# Práctica 2: Procesos ETL en un Data Mart

## Almacenes y Minería de Datos

Pedro Allué Tamargo (758267) Cristina Oriol García (755922)
Alejandro Paricio García (761783)

16 de noviembre de 2020

# Índice

**1. Introducción** **2**

**2. Modificación del esquema en estrella existente** **2**

**3. Proceso ETL** **3**
3.1. Planteamiento y diseño . . . 3
3.1.1. Conjuntos de datos . . . 4
3.1.2. Tratamiento de los vuelos . . . 4
3.1.3. Tratamiento de las ciudades y los aeropuertos . . . 6
3.2. Implementación . . . 8

**4. Conclusiones** **9**
4.1. Control de esfuerzos . . . 9

**5. Anexo 1: Figuras** **10**

**6. Anexo 2: Códigos** **18**

**Referencias** **20**

# 1. Introducción

Tras el desarrollo del *Data Mart* de vuelos comerciales en la práctica anterior se ha procedido al siguiente paso: la población del almacén de datos con datos reales. En esta práctica se va a explorar el concepto de proceso *ETL*. Se va a diseñar e implementar el proceso *ETL* para insertar los datos en el almacén.

# 2. Modificación del esquema en estrella existente

Se ha modificado el esquema en estrella de la práctica anterior para añadir información acerca de las ciudades en las que se encuentra el aeropuerto. Las modificaciones se centran en la dimensión *Aeropuerto*. Se han añadido dos atributos que indican el número de habitantes de la ciudad y la zona horaria de la misma: `ciudadNumHabitantes` y `ciudadTimeZone`, respectivamente. También en esta dimensión se ha eliminado el atributo `codigoAeropuerto` ya que con los conjuntos de datos utilizados no se podía obtener. Al querer incluir la idea de código del aeropuerto que se pretendía obtener en la práctica anterior, se terminó dejando como identificador numérico del aeropuerto el código del mismo.
Durante la labor de búsqueda de los datos se encontraron problemas relacionados con los aviones. En esta dimensión se había decidido considerarlos como los aviones que realizan los vuelos, manteniendo su código de cola y modelo. Con los conjuntos de datos utilizados no había forma de obtener esa información y por lo tanto se ha mantenido en el esquema pero en su posterior traducción a *SQL* se permitirán valores nulos ya que en un futuro es probable que se encuentren estos valores y se puedan asociar a la dimensión avión.
En la tabla de hechos (vuelos) se ha eliminado el hecho `tiempoTotalVuelo` ya que era redundante con el atributo `tiempoRealVuelo`, y no tenía el interés necesario para almacenarlo.
En la dimensión hora se ha eliminado el atributo `timeZone` ya que en esta práctica se pedía que se añadiera esa información a la dimensión de aeropuertos.
En la dimensión fecha se ha realizado una eliminación de atributos que dependían del calendario festivo.
Por lo tanto, el esquema en estrella modificado se puede observar en la Figura 1.

De cara a la implementación *SQL* se ha modificado el código de la práctica anterior para reflejar los cambios comentados anteriormente (Código 1).

# 3. Proceso ETL

## 3.1. Planteamiento y diseño

Se pide la implementación de un proceso ETL que realice una única carga de datos al almacén de datos derivado de las modificaciones realizadas al esquema en estrella de la práctica anterior. Debido al tipo de carga, no va existir una carga incremental, lo que variará el enfoque tomado para desarrollar el proceso ETL, pero se comentará a lo largo del informe alguna opción a considerar en caso de tratarse de esta última. Además, debido al conocimiento previo de uno de los miembros de KNIME, se tomaría como herramienta para el desarrollo, ya que eso facilitaría el lograr un resultado apropiado.

El primer tema tratado es la obtención y extracción de datos de las fuentes de información. Siguiendo las consideraciones dadas por el enunciado, se consultaron las fuentes **GeoNames**[1], una base de datos geográfica, y **RITA**[2] (Research and Innovative Technology Administration), que proporciona información de vuelos comerciales en Estados Unidos. Se observó que la información contenida en ambas era suficiente como para obtener los datos de los vuelos, las aerolíneas, aeropuertos de origen y destino, así como sus ciudades y estados con su población y zona horaria, completándose los requisitos del enunciado de la práctica 2. Una información faltante respecto a la incluida en el esquema en estrella de la práctica anterior eran los modelos de los aviones, tuvo que extraerse del esquema. No suponía un gran problema dado a que no se incluía como uno de los requisitos del almacén, sino que se añadió debido a que podía ser interesante para el análisis. No obstante, de haberse contado con esa información, podría incluirse en el mismo. Decididas las fuentes de información, se obtendrían los datos descargando la información directamente de las bases de datos anteriores en formato csv.

El siguiente paso sería la limpieza y conformado de los datos. Además, se planificaría a grandes rasgos cómo se iban a generar los módulos que obtenían la información en los formatos deseados. También se llevó a cabo la división del trabajo y se decidió quién se encargaría de qué parte, partiendo entre obtención de los datos de aeropuertos y su ciudad y el resto de información de los vuelos, contenida únicamente en la fuente RITA. Más adelante se detectarían casos especiales en los datos, debido principalmente a la no coincidencia de los datos de ambas fuentes en casos específicos, lo que se resolvió durante la implementación. Por ejemplo, aeropuertos con un código de estado distinto a la ciudad en la que se encuentran, lo que se daba, por ejemplo, en algunas localizaciones pertenecientes a Estados Unidos pero fuera del territorio continental.

Finalmente se procedería con la implementación y el poblado del almacén de datos, seguida de la redacción del informe. De la implementación en adelante se detalla en mayor medida, incluidos los problemas encontrados, en los siguientes apartados.

Si se quisiera realizar la carga de los datos de manera incremental se debería automatizar la ejecución del proceso *ETL*. Para ello los ficheros de datos se deberían marcar con un *timestamp* para conocer la fecha de la última ejecución del proceso. El *script* que ejecute la carga incremental cogerá los datos a partir de esa fecha y se los pasará al proceso *ETL*. Esta automatización podría conseguirse utilizando alguna herramienta como *cron* en sistemas *UNIX*.

