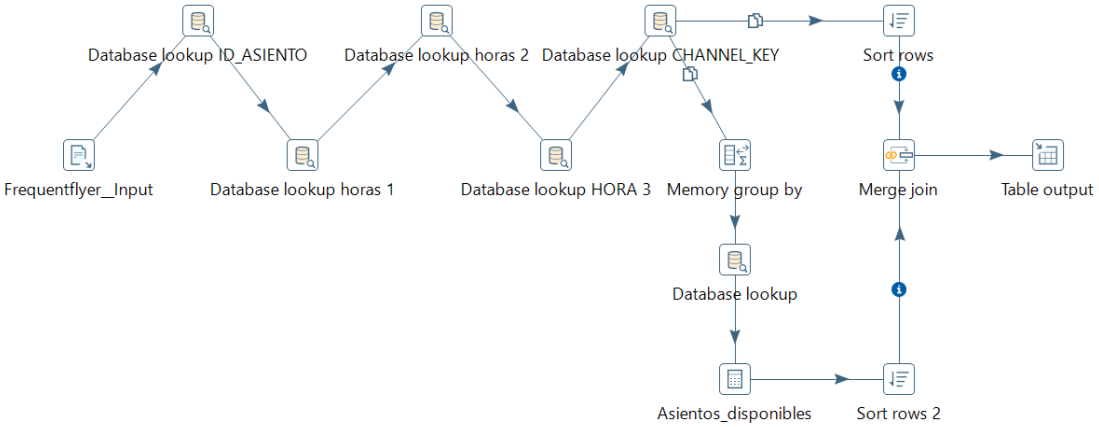
En esta parte final del conjunto de tareas destinadas a diseñar, implementar y cargar datos en un data warehouse, se va a tratar la parte final. Se procederá a la carga de datos en el data warehouse, sin embargo, antes de iniciar este procedimiento se debe de estar completamente seguro de que los datos están de manera correcta para su carga.

A continuación, se muestra el esquema general en la herramienta Pentaho de todas las transformaciones que se han llevado a cabo. El siguiente conjunto de transformaciones corresponden a las dimensiones:



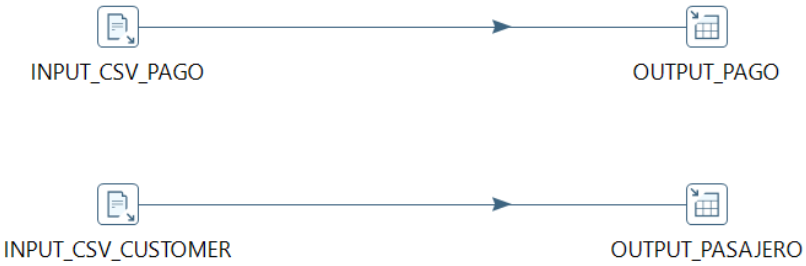


A continuación, se muestra el conjunto de transformaciones para el hecho:

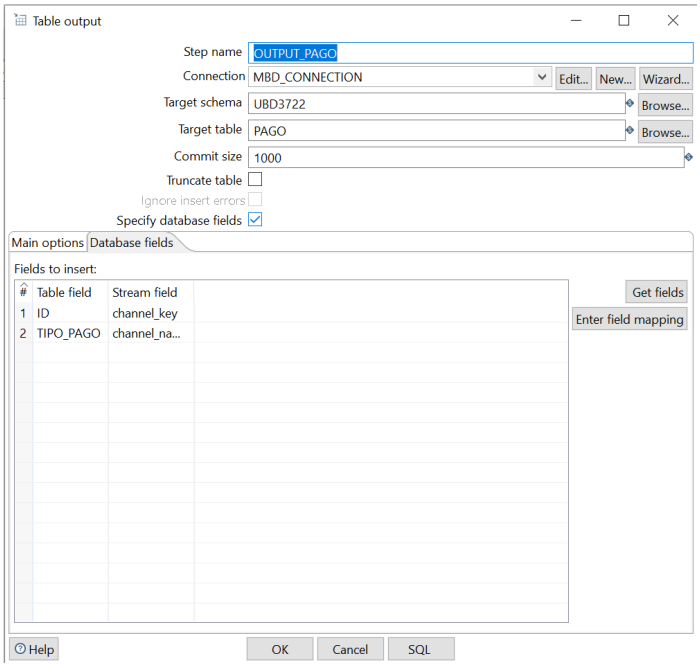


Una vez observado el esquema general de las transformaciones se realiza un análisis más individualizado de estas mismas, explicando el objetivo de cada una, así como la lógica que hay detrás. Como es lógico, todas estas transformaciones comparten un mismo objetivo, que es la carga de unos datos de entrada en unas tablas de salidas que se encuentran en una base de datos. Para cumplir con este objetivo, se llevarán a cabo diferentes etapas dentro de cada transformación.

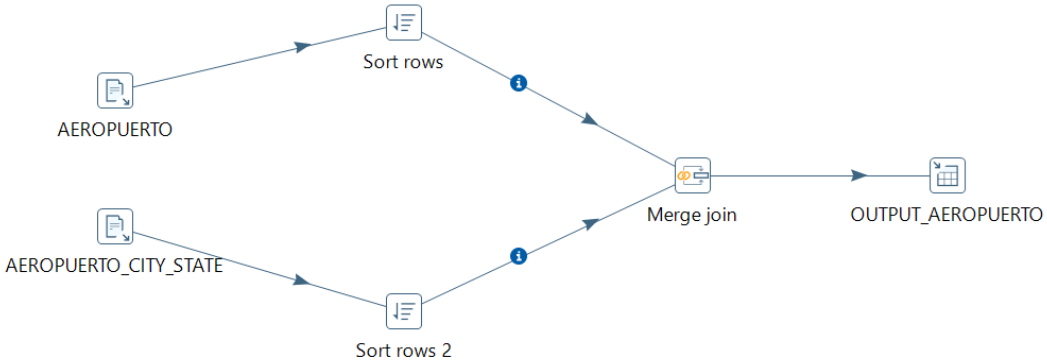
En primer lugar, se observa las siguientes dos transformaciones:



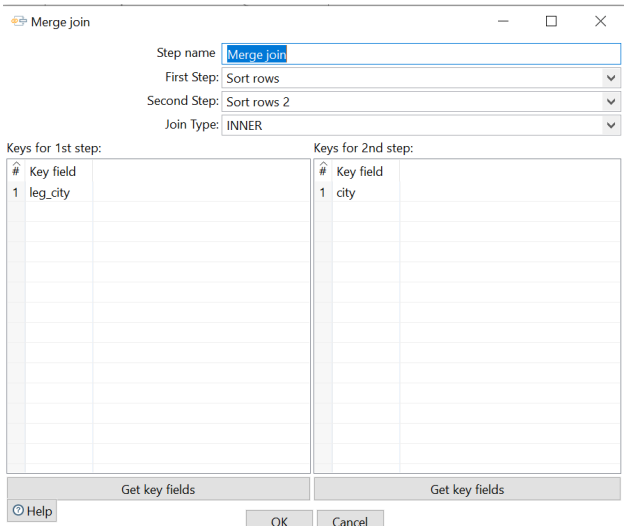
Estas transformaciones tienen como meta la carga directa de los datos en las tablas objetivos de la base de datos de Oracle que se creó anteriormente, en los ejercicios previos. Se trata de un proceso sencillo, donde en primer lugar se han cargado como inputs los “csv” de las tablas pago y customer, y luego se han enlazado a sus correspondientes salidas de tablas de la base de datos Oracle, como se observa en el siguiente ejemplo:



