Almacéns e Minaría de Datos

Práctica 3: Apache Hop

Hugo Gómez Sabucedo
hugo.gomez.sabucedo@rai.usc.es

Índice

1-Importar datos dende Talend

En primeiro lugar, para importar os datos a PostgreSQL empregando Talend, temos que facer uns pequenos cambios no arquivo csv. O primeiro que fixemos foi substituír as ‘,’ que separaban os campos por ‘;’. Isto fixémolo porque hai un país (*Bonaire, Saint Eustatius And Saba*) que contiña unha coma no nome, e isto daba erros á hora de importar os datos. En relación con isto, o nome de dito país viña no CSV entre “”. Por tanto, foi preciso eliminalas. Da mesma forma, o código de país para Namibia (NAM) tamén viña entre “”, polo que foi preciso substituílo para eliminar as comiñas.

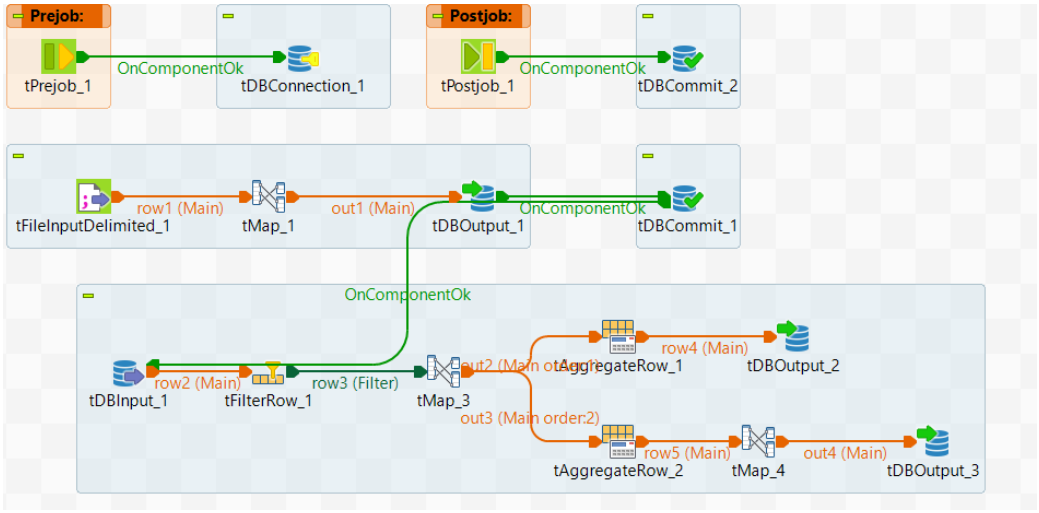
1.1. Creación da táboa para Postgres

Unha vez que temos os datos con formato axeitado, crearemos a táboa de Postgres onde os almacenaremos. Isto facémolo dende fóra do programa Talend, e antes de executalo, posto que, se o facemos dende o propio Talend, dá erros. A táboa que creamos será a que se ve na imaxe. Os tipos de datos elixidos son bastante intuitivos en tódolos casos, pero cabe destacar o dalgúns atributos en específico.

```
CREATE TABLE datoscovid(country varchar(35),
                        code varchar(3),
                        continent varchar(9),
                        population bigint,
                        indicator varchar(1),
                        weeklycount int,
                        year smallint,
                        week smallint,
                        rate14 float,
                        cumulative int,
                        source varchar(55),
                        note varchar(30) )
```

- O atributo *population* tivo que ser definido como un tipo *bigint* xa que, de empregarmos un *int*, este non é suficiente. Isto débese a que, no arquivo, tamén veñen os datos almacenados por continente, e no caso de Asia, a súa poboación (4.498.460.442) é superior ó valor máximo permitido polo tipo de datos int (2.147.483.647).
- *Indicator*, posto que é un dato que só tomará os valores *cases* ou *deaths*, gardarémolo como un varchar de lonxitude 1, co obxectivo de minimizar o espazo usado poola nosa base de datos, de forma que unha *c* se corresponderá cos casos, e unha *d*, coas mortes.
- No que respecta ó atributo *year_week* do arquivo csv, este é un dato co formato AAAA-SS, onde AAAA é o ano e SS a semana do ano. Para que sexa máis sinxelo manexar os datos, separarémolo en dous atributos distintos (*year* e *week*), ambos de tipo smallint, posto que o máximo valor que nos permite almacenar un smallint é 32.767 (e os nosos datos, en todo caso, tomarán como máximo o valor 2022).
- Por último, cabe destacar o campo *Note*. Se observamos o arquivo CSV, vemos que o seu valor é, en tódolos casos, “NA” (é dicir, que non hai ningunha anotación), polo que poderíamos decidir prescindir deste campo na base de datos e aforrar ese espazo.