[1] http://www.geonames.org/,http://download.geonames.org/export/dump/
[2] http://www.transtats.bts.gov

### 3.1.1. Conjuntos de datos

Los conjuntos de datos utilizados se corresponden con los propuestos en el enunciado de la práctica.
Para obtener la información acerca de las ciudades se ha utilizado el conjunto de datos *US* de *GeoNames*[3].
Para obtener la información acerca de los vuelos, se ha obtenido un volcado de la base de datos de *RITA* utilizando el enlace: `https://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=237`. Este enlace ofrece al usuario la posibilidad de elegir qué columnas aparecerán en el fichero de salida.

### 3.1.2. Tratamiento de los vuelos

De la base de datos *RITA* comentada anteriormente se han obtenido los datos que conformaran las tablas *Vuelos, Aeropuertos, Aerolíneas.*
En el fichero de *76480265_T_ONTIME_MARKETING.csv*(vuelos) las columnas son:

- ORIGIN_AIRPORT_ID: id aeropuerto origen.
- DEST_AIRPORT_ID: id aeropuerto destino.
- MKT_UNIQUE_CARRIER: id aerolínea.
- OP_UNIQUE_CARRIER: id operador.
- YEAR: año del vuelo.
- QUARTER: cuatrimestre del año del vuelo.
- MONTH: mes del vuelo.
- DAY_OF_MONTH: día del mes del vuelo.
- DAY_OF_WEEK: día de la semana del vuelo.
- FL_DATE: fecha del vuelo en formato año-mes-día.
- TAIL_NUM: identificador del avión con el que se realiza el vuelo.
- ORIGIN_CITY_NAME: nombre de la ciudad de origen.
- DEST_CITY_NAME: nombre de la ciudad de destino.
- ORIGIN_STATE_NM: nombre del estado de origen.
- DEST_STATE_NM: nombre del estado de destino.
- CRS_DEP_TIME: tiempo planeado de salida.
- CRS_ARR_TIME: tiempo planeado de llegada.
- DEP_TIME: tiempo real de salida.
- ARR_TIME: tiempo real de llegada.
- DEP_DELAY_NEW: diferencia en minutos entre la hora de salida real y la planeada, en caso de adelanto de los vuelos es un número negativo.
- ARR_DELAY_NEW: diferencia en minutos entre la hora de llegada real y la planeada, en caso de adelanto de los vuelos es un número negativo.
- CRS_ELPSED_TIME: tiempo de vuelo en minutos planeado.
- ACTUAL_ELAPSED_TIME: tiempo real de vuelo en minutos.
- AIR_TIME: tiempo en minutos que el avión está volando.

[3] `http://download.geonames.org/export/dump/`

Para la tabla aeropuerto se ha utilizado el campo *OriginAirportID* que contiene el identificador único asignado a cada aeropuerto y que vincula al fichero *L_AIRPORT_ID* que contiene el campo *Description* (nombre de cada aeropuerto). En el fichero de vuelos se encuentran dos referencias a este fichero *ORIGIN_AIRPORT_ID, DEST_AIRPORT_ID* que referencian el aeropuerto de origen y el de destino.
Para las tablas Aerolínea y Operadora se tienen los campos *MKT_UNIQUE_CARRIER, OP_UNIQUE_CARRIER* respectivamente que contienen el identificador único asignado a cada aerolínea u operadora. Estas columnas nos relacionan el fichero de vuelos con el fichero *L_UNIQUE_CARRIERS* que también contiene el campo *Description* en el que se encuentra el nombre de la aerolínea. En el fichero de vuelos se encuentran dos referencias a este fichero *MKT_UNIQUE_CARRIER, OP_UNIQUE_CARRIER* que referencian a la aerolínea y a la operadora del vuelo.

### 3.1.3. Tratamiento de las ciudades y los aeropuertos

Para la obtención relacionada con los aeropuertos y su localización, es decir, su estado, ciudad y la población de la misma entre otros, se han utilizado los siguientes ficheros:

- *L_AIRPORT_ID.csv*: Contiene la información del identificador numérico de cada vuelo, su nombre y su estado, descrito previamente.
- *76480265_T_ONTIME_MARKETING.csv*: Contiene, entre otros atributos descritos anteriormente, el código, nombre de la ciudad y nombre del estado de los aeropuertos de origen y destino de cada vuelo.
- *cities500.txt*: Obtenido de la fuente de datos de *GeoNames* `http://download.geonames.org/export/dump/`, concretamente del fichero cities500.zip. El mismo contiene información acerca de todas las ciudades almacenadas en *GeoNames* con 500 o más habitantes. El conjunto de atributos del mismo se compone de:
  - geonameid: Id entero de la base de datos de *GeoNames*.
  - name: Nombre del punto geográfico (*utf8*) `varchar(200)`.
  - asciiname: Nombre del punto geográfico en caracteres *ascii*, `varchar(200)`.
  - alternatenames: Nombres alternativos separados por coma, obtenidos automáticamente por medio de transliteración [1].
  - latitude: Latitud del punto geográfico en grados decimales.
  - longitud: Longitud del punto geográfico en grados decimales.
  - feature class: Identificador de *GeoNames* de la clase del punto geográfico, ver `http://www.geonames.org/export/codes.html`.
  - feature code: Código de *GeoNames* del del tipo de elemento dentro de una clase del punto geográfico, ver `http://www.geonames.org/export/codes.html`.
  - country code: Código del país, definido por el estándar *ISO-3166* [2] (2 letras).
  - cc2: Códigos del país alternativos, definidos por el estándar anterior.
  - admin1: Para los puntos estadounidenses, se corresponde con el código *FIPS* [3].
  - admin2: Código para la segunda división administrativa, un condado en Estados Unidos.
  - admin3: Código para el tercer nivel de división administrativa.
  - admin4: Código para el cuarto nivel de división administrativa.
  - population: Población de la ciudad, entero de 8 *bytes*.
  - elevation: Elevación en metros del terreno correspondiente a la ciudad.
  - dem: Modelo de elevación digital, *rtm3* o *gtopo30*, elevación media de 3"x3" (ca 90mx90m) or 30"x30" (ca 900mx900m) área en metros, entero.
  - timezone: *iana timezone id*, ver `http://download.geonames.org/export/dump/timeZones.txt`.
  - modification date: Fecha de última modificación en formato yyyy-MM-dd.