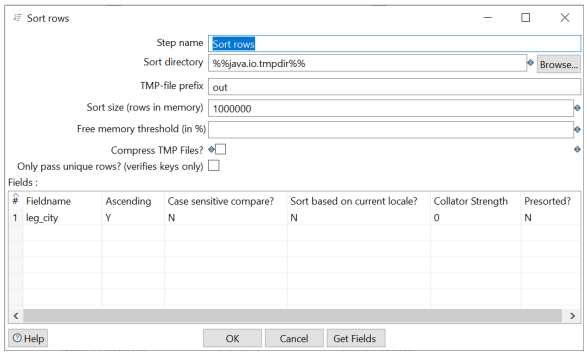
Cabe indicar, que el proceso es tan simple y directo, porque los datos de los “csv” ya vienen estructurados de forma correcta para la carga. En contraste, no sucede lo mismo a la hora de cargar los datos en la tabla aeropuerto:

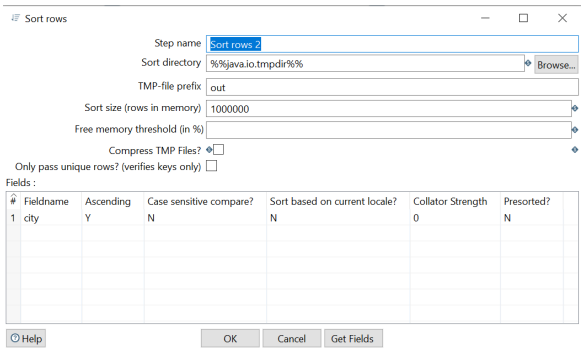


La problemática que se encuentra a la hora de realizar la carga en la tabla aeropuerto, es que el "csv" llamado "airport" contiene todas las columnas necesarias para la carga excepto la columna "state", que se encuentra en el "csv" llamado "airport_city_state", por ello, se debe realizar una etapa de unión denominada "merge join" para unir esta columna faltante a las demás, obteniendo una tabla con todas las columnas necesarias para la carga correcta de los datos:

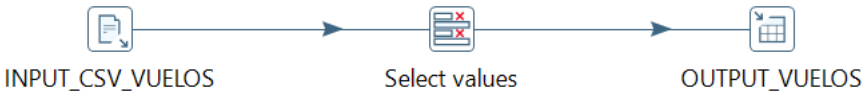


Como se observa, la unión en la etapa "merge join" se está realizando a través de una columna en común en ambas tablas, que es la columna con el nombre de la ciudad. Cabe añadir, que para que dicha unión sea correcta y se evite cualquier tipo de error en la carga, es altamente recomendable realizar a cada tabla una ordenación ascendente de la columna por la que se quiere unir ambas tablas, como se ha realizado en la transformación. A continuación, se muestra ambas ordenaciones:

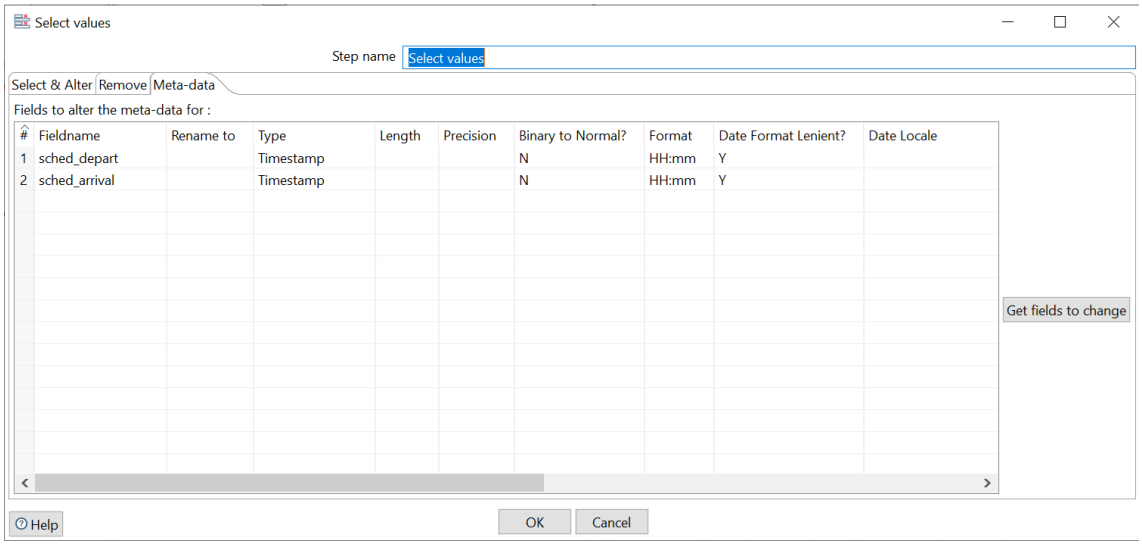




La siguiente transformación a tratar es:



En este caso, se podría hacer una carga directa de los datos, sin etapas intermedias, si no fuera porque en la tabla de destino en la base de datos, las horas de llegadas y salidas están en formato “timestamp” mientras que en el “csv” estas horas se encuentran en formato de “string”. Para solucionar esto, se ha introducido una etapa intermedia denominada “select values”, cuya función es la transformación del tipo de datos de estas celdas, es decir, ayudará a pasar de “string” a “timestamp”:

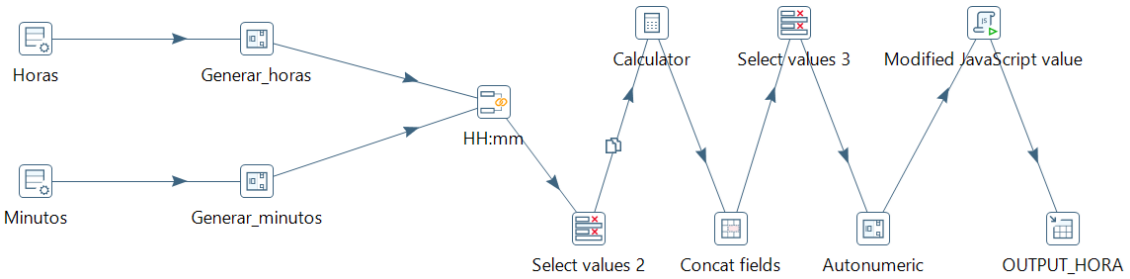


Como se observa, también se la ha introducido el formato que tienen las horas en el input. Además, se tiene habilitada la opción “Date Format Leniente”, esto posibilita que si algún dato

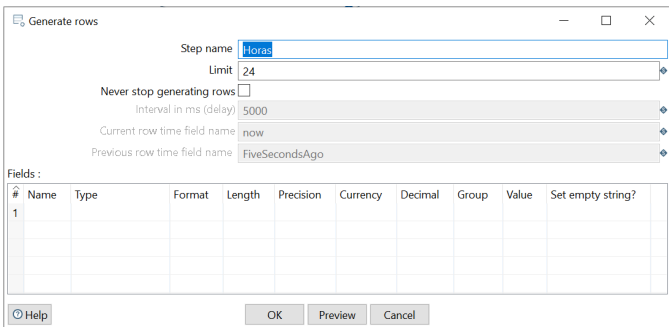
viene de forma diferente, Pentaho mediante determinadas herramientas internas, pueda procesarlo en el formato que se le ha indicado.