1.2. Importación de datos a Postgres



Coa táboa creada, pasamos ó *job* de Talend, o cal definimos como se pode ver na imaxe superior. Adxunto a este documento hai un arquivo zip (`covidTalendProject.zip`), que contén o executable obtido dende Talend deste *job*. O primeiro que podemos ver é o *prejob* e o *postjob*. No *prejob*, que será o primeiro en executarse, establecemos a conexión á base de datos. Para isto, precisamos definir os seguintes parámetros: host="localhost", port="5432", database="covid", username="postgres" e o contrasinal que teñamos para o usuario *postgres*. No *postjob*, faremos un *commit* á base de datos e pecharemos a conexión. O seguinte que vemos no *job* son dous *subjobs* distintos: un no cal metemos os datos á táboa de postgres previamente creada, e outro no cal facemos as dúas consultas e as gardamos en cadansúa táboa. Analizaremos estes apartados por separado.

No primeiro *subjob*, temos un *tFileInputDelimited*, co cal importamos os datos dende o arquivo CSV. Para isto, definimos previamente no *metadata* un esquema coa estrutura de dito arquivo. Os datos que obteñamos do CSV pasarémolo a un *tMap*, para crear unha táboa que mapee os atributos que temos no CSV cos da base de datos. Neste tMap, temos que destacar tres cousas principais:

1. En primeiro lugar, é aquí onde converteremos o atributo *indicator*, para que consista dun só carácter. Para isto, simplemente temos que poñer na entrada da columna *indicator* a expresión `row1.indicator.equals("cases")?"c":"d"`. Desta forma, se nesa fila temos "cases", poremos unha "c", e en calquera outro caso, unha "d". Isto facémolo así tendo comprobado previamente que non hai ningún outro atributo nesta columna, o que podería crear inconsistencias na base de datos.
2. En segundo lugar, tal e como explicamos anteriormente, temos que separar o atributo *year_week* do arquivo CSV en dous atributos: *year* e *week*. Como os datos se tratan coma Strings, podemos empregar funcións da categoría StringHandling, que nos permite traballar con Strings. Para obter o ano, na entrada da columna *year* poremos `Integer.parseInt(StringHandling.LEFT(row1.year_week,4))`. De forma análoga, para obter a semana, á entrada da columna *week* poremos `Integer.parseInt(StringHandling.RIGHT(row1.year_week,2))`. No primeiro caso, quedámonos cos 4 primeiros caracteres do campo *year_week* (os correspondentes ó ano) e, no segundo caso, cos 2 últimos

caracteres (correspondentes ó número de semana). É importante realizar o *parseInt*, xa que os datos de entrada que temos son de tipo String, pero queremos obter datos numéricos; se non fixésemos isto, teríamos un erro.

3. Por último, é preciso corrixir un erro que nos atopamos nos campos "weekly_count", "rate_14_day" e "cumulative_count". Para algúns rexistros, estes campos aparecen como *NA*, o cal nos provocaría erros ó intentar inserilos na base de datos. Por iso, é preciso eliminar estes valores. Para isto, á entrada de cada unha das columnas, poremos a expresión: `row1.weekly_count.equals("NA")?Integer.parseInt("0"): Integer.parseInt(row1.weekly_count)`. Neste caso, empregamos como exemplo o campo "weekly_count", e o que facemos é ver se algún deses rexistros é "NA"; se o é, poñemos un 0, se non, o valor que houbera nese campo. É importante facer o *parseInt*, para que o tipo de datos concorde co que hai na base de datos. No caso de "rate_14_day", poríamos o valor 0.0, e faríamos un *parseFloat*.

Unha vez feito o *tMap*, enviaremos eses datos a un *tDBOutput*. Definimos no metadata un esquema que reflicta a estrutura da base de datos para aplicalo no *tDBOutput*. A acción que realizará sobre os datos será un *insert*, mentres que sobre a táboa poremos que faga un *clear table*, para que elimine os datos que poida haber na táboa. A continuación disto, faremos un *tDBCommit*, para gardar os datos xa na base de datos, pero sen pechar a conexión. Isto poderiámolo facer directamente no *postjob* pero, facéndoo así, podemos evitar que o *job* se siga executando no caso de que houbera algún erro ó inserir os datos na táboa.