**Problemas encontrados**

Partiendo de la descripción previa de las fuentes de datos se observa que deben integrarse datos de dos fuentes de datos distintas. Durante el proceso han surgido una serie de problemas en la integración de la información de ambas. Al comienzo del proceso se obtuvieron los aeropuertos que eran origen o destino de alguno de los vuelos extraídos, quedando 365 aeropuertos, aquellos relevantes para el análisis del rendimiento y de los vuelos (si no hay vuelos, no sirven para el propósito actual del almacén de datos, el análisis temporal de los vuelos, lo que se suma a las restricciones de espacio a la hora de descartar su inclusión). Ellos, se completan fácilmente con su nombre y su ciudad mediante el *join* por *AIRPORT_ID*, que aparece en los ficheros *L_AIRPORT_ID.csv* y *76480265_T_ONTIME_MARKETING.csv*, que provienen de la misma fuente. A continuación, ha de llevarse a cabo el emparejamiento de las ciudades de los aerpouertos con las ciudades obtenidas por *GeoNames* (hubo que eliminar una línea del fichero debido a dar problemas en la lectura, no obstante, era de una ciudad rumana que no afecta de ninguna forma al resultado), donde surge una variedad de problemas:

1. Puerto Rico y Virgin Islands: Aeropuertos de éstas localizaciones tienen un country code distinto a *US*. Hay que tener en cuenta su existencia para permitir que éstos avancen al proceso de emparejamiento aeropuertos-ciudades pasando a través del filtrado de localizaciones, que incluye localizaciones con country code *US*, *PR* o *VI*. A diferencia del siguiente punto, el country code en éstos casos coincide con el estado dado al aeropuerto.

2. U.S. Pacific Trust Territories and Possessions: Los aeropeurtos en éstas regiones tienen como estado *TT*. En cambio, el country code de *GeoNames* se corresponden con una variedad de códigos, como *AS* para American Samoa, *GU* para GUAM y *MP* para las Northern Mariana Islands. Ha habido que incluir módulos al poceso ETL para que empareje correctamente estos estados con su localización y unifique su estado a *TT*. Aeropuertos afectados por este suceso eran Pago Pago International, Guam International y Francisco C. Ada Saipan International.

3. Aeropuerto St George Regional, Utah: Existe una diferencia entre St. y Saint en las fuentes de datos, lo que lleva a que no se emparejen. Puesto que ocurre una única vez se ha completado a mano. En caso de ser una situación frecuente podría llevarse a cabo una expansión de nombres en el matching, pero, al haber una única carga, no es imprescindible incorporarlo al proceso. De consistir en una carga incremental debería incluirse la mencionada expansión para asegurar que el match sea automático y no se llevase a cabo el proceso manualmente cada vez.

4. Aeropuerto de Ashland, West Virginia: El caso del aeropuerto de Ashland es una situación muy concreta, ya que se encuentra en el área de los tres estados, siendo la *Tri-State Airport Authority* la propietaria del aeropuerto. En una fuente de datos indica que el aeropuerto está en el estado de West Virgina, mientras que en la otra no aparece ninguna ciudad de ese nombre en ese estado, sino que aparecen ciudades con ese nombre en otros estados. Es posible que se deba a que Ashland, West Virginia es una unincorporated area [4].

5. Ciudades que no aparecen en *cities500.txt*: Christiansted (Virgin Islands), Deadhorse (Alaska), Hoolehua (Hawaii), Kapalua (Hawaii), Adak Island (Alaska). No aparecen en la fuente de datos GeoNames, algunos por tener menos de 500 habitantes y otros simplemente no se contabilizan en el conjunto, por ejemplo, Adak Island no es una ciudad propiamente dicha y no aparece. La información de estas 5 localizaciones se ha completado manualmente. Este tipo de aeropuertos está en localizaciones despobladas, siendo de las partes de Estados Unidos más alejadas del núcleo del país.

## 3.2. Implementación

Para la implementación del proceso *ETL* se ha utilizado la herramienta *KNIME*[4]. *KNIME* es una herramienta gratuita de código abierto para el análisis de datos, creación de informes. Esta herramienta cuenta con componentes de *Machine Learning* y minería de datos. Cuenta con una interfaz gráfica que permite arrastrar unos componentes (nodos) desde una sección al flujo de trabajo (*workflow*). Estos componentes se pueden conectar entre ellos como tuberías para conseguir funcionalidades complejas [5].

Utilizando los conjuntos de datos descritos anteriormente se han generado dos *workflows* distintos. El primero (Figura 2) se utiliza para transformar el fichero *76480265_T_ONTIME_MARKETING.csv* en los datos de las tablas de dimensiones. Este *workflow* separa las columnas necesarias para cada dimensión en este fichero y realiza las transformaciones necesarias utilizando nodos de *KNIME* y *snippets* de *Java* con el nodo *Java Snippet*.
Tras la transformación de todas las tablas se realizan los *joins* con la tabla de hechos para sustituir las columnas textuales por los identificadores de las tablas de dimensión. Ese proceso se realiza en la Figura 3. Se puede observar que se realizan *joins* con la tabla de hecho, se filtran las columnas necesarias y se renombran para que obtengan el formato que se especificó en el esquema en estrella de la Figura 1.

Para el procesamiento de los ficheros *L_AIRPORT_ID.csv* y *cities500.csv* se ha utilizado el *workflow* de la Figura 4. Este *workflow* se divide en 4 partes.
La primera (Figura 5) se corresponde con la lectura de los ficheros y su limpieza inicial de cara a la segunda etapa.
La segunda etapa (Figura 6) se corresponde con la relación mediante un *join* de los aeropuertos con sus nombres.
La tercera etapa (Figura 7) se corresponde con la relación de los aeropuertos con las ciudades para obtener el *timeZone*, y su número de habitantes. Como se ha comentado anteriormente, existen ciudades con problemas para integrarse con los aeropuertos y por eso se pueden observar dos caminos, el de arriba se corresponden con las ciudades con las que no ha habido ningún problema en el *matching*. El camino de abajo se corresponde con las ciudades especiales (cuyo estado es $TT$).
La cuarta etapa (Figura 8) se corresponde con la concatenación de los dos caminos de la etapa anterior y su escritura en un fichero *CSV*.
Como se ha comentado anteriormente, tras la generación de este *CSV* se han tenido que añadir de forma manual 5 aeropuertos que no constaban en la fuente de datos de *Geonames*.