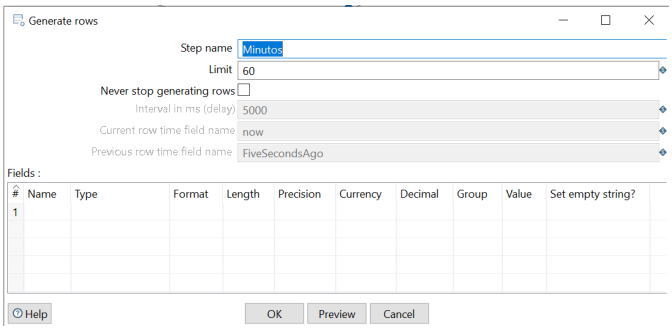
La siguiente transformación es:

En este caso, se trata de una transformación muy simple, cuya lógica es la misma que en las dos primeras transformaciones. Por lo tanto, se procede a explicar la siguiente transformación:

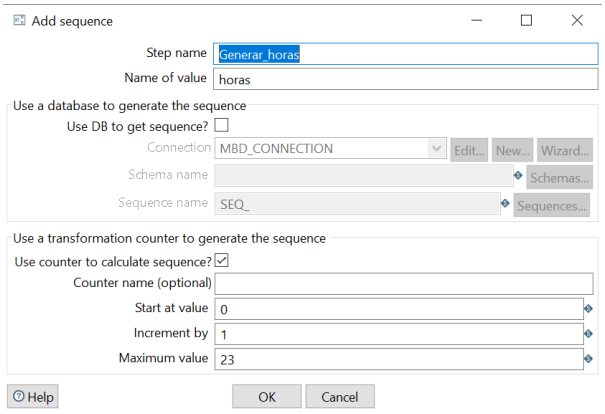


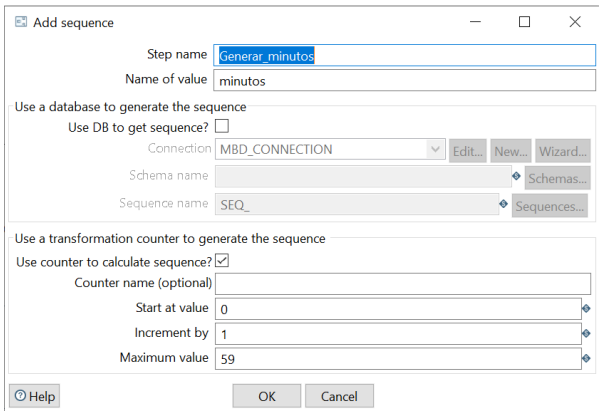
Como se muestra a simple vista en la imagen anterior, se trata de una transformación con múltiples etapas cuyo objetivo es poder cargar en la tabla “HORA” de la base de datos todas las horas en formato HH:mm que puede tener un día. Para ello, en primer lugar, se han utilizado dos estapas, “Horas” y “Minutos”, del tipo “Generate Rows”, con la finalidad de crear dos tablas con el número de filas necesarias para almacenar las horas y minutos generados en la etapa posterior:



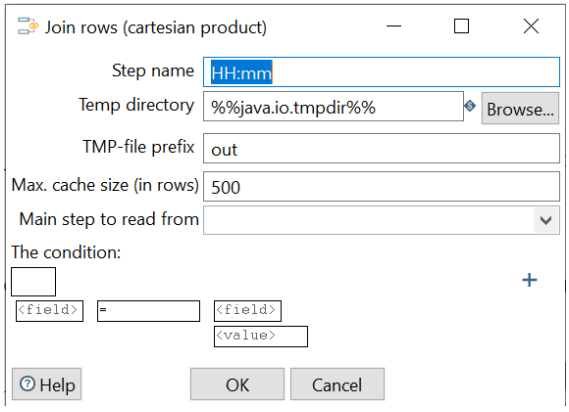


Como se observa, se ha colocado un límite de 24 filas en “Horas” y de 60 filas en “Minutos”. La siguiente etapa consiste en generar por un lado las horas, y por otro lado los minutos utilizando transformaciones del tipo “Add sequence”:

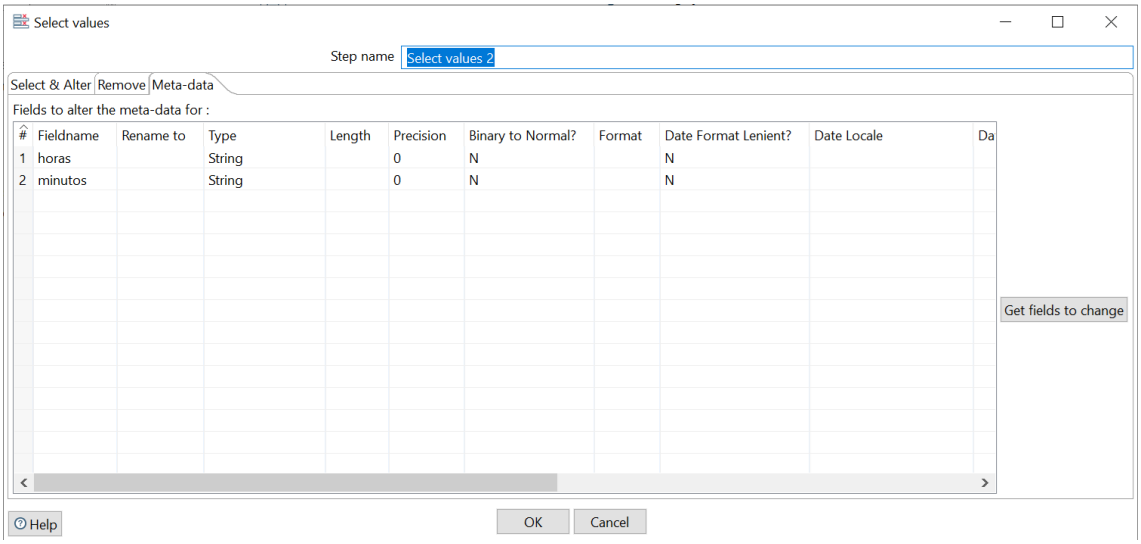




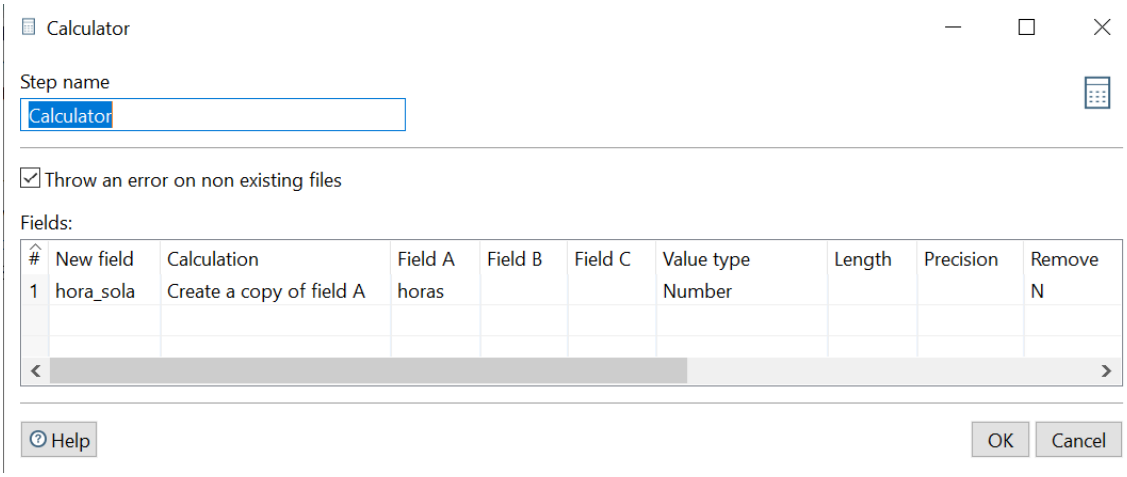
Se ha utilizado en sistema horario de 24 horas. Por lo tanto se ha obtenido por un lado una columna con horas que van de los números 0 al 23, y otra columna con minutos que van de los números 1 al 59. La siguiente etapa, consiste en la combinación de estas dos columnas, utilizando la etapa de transformación de Pentaho denominada “Join rows (cartesian producto)”. Con esta etapa se realizará un producto cartesiano entre ambas filas, lo cual generará todas las combinaciones posibles, que es justo lo que se quiere lograr:



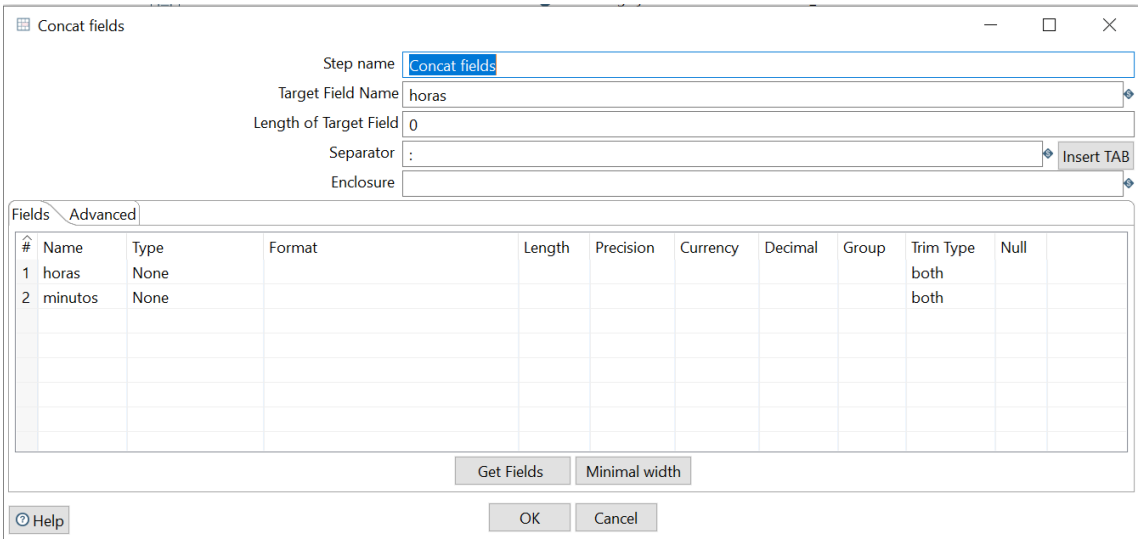
A continuación, se necesita determinar qué tipo de datos son estas horas y minutos, donde en primera instancia, serán tipos “string” para facilitar las siguientes etapas de transformación que se van a ver(concatenación). Para obtener las horas y minutos en tipo “string”, se utiliza la etapa “select values” de la misma forma que vimos en transformaciones anteriores:



Como en la tabla destino tenemos un campo denominado “hora_sola” de tipo “number”, en la siguiente etapa de transformación se creará una nueva columna, que será una copia de la columna hora, utilizando la etapa calculator:

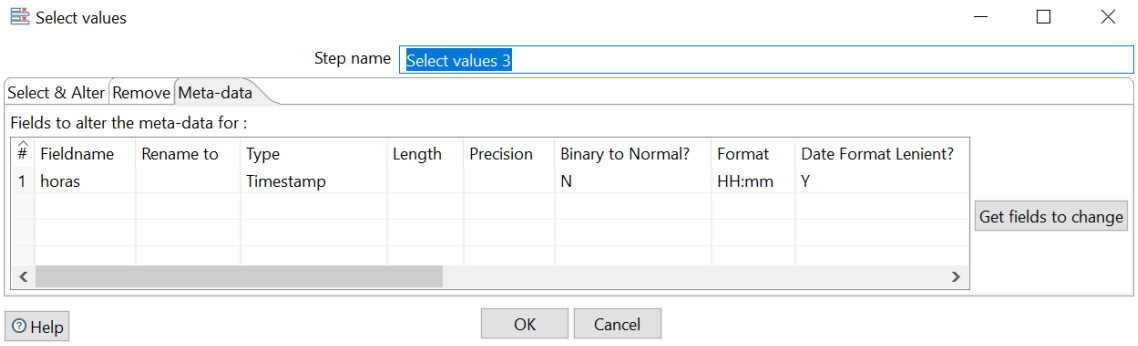


Una vez se tiene la nueva columna, la siguiente etapa consiste en una concatenación para conseguir tener en una misma columna las horas en formato “HH:mm”:

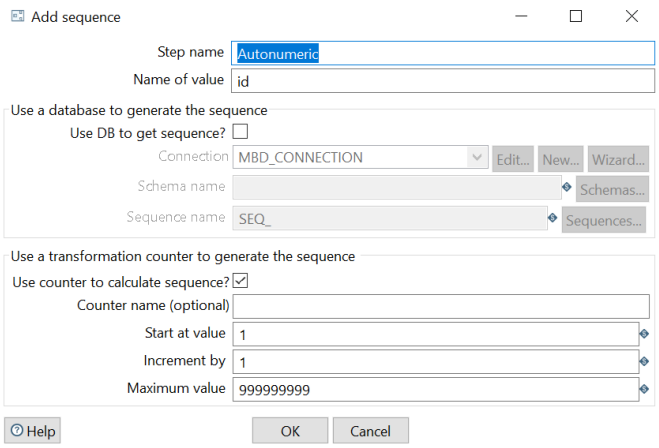


Cabe señalar, que se ha utilizado como separador ":" para lograr el formato deseado, y que, "Trim Type" hace referencia a la eliminación de espacios en blanco, y en este caso, se ha seleccionado que se eliminen los espacios en blanco por ambos lados.