1.3. Consultas sobre os datos e creación de novas táboas

O segundo *subjob* é onde realizaremos as consultas sobre os datos e os meteremos nas táboas de Postgres. Para isto, porén, é preciso ter creadas as táboas de antemán. Isto facémolo cos comandos que se ven na imaxe inferior. No que respecta á primeira consulta, pídesenos agrupar os casos por país e ano. Para isto, creamos a táboa `casospaisano`, con tres columnas que manteñen o mesmo tipo de datos que a táboa orixinal: *country*, onde almacenaremos o nome do país; *year*, onde gardaremos o ano; e *totalcases*, onde teremos o total de casos. Para a segunda consulta, pídesenos agrupar os casos por continente (de forma que teñamos o número de casos por continente para cada semana) e, ademais, engadir á saída un campo de incidencia semanal por millón de habitantes. Neste caso, a táboa (`casoscontinente`) terá cinco columnas: *continent*, para gardar o nome do continente; *year*, para almacenar o ano; *week*, onde gardaremos o número da semana; *totalcases*, onde teremos o total de casos nesa semana; e *incidence*, onde almacenaremos a incidencia semanal para cada continente, que deberá ser de tipo float.

```
CREATE TABLE casospaisano(country varchar(35),
                          year smallint,
                          totalcases int)
```

```
CREATE TABLE casoscontinente(continent varchar(9),
                             year smallint,
                             week smallint,
                             totalcases int,
                             incidence float)
```

O *subjob* para estas dúas consultas está dividido nunha parte común a ambas, e nunha parte na cal facemos cada unha das consultas. En primeiro lugar, temos un *tDBInput*, co cal recuperamos tódolos datos que temos na táboa á cal volcamos o contido do CSV. Este

campo actívase se o *tDBCommit* que fixemos anteriormente se executou correctamente; desta forma, se houbera algún erro, xa non executaríamos as subconsultas. A continuación, temos un *tFilterRow*, no cal aplicamos dúas condicións. A primeira delas é que o campo *indicator* sexa "c", para coller só as filas que teñan datos dos casos. A segunda condición é que o campo *code* sexa distinto de "NA". Se nos fixamos, o CSV, ademais dos datos por países, tamén contén datos por continentes, en cuxo caso no nome do país temos o nome do continente e, no código, NA. Desta forma, eliminamos todos estes rexistros, xa que non nos interesan. Coa saída filtrada, temos un *tMap*, no cal separamos, por unha parte, as columnas necesarias para a primeira consulta (*country*, *year* e *weeklyCount*) e para a segunda consulta (*country*, *continent*, *population*, *year*, *week* e *weeklyCount*). A partir de aquí é onde faremos as dúas consultas.

1.3.1. Casos por país e ano

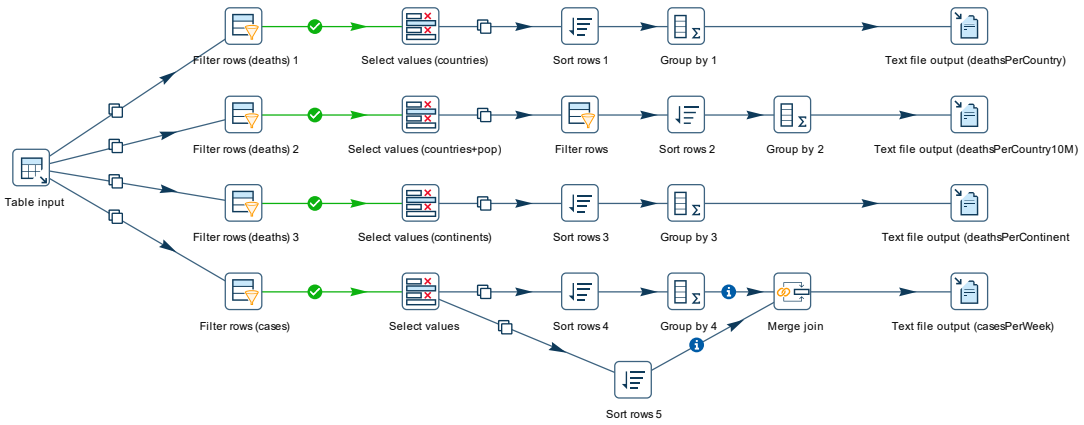
Para obter os casos por país e ano, empregaremos un *tAggregateRow*, no cal agrupamos os datos, cun *group by*, por país (*country*) e ano (*year*), e facemos unha suma dos casos totais (*totalCases*). Isto xerará unha táboa cunha fila por país e ano, e o total de casos nese ano. Así, podemos dirixir a saída deste módulo ó *tDBOutput*, para meter os datos na táboa `casospaisano`. No *tDBOutput*, definiremos a acción sobre os datos como un *insert*, e a acción sobre a táboa como un *clear*, para que elimine posibles datos que puidera haber na táboa. Isto dirixímolo directamente a Postgres, pero na entrega adxúntase un arquivo CSV (`casesCountryYear.csv`) que contén os datos volcados dende a táboa, para a súa comprobación.