Tras la obtención de los ficheros *CSV* se ha procedido a la ǵeneración de los ficheros *SQL* asociados. Estos ficheros serán un conjunto de sentencias *INSERT* de tuplas en la Base de Datos *Oracle* situada en *Danae04*. Para la generación de los ficheros *SQL* se ha utilizado una herramienta web[5] que permite convertir los ficheros *CSV* a *SQL*. Debido al tamaño del fichero *vuelos.csv* para la generación del *SQL* se ha utilizado la herramienta *DBeaver*. Con esta herramienta se han cargado los datos directamente del *CSV* a la Base de Datos y luego se han exportado los datos de la tabla generada a un fichero de inserciones *SQL*. No obstante debido a las restricciones de cuota de disco de la Base de Datos no se han podido introducir todas las tuplas correspondientes a esa tabla, y por lo tanto se han introducido 373288 tuplas.
No se ha utilizado la propia conexión a la base de datos de *KNIME* ya que se encontraron problemas con la generación del *driver* asociado a la misma. Los problemas encontrados eran debido a que no se consiguió configurar correctamente las librerías necesarias dentro del entorno gráfico.

[4] `https://www.knime.com/`
[5] `http://www.convertcsv.com/csv-to-sql.htm`

# 4. Conclusiones

Durante el proceso de creación del ETL se ha pasado por una serie de fases en las que se pueden destacar algunos elementos. El primero de los pasos fue el estudio de las fuentes de información, para saber que información se podía obtener para poblar el almacén de datos creado en la parte anterior. Con la información localizada se extrajo de las mismas en formato csv y comenzó el diseño del proceso ETL. Aparece entonces uno de los problemas del proyecto, el cómo dividir el trabajo para poder trabajar al mismo tiempo cada miembro del equipo sin entorpecer a los demás. Finalmente se decidió dividir el trabajo en dos partes: obtener la información de los aeropuertos, sus ciudades y la población de la mismas y el resto de la información relativa a los vuelos, con sus aerolíneas, operadoras, tiempo de retardo... Una vez completado este paso se procedió a implementar el proceso ETL en *KNIME* para obtener los ficheros csv con los que se poblaría el almacén de datos.

Comenzada la implementación del ETL surgieron algunos problemas nuevos que no se habían tenido en cuenta, como los nombrados en el apartado 3.1.3. Gracias a ello se recalca la importancia y necesidad de llevar un estudio exhaustivo de las fuentes de información antes de comenzar con la implementación. La solución para los anteriores no fue excesivamente costosa, ya que no era un problema que afectara a una gran mayoría de aeropuertos, pero si que implicó un retraso que quizá podría haberse eliminado de haber estudiado más la base de datos, pese a que ya se dedicó una cantidad de tiempo relevante para el total de la práctica.

Finalmente, con los *CSV* obtenidos por medio del proceso ETL, se pobló el almacén de datos, surgiendo problemas de limitaciones de espacio, que obligaron a eliminar una parte de los vuelos. Con ello se dejaba el almacén preparado para posteriores prácticas. El equipo considera que se ha afrontado la creación de un proceso ETL con éxito, aprendiendo a utilizar nuevas herramientas y resolviendo los problemas encontrados durante el desarrollo, especialmente aquellos relacionados con la integración de múltiples fuentes de datos.

## 4.1. Control de esfuerzos

- Pedro Allué Tamargo: Modificación del esquema en estrella para admitir los nuevos datos de ciudad, aprendizaje de *KNIME*, limpiar datos de *ONTIME MARKETING*, modificar esquema *SQL*, poblar el almacén de datos, memoria.
  - Horas invertidas: 15 horas
- Cristina Oriol García: Modificación del esquema en estrella para admitir los nuevos datos de ciudad, aprendizaje de *KNIME*, limpiar datos de *ONTIME MARKETING*, obtención de datos de vuelos, memoria.
  - Horas invertidas: 13 horas
- Alejandro Paricio García: Modificación del esquema en estrella para admitir los nuevos datos de ciudad, aprendizaje de *KNIME*, limpiar datos de *Aeropuertos*, obtención de datos de ciudades, memoria.
  - Horas invertidas: 13 horas