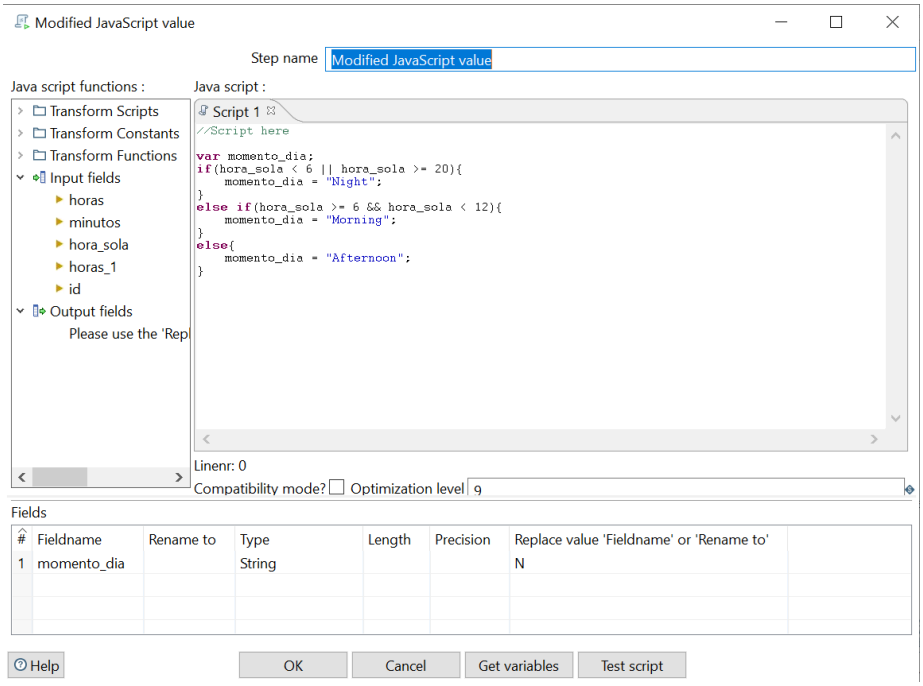
Después de esto, la siguiente etapa es pasar estas horas a formato "Timestamp", utilizando una etapa de "select values", como se ha realizado en pasos anteriores:



De esta forma se obtienen las horas generadas y en el formato requerido, sin embargo, para cargarlas se necesita de la generación de una clave primaria, la cual se generará mediante una etapa de "add sequence":



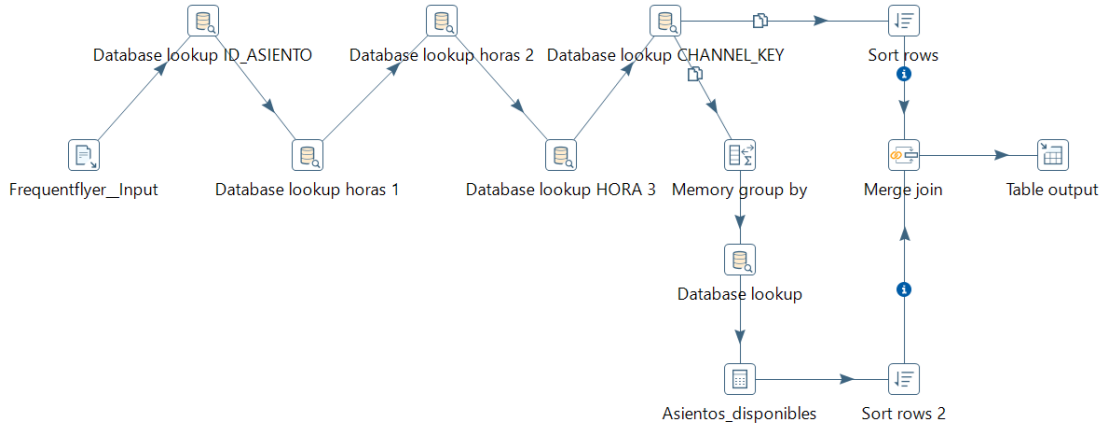
La penúltima etapa que queda por explicar es la generación de la columna “momento_dia” que se encuentra en la base de datos. Esta es la última columna a generar antes de proceder a la carga de lo datos. Para realizar esta carga de datos, se utiliza una etapa de Pentaho denominada “Modified JavaScript value”:



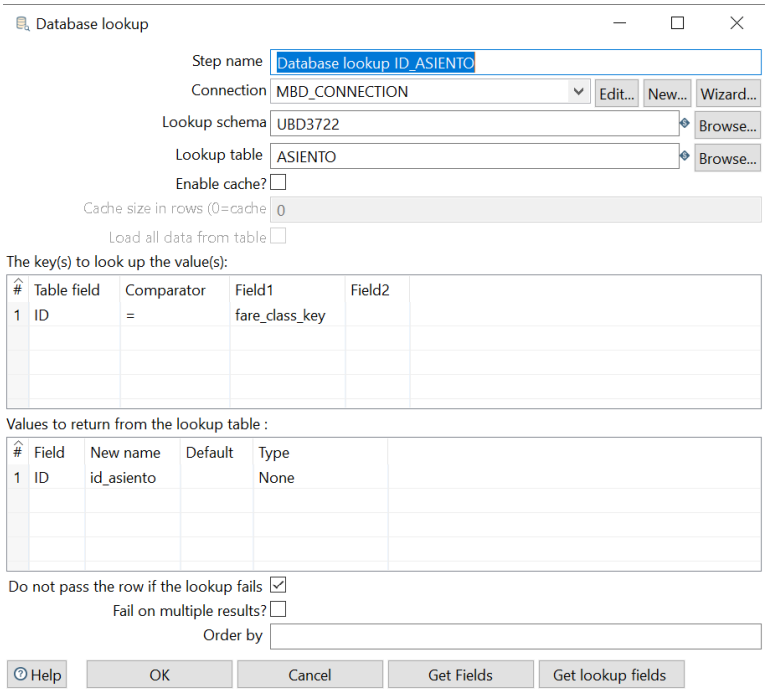
En esta etapa, se ejecuta un script que realiza transformaciones en el flujo de datos de entrada, dando lugar a un flujo de datos de salida. En este caso, como se observa en el script, se crea la variable “momento_día”. Este campo es completado con los valores “Night”, “Morning” o “Afternoon”, según el rango de valores numéricos en el que se encuentre el campo “hora_sola” que se había generado con anterioridad. Esta comprobación o cribado se realiza gracias al uso de condicionales. En la parte baja de esta ventana, se observa cómo se establece que esta etapa devuelva el campo creado “momento_dia” en formato “String”.

Por último, solo queda utilizar la etapa “Table output” de Pentaho para cargar todos los datos en la tabla de “hora” de la base de datos. Ahora sí, se procede a ejecutar las transformaciones en Pentaho para cargar todos los datos de las dimensiones en la base de datos, y así tener estas tablas preparadas para la posterior carga de los datos en la tabla de hecho tras las transformaciones pertinentes.

Hasta este punto, se tienen todas las transformaciones referentes a las dimensiones, a continuación, se va a proceder a explicar en detalle la transformación del hecho:

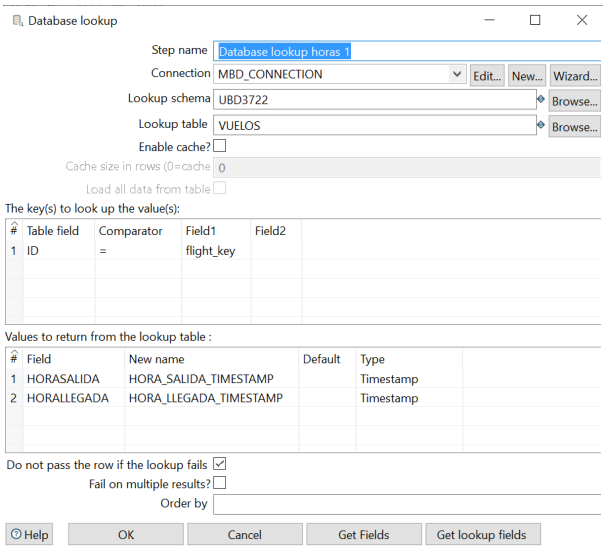


Como se observa, se parte de una etapa inicial, donde se carga el “csv” denominado “frequentflyer”. Luego se realiza una etapa “Database lookup”. Este tipo de etapas traen a la tabla de entrada los campos de la entrada de referencia donde se busca, extrayendo solamente los datos donde las columnas de valores elegidas de ambas tablas coincidan. En este caso, el lookup es el siguiente:

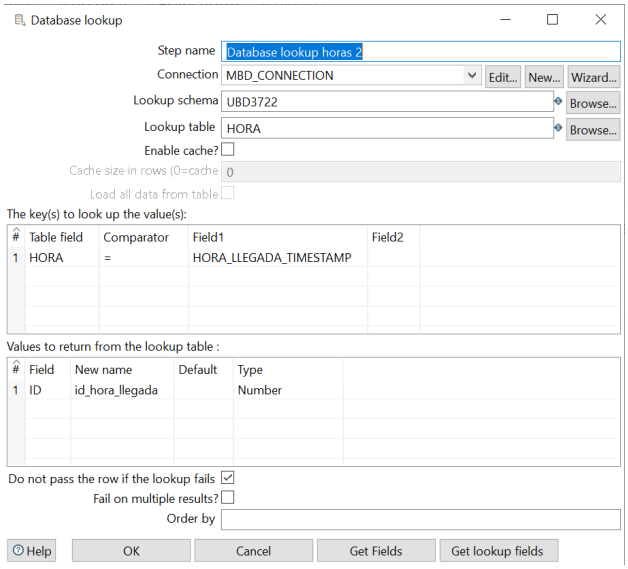


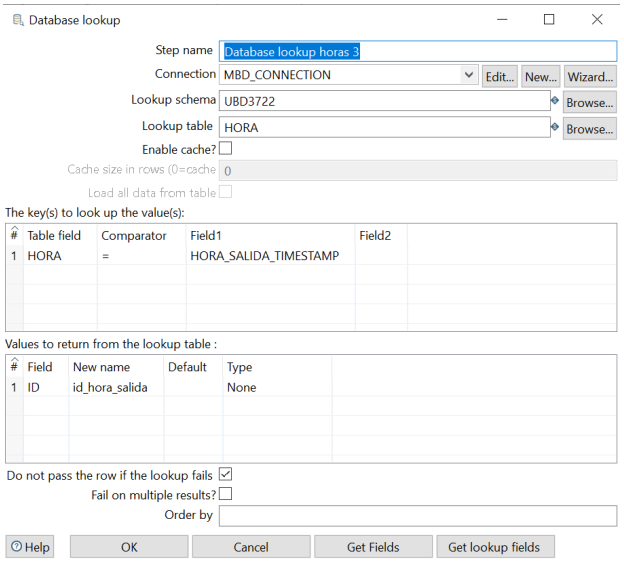
Aquí se observa, que se está buscando traer a la tabla de entrada aquellos valores de la tabla ASIENTO de la base de datos, donde los campos “ID” de la tabla ASIENTO y los campos “fare_class_key” de la tabla de entrada coincidan. El objetivo detrás de este movimiento es añadir a los datos de entrada la columna id_asiento, el cual es el campo retornado por el “Database lookup” y necesario para cargar los datos en la tabla hecho de la base de datos.

Otras columnas que se necesitan añadir a los datos de entradas son “id_hora_salida” y “id_hora_llegada”. Para lograrlo, primero se añaden estas variables en formato “timestamp”, mediante un “Database lookup” a la tabla de referencia “VUELOS” de la base de datos, donde han de coincidir las columnas de ambas tablas que hacen referencia a las claves de vuelo:

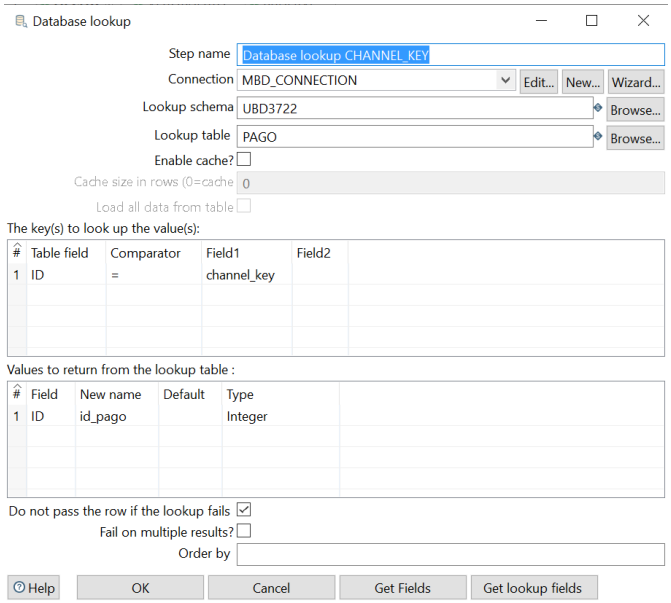


Una vez se han retornado los dos campos desde la tabla de referencia, se deben realizar dos etapas más de “Database lookup”, una para cada campo, tomando como tabla de referencia en la base de datos la tabla “HORA”, y haciendo coincidir los campos de ambas tablas que contienen las horas en “timestamp”, retornando el valor de “ID” de la tabla “VUELOS”. De este modo, las dos etapas de “Database lookup” son las siguientes:

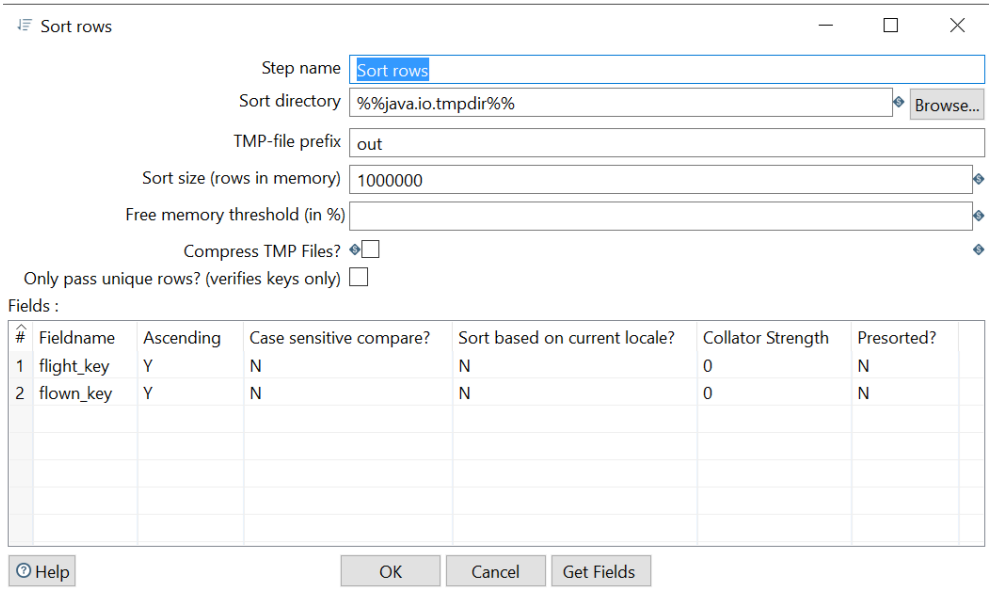




La siguiente etapa es también de tipo “Database lookup”, donde se busca un filtrado de datos como también sucede en las etapas anteriores, donde al tener marcada la opción “Do not pass the row if the lookup fails” se eliminan aquellas filas cuyo campo se está comparando tiene valores que no se encuentran dentro de las restricciones establecidas. Normalmente, cuando existe un gran volumen de datos, se debe realizar un análisis exploratorio de datos para averiguar a que campos se debe de realizar este filtrado, pero en este caso como se tratan de pocos datos y fácilmente analizables a simple vista, se puede obviar este paso. Por lo tanto, este lookup se realiza sobre la tabla pago obtener los “id_pago” correctamente filtrados, por lo que queda de la siguiente forma:

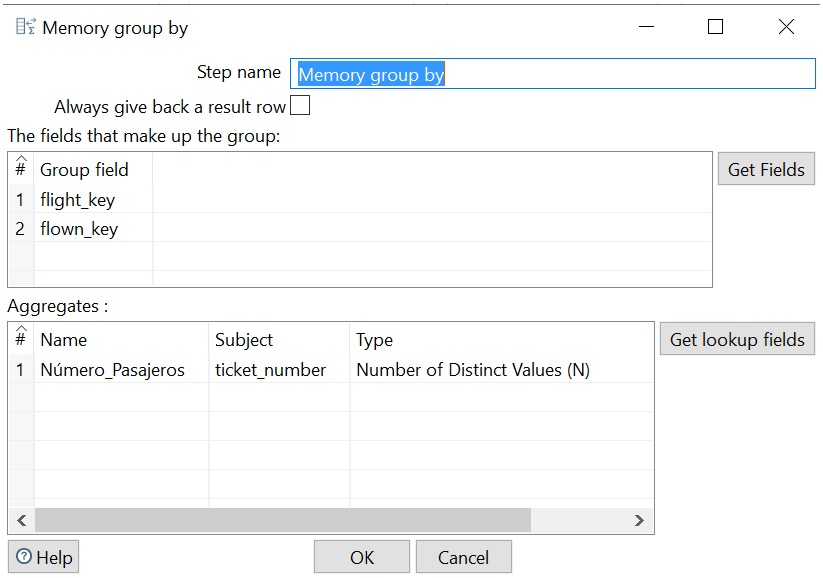


Desde esta etapa surgen dos caminos en paralelos, pero antes de su explicación, cabe indicar que cuando desde un input se ramifica en dos o más ramas diferentes como en este caso, Pentaho te da la opción de distribuir la información entre las ramas o copiar esta información de entrada en cada una de las ramas, en este caso se ha elegido la opción de copiar, para evitar pérdida de datos o posibles complicaciones. Bien, siguiendo con la explicación, como posteriormente se deben unir ambos caminos, se realiza una ordenación ascendente en la etapa llamada “sort rows”, esta ordenación se realiza tanto en la “flight_key” como en la “flown key”:

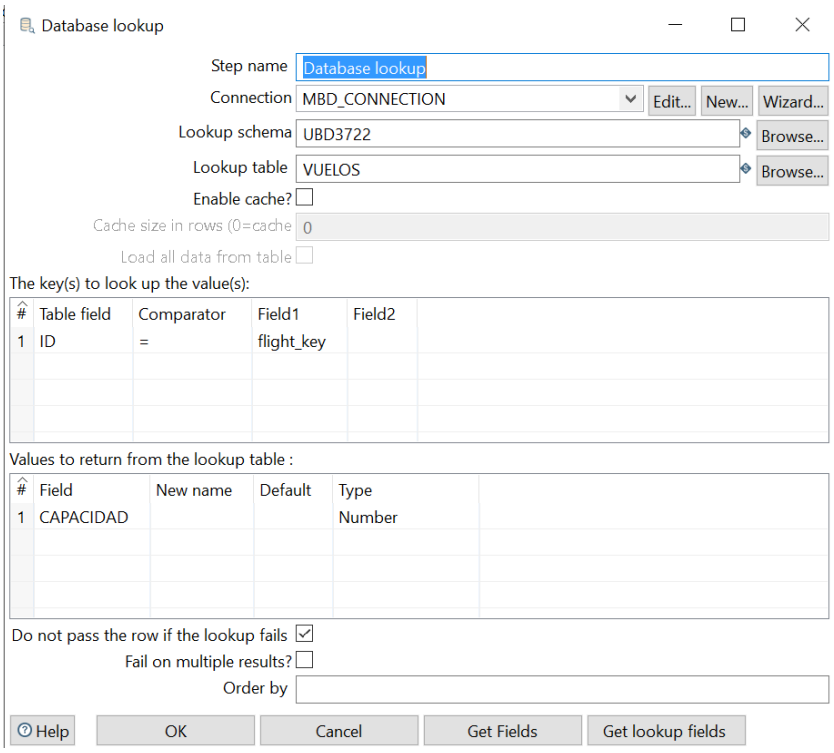


A continuación, se procede a explicar el otro camino, donde la finalidad de este conjunto es crear la última columna que falta para poder insertar todos los datos en la tabla, esta columna es la de “asientos_disponibles”, que se calcularán restando a la capacidad el número de pasajeros que viaja en cada vuelo. Por lo tanto, se debe de crear el campo “número_pasajeros”, lo cual se realiza mediante una etapa “Memory group by”, donde se realiza la agrupación por “flight_key” y “flown_key” de los diferentes números de tickets, obteniendo la cantidad de tickets diferentes que hay en cada vuelo realizado, lo que vendría a ser el número de pasajeros:

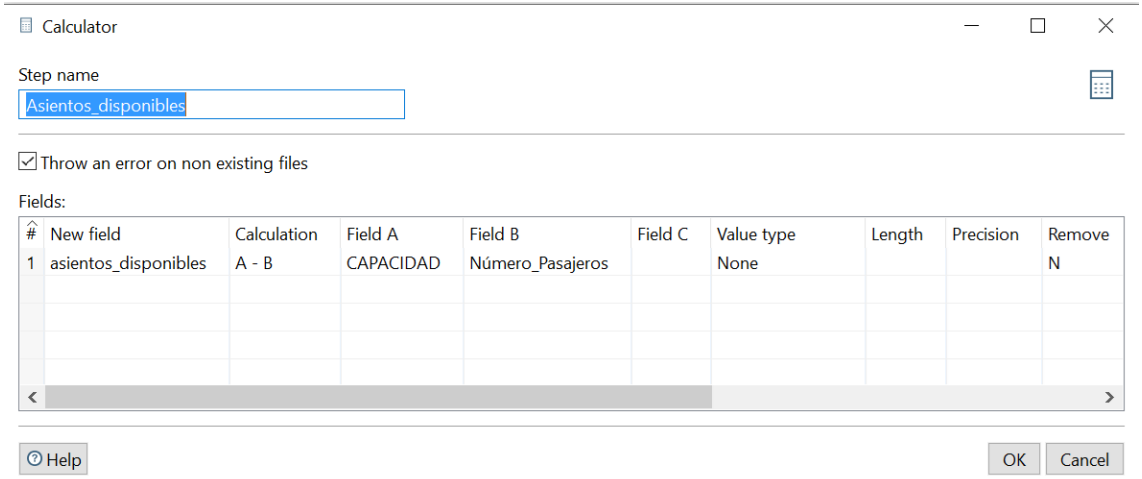
Nota: como en los datasets no vienen datos de fechas, esta agrupación se realiza de la forma anteriormente explicada. Si hubiera información de fechas, se realizaría la agrupación por la “flight_key”, “flown_key”, “fecha” y “hora”, por ese orden.