1.3.2. Casos por continente e incidencia

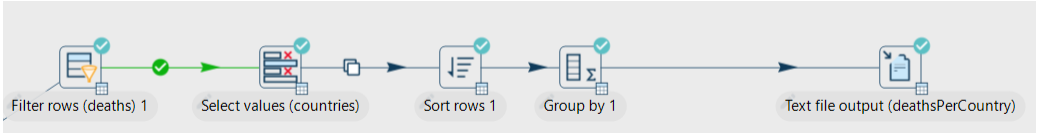
Por último, para os casos por continente, así como a incidencia, faremos tamén un *tAggregateRow*. Neste caso, agruparemos os datos por continente (*continent*), ano (*year*) e semana (*week*), e faremos unha suma dos casos por semana (*weeklyCount*) e da poboación do continente (*population*). A continuación, temos un *tMap*, o cal empregamos para engadir á táboa cos datos a columna da incidencia, e para eliminar as columnas que non queremos na base de datos. Desta forma, á saída do *tMap* teremos as columnas *continent*, *year*, *week*, *weeklyCount* e *incidence*. Para calcular a incidencia, simplemente temos que poñer, á entrada da columna *incidence*, a expresión `((double)row5.weeklyCount*(double)1000000)/(double)row5.population`. A fórmula para calcular a incidencia é (casos*1.000.000)/poboación. Todos estes datos podemos obtelos dende a táboa que temos como entrada. É importante poñer o `(double)` para forzar a que os datos sexan recoñecidos como un número decimal, e o resultado da operación sexa un número decimal tamén. Agora xa só nos quedaría dirixir a saída do *tMap* a un *tDBOutput* onde, igual que para a anterior consulta, facemos un *insert* sobre os datos e un *clear* sobre a táboa. Tamén, igual que no anterior dato, introducimos directamente os datos en Postgres, pero na entrega hai un arquivo adxunto (`casesContinent.csv`) cos datos volcados directamente dende a táboa, para a súa comprobación.

2-Importar datos dende Apache Hop

Para traballar co Apache Hop, creamos no entorno do proxecto unha *pipeline*. A vantaxe de usar unha *pipeline* é que realiza operacións en paralelo, o que nos permitirá resolver antes as consultas e ter un tempo de execución máis rápido. No que respecta ós datos, obterémolos directamente dende Postgres, da táboa na cal volcamos o arquivo CSV. Desta forma, o esquema de Apache quedaría como se ve na imaxe inferior. Adxunto á práctica hai un arquivo zip (`covidApacheProject.zip`) co proxecto exportado dende Talend. Como se pode ver, collemos tódolos datos dende Postgres cun *Table input*, dende o cal sacamos 4 fluxos (*hop*), un para cada unha das consultas que temos que realizar. Vamos analizalos un por un.

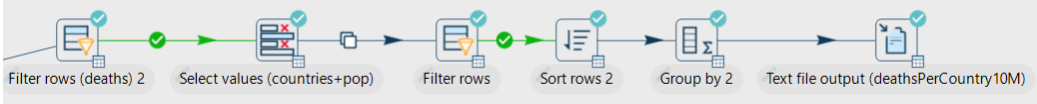


2.1. Número de mortes por país



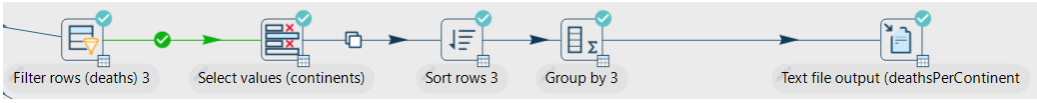
Cos datos de entrada de Postgres, o primeiro que temos que facer é un filtrado das filas para, de forma similar a como ocorría no apartado anterior, eliminar as filas cuxo código de país é "NA". Ademais, estableceremos unha condición adicional, para quedarnos cos datos nos cales o indicador é "d" (mortes). A continuación, temos un *transform* do tipo "Select values", que nos permitirá elixir as filas coas que nos queremos quedar. Neste caso, elixiremos só *country* (país) e *weeklyCount* (casos semanais). A continuación, teríamos que facer un *group by*, para agrupar e sumar as mortes por país; porén para isto, é un requisito que teñamos os datos ordenados. É por iso polo que introducimos un módulo "Sort rows", para ordenar os datos por país, de forma ascendente. Unha vez feito isto, podemos agrupar os datos por país, e realizar un *sum* sobre a columna *weeklyCount*, o cal denominaremos *totalDeaths*. Por último, almacenaremos a saída nun arquivo csv, denominado `deathsPerCountry.csv`, o cal está adxunto a este documento.