# 5. Anexo 1: Figuras

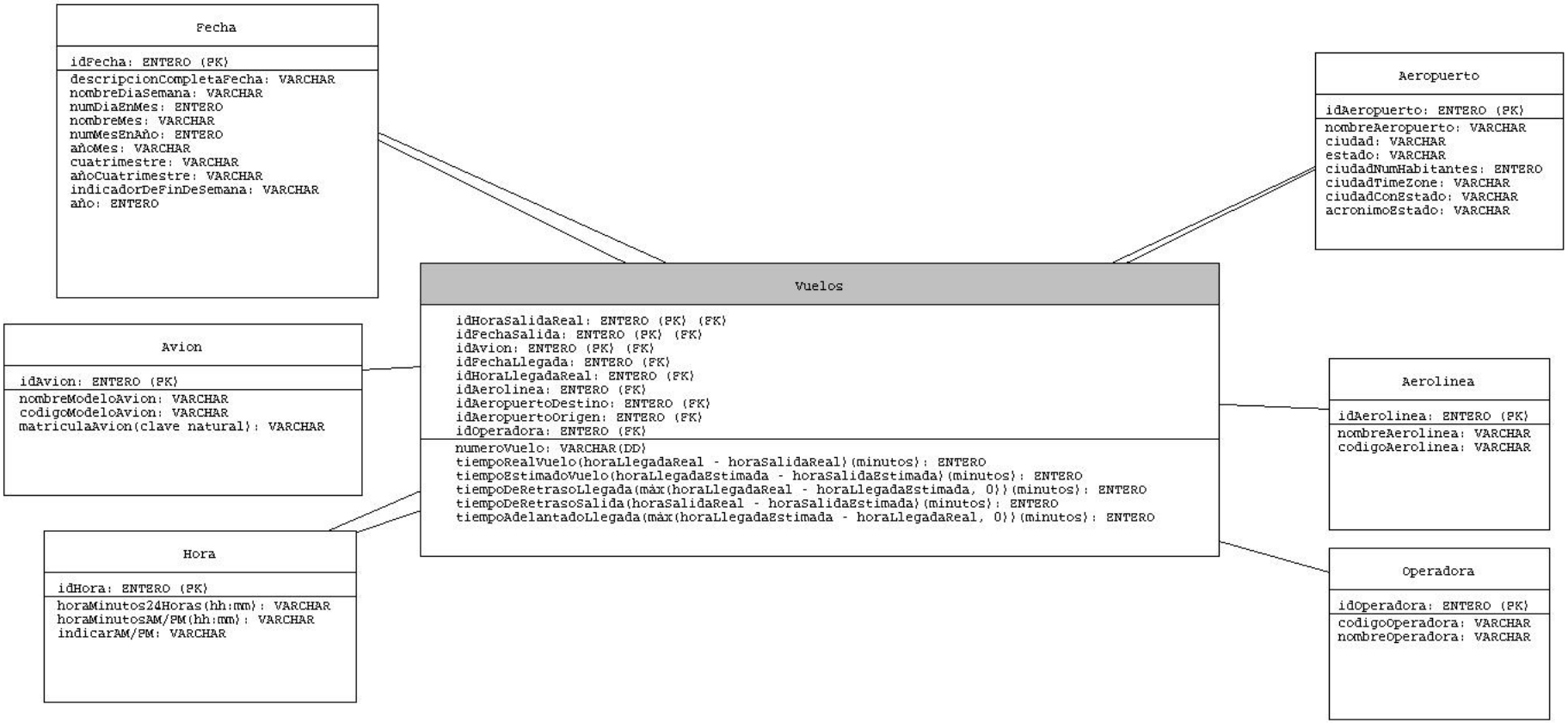


Figura 1: Esquema en estrella modificado para albergar cambios de los aeropuertos