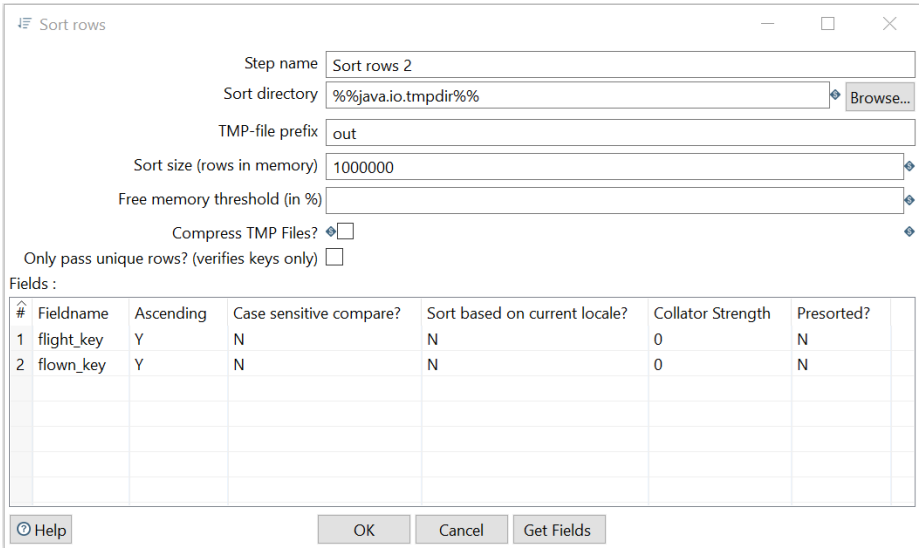
Luego, para calcular el campo de asientos disponibles, falta en los datos de entrada el campo “capacidad”, por lo que se debe de realizar una etapa “Database lookup” a la tabla vuelos para conseguir el campo “capacidad”:



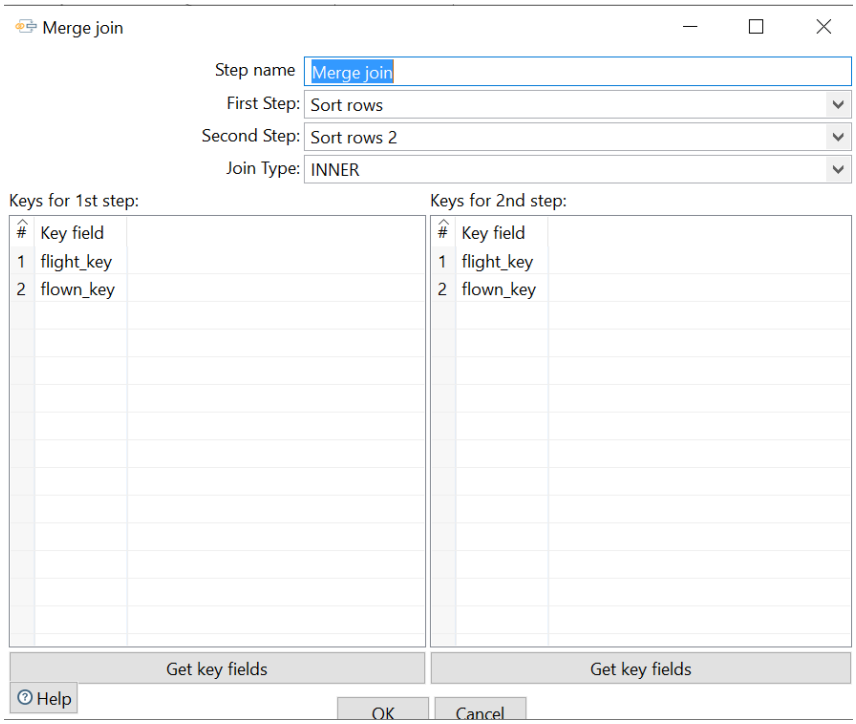
Como ya se ha obtenido la capacidad y el número de pasajeros, se puede proceder a calcular los asientos disponibles con la utilización de la etapa “calculator”:



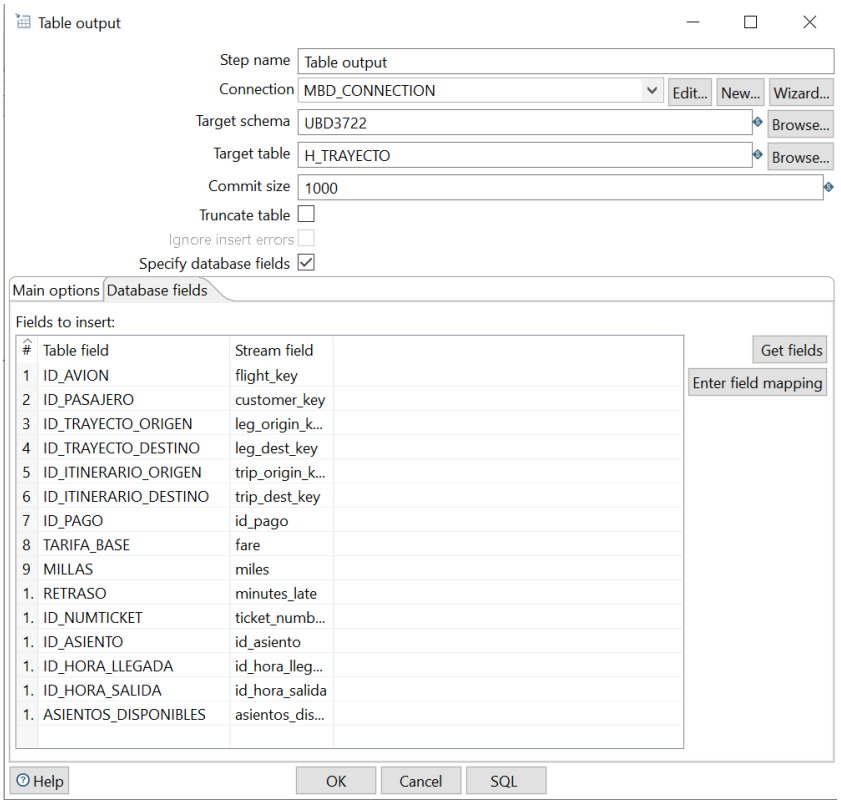
Como se observa, los “asientos_disponibles” se obtienen restando la columna “capacidad” a la columna “numero_pasajeros”. Por último, solamente queda la etapa de ordenación ascendente de las mismas claves, para poder realizar la etapa posterior “Merge join” con garantía de éxitos:



Como penúltimo paso, se tiene la etapa “Merge join” para unificar todos los datos de ambas ramas, esta unión se realizará por los campos “flight_key” y “flown_key”:



Ahora sí, el último paso es la etapa “table output”, gracias a la cual se cargarán los datos a la tabla del hecho (H_TRAYECTO):



Estos son los registros que contienen cada tabla de la base de datos:

	NOMBRE TABLA	Nº REGISTROS
1	AEROPUERTO	20
2	ASIENTO	4
3	HORA	1440
4	H_TRAYECTO	285
5	PAGO	4
6	PASAJERO	100
7	VUELOS	100