2.2. Mortes anuais por países con máis de 10 millóns de habitantes



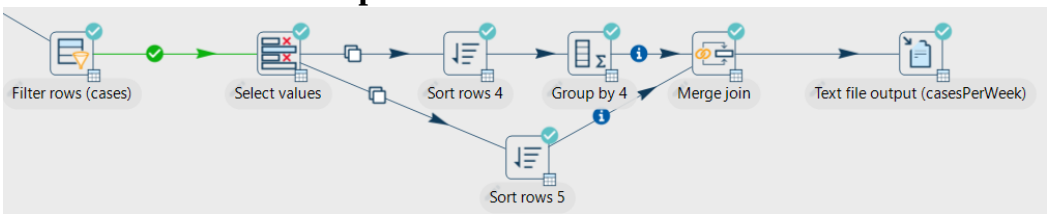
Para calcular as mortes anuais por países con máis de 10 millóns de habitantes, facemos un proceso similar ó anterior. En primeiro lugar, filtramos as filas, para eliminar aquelas cuxo código de país é "NA" e quedarnos coas que teñen como indicador "d". Non é posible empregar a saída do "Filter rows" dúas veces, posto que esta só nos deixa elixir entre os datos que cumpren os requisitos dos filtros e os que non os cumpren. A continuación, seleccionamos da táboa as columnas *country*, *population*, *year* e *weeklyCount*, que son as que empregaremos na consulta. Volvemos aplicar outro filtro ás filas, neste caso para quedarnos só cos datos dos países cunha poboación superior ós 10.000.000 habitantes, como se nos pide no enunciado. Cun "Sort rows" ordenamos os datos resultantes por país e ano, de forma ascendente, e co "Group by" agrupámolos por país e ano, e facemos un sum sobre *weeklyCount*, para obter as mortes totais, que denominaremos *totalDeaths*. Ó non especificar no *group by* o campo *population*, este non se gardará xa na táboa resultante. Só nos queda almacenar os resultados da consulta nun arquivo CSV, que denominaremos `deathsPerCountry10M.csv`, e que está adxunto a este informe.

2.3. Número de mortes por continente



Neste apartado, realizamos tamén un filtrado de filas igual que no anterior, para eliminar aquelas cuxo código de país sexa "NA" e, ademais, quedarnos só coas que teñen como indicador "d". No "Select values", quedámonos só cos datos de continentes e casos por semana (*continents* e *weeklyCount*). Ordenamos os datos por continente, e aplicamos un *group by* para agrupalos por continente e facer un sum sobre os casos semanais, o cal denominaremos *totalDeaths*. O último paso é gardar os resultados no arquivo csv, tamén adxunto a este informe, denominado `deathsPerContinent.csv`.

2.4. País con máis casos por semana



Este último apartado é distinto ó anterior, e tamén máis complexo. O "Filter rows" ten unha estrutura similar ó dos anteriores apartados: eliminamos as filas cuxo código de país é "NA" pero, neste caso, como se nos pide os países con máis casos por semana, temos que elixir aquelas nas que o indicador é "c". No "Select values", escolleremos os campos *country*, *year*, *week* e *weeklyCount*.

A partir de aquí, para realizar o apartado, podemos pensar a consulta como unha subconsulta, na cal seleccionamos, para cada semana, o número máximo de casos e, empregando os resultados da subconsulta, facemos un *join* no ano, semana, e casos, dende

a táboa inicial. Desta forma teríamos, para cada semana, o país con máis casos, e o número de casos dese país. A subconsulta, en SQL, sería da forma en que se ve na imaxe.

```sql
SELECT d1.country, d1.year, d1.week, d1.weeklycount
FROM datoscovid as d1
JOIN
    (SELECT year, week, max(weeklycount) as maxcases
    FROM datoscovid
    WHERE indicator='c' AND code!='NA'
    GROUP BY year, week) as d2
ON d1.year=d2.year
AND d1.week=d2.week
AND d1.weeklycount=d2.maxcases
WHERE indicator='c' AND code!='NA'
ORDER BY year ASC, week ASC
```

Para realizar esta operación en Apache Hop, é evidente que