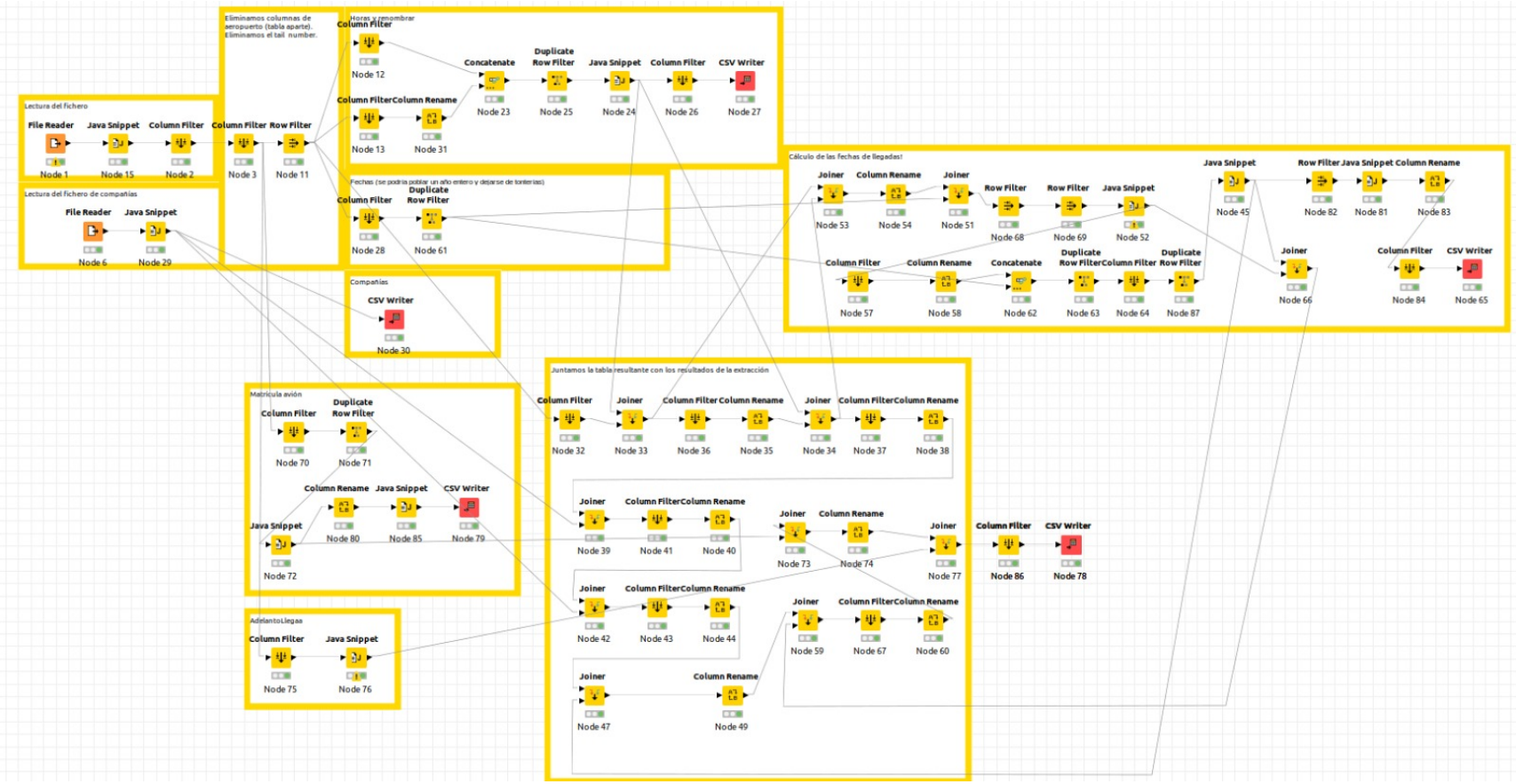


Figura 2: *Workflow* del fichero de vuelos

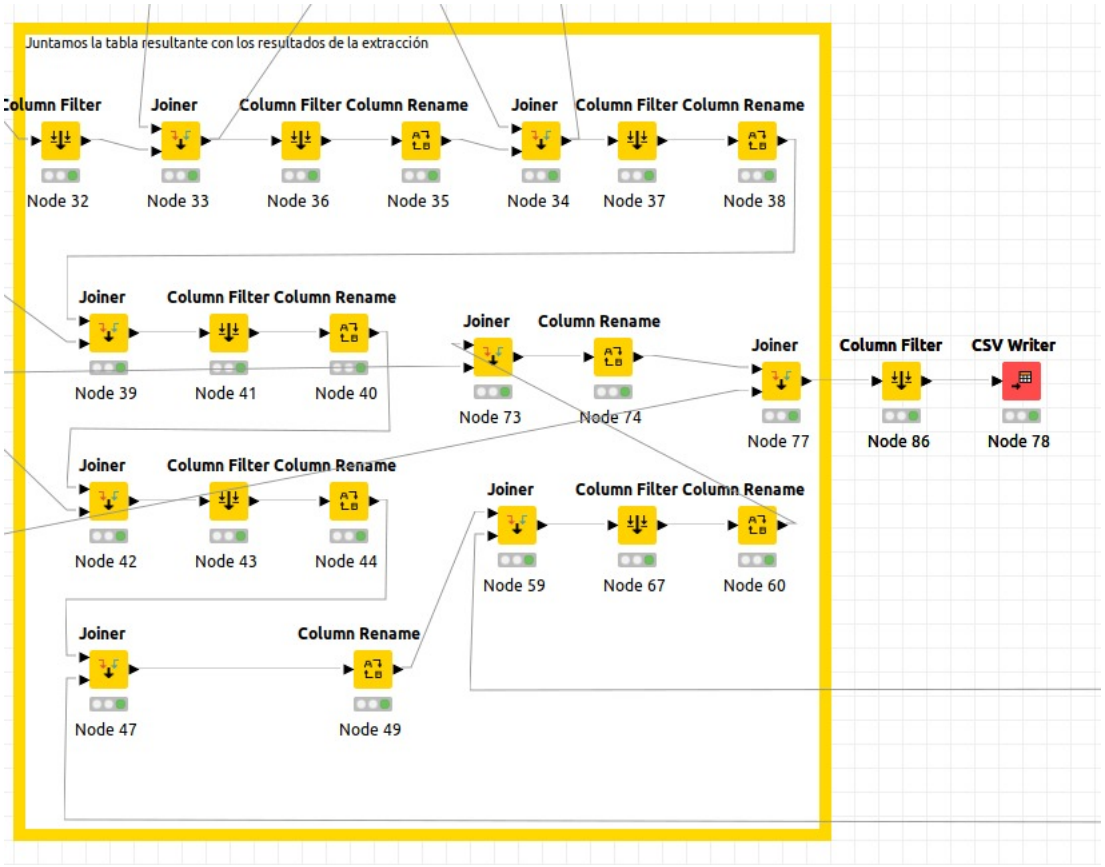


Figura 3: Parte del *Workflow* de procesamiento del fichero de vuelos que se encarga de juntar los identificadores de las tablas de dimensión

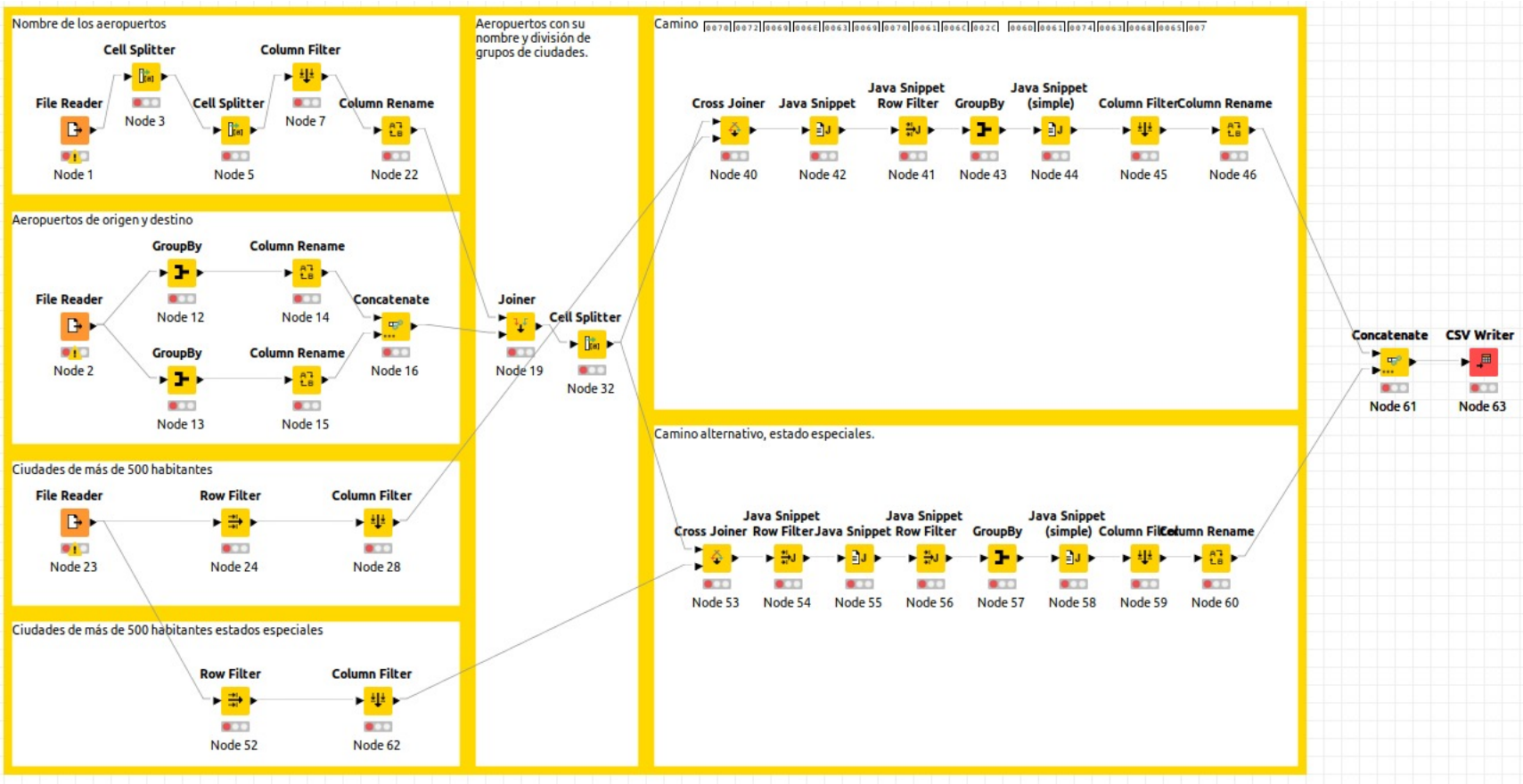


Figura 4: *Workflow* de procesamiento del fichero de aeropuertos y ciudades

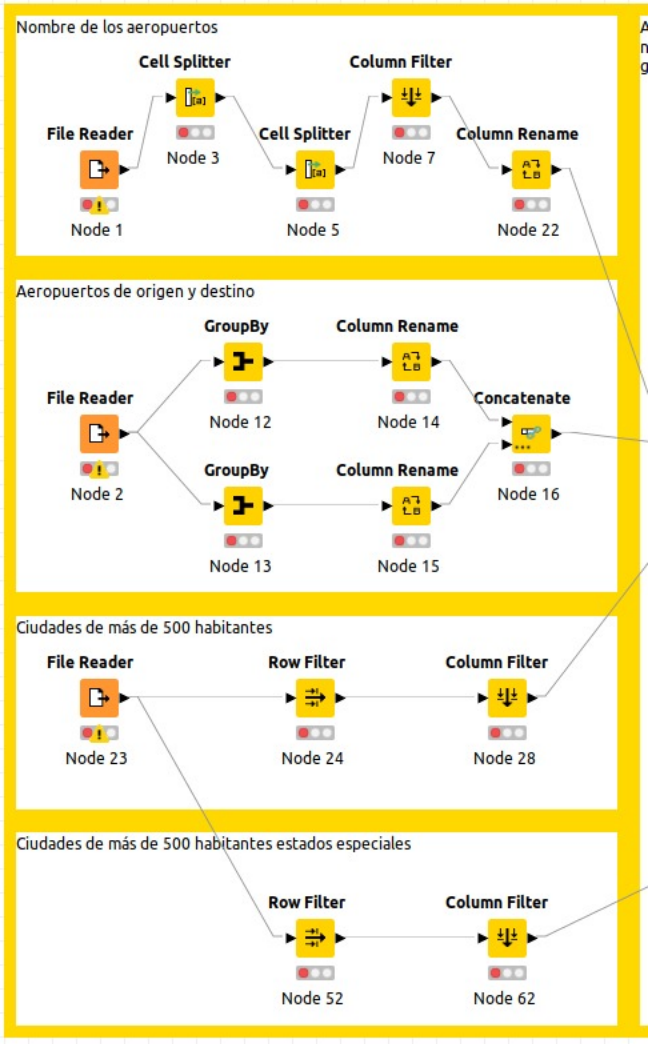


Figura 5: Primera etapa del *Workflow* de procesamiento de aeropuertos y ciudades

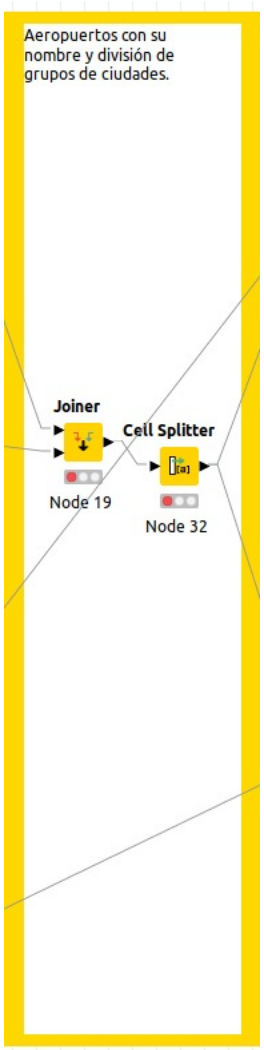


Figura 6: Segunda etapa del *Workflow* de procesamiento de aeropuertos y ciudades

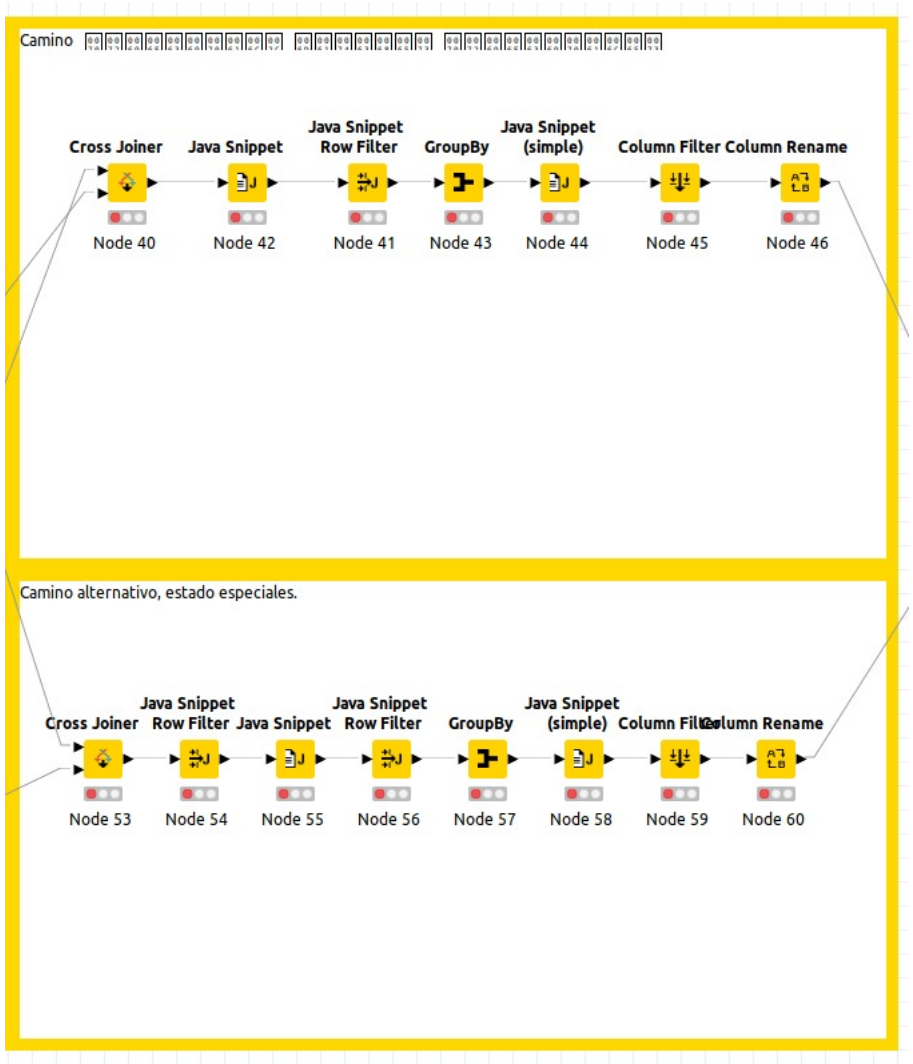


Figura 7: Tercera etapa del *Workflow* de procesamiento de aeropuertos y ciudades

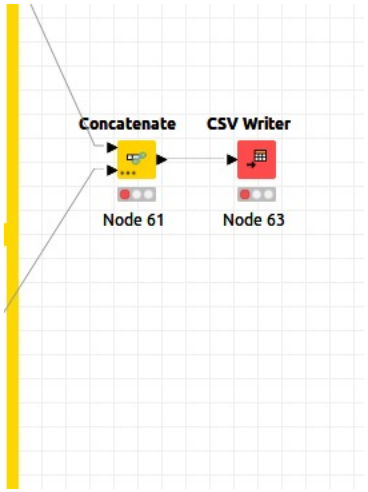


Figura 8: Cuarta etapa del *Workflow* de procesamiento de aeropuertos y ciudades

# 6. Anexo 2: Códigos

Listing 1: Código *SQL* de creación de tablas

```
CREATE TABLE Fecha
(
        idFecha                                 INT,
        descripcionCompletaFecha                VARCHAR(100) NOT NULL,
        nombreDiaSemana                 VARCHAR(100) NOT NULL,
        numDiaEnMes                             INT NOT NULL,
        nombreMes                               VARCHAR(100) NOT NULL,
        numMesEnAo                              INT NOT NULL,
        aoMes                                   VARCHAR(100) NOT NULL,
        cuatrimestre                            VARCHAR(100) NOT NULL,
        aoCuatrimestre                          VARCHAR(100) NOT NULL,
        indicadorDeFinDeSemana                  VARCHAR(100) NOT NULL,
        ao                                      INT NOT NULL,
        PRIMARY KEY(idFecha));


CREATE TABLE Avion
(
        idAvion                                 INT,
        nombreModeloAvion                       VARCHAR(100),
        codigoModeloAvion                       VARCHAR(100),
        matriculaAvion                          VARCHAR(100) NOT NULL,
        PRIMARY KEY(idAvion));


CREATE TABLE Aeropuerto
(
        idAeropuerto                            INT,
        nombreAeropuerto                        VARCHAR(100) NOT NULL,
        ciudad                                  VARCHAR(100) NOT NULL,
        ciudadesConEstado                       VARCHAR(100) NOT NULL,
        estado                                  VARCHAR(100) NOT NULL,
        acronimoEstado                          VARCHAR(100) NOT NULL,
        ciudadNumHabitantes                     INT NOT NULL,
        ciudadTimeZone                          VARCHAR(100) NOT NULL,
        PRIMARY KEY(idAeropuerto));


CREATE TABLE Aerolinea
(
        idAerolinea                             INT,
        nombreAerolinea                         VARCHAR(100) NOT NULL,
        codigoAerolinea                         VARCHAR(100) NOT NULL,
        PRIMARY KEY(idAerolinea));


CREATE TABLE Operadora
(
        idOperadora                             INT,
        codigoOperadora                         VARCHAR(100) NOT NULL,
        nombreOperadora                         VARCHAR(100) NOT NULL,
        PRIMARY KEY(idOperadora));
```

```
CREATE TABLE Hora
(
        idHora                                  INT,
        horaMinutos24Horas                      VARCHAR(100) NOT NULL,
        horaMinutosAMPM                         VARCHAR(100) NOT NULL,
        indicarAMPM                             VARCHAR(100) NOT NULL,
        PRIMARY KEY(idHora));


CREATE TABLE Vuelos
(
        idHoraSalidaReal                        INT,
        idFechaSalida                           INT,
        idAvion                                 INT,
        idFechaLlegada                          INT NOT NULL,
        idHoraLlegadaReal                       INT NOT NULL,
        idAerolinea                             INT NOT NULL,
        idAeropuertoDestino                     INT NOT NULL,
        idAeropuertoOrigen                      INT NOT NULL,
        idOperadora                             INT NOT NULL,
        numeroVuelo                             VARCHAR(100) NOT NULL,
        tiempoRealVuelo                         INT NOT NULL,
        tiempoEstimadoVuelo                     INT NOT NULL,
        tiempoDeRetrasoLlegada                  INT NOT NULL,
        tiempoDeRetrasoSalida                   INT NOT NULL,
        tiempoAdelantadoLlegada                 INT NOT NULL,
        PRIMARY KEY(idHoraSalidaReal, idFechaSalida, idAvion),
        FOREIGN KEY(idHoraSalidaReal) REFERENCES Hora (idHora),
        FOREIGN KEY(idFechaSalida) REFERENCES Fecha (idFecha),
        FOREIGN KEY(idAvion) REFERENCES Avion (idAvion),
        FOREIGN KEY(idFechaLlegada) REFERENCES Fecha (idFecha),
        FOREIGN KEY(idHoraLlegadaReal) REFERENCES Hora (idHora),
        FOREIGN KEY(idAerolinea) REFERENCES Aerolinea (idAerolinea),
        FOREIGN KEY(idAeropuertoDestino) REFERENCES Aeropuerto (idAeropuerto),
        FOREIGN KEY(idAeropuertoOrigen) REFERENCES Aeropuerto (idAeropuerto),
        FOREIGN KEY(idOperadora) REFERENCES Operadora (idOperadora));
```

# Referencias


[1] Wikipedia. Transliteración. https://en.wikipedia.org/wiki/Transliteration. Consultado el 14/11/2020.

[2] ISO-3166. Código países. https://www.iso.org/obp/ui/#search. Consultado el 14/11/2020.

[3] Wikipedia. Código condados fips. https://en.wikipedia.org/wiki/FIPS_county_code. Consultado el 14/11/2020.

[4] Wikipedia. Unincorporated area. https://en.wikipedia.org/wiki/Unincorporated_area. Consultado el 14/11/2020.

[5] Wikipedia. Knime. https://en.wikipedia.org/wiki/KNIME. Consultado el 15/11/2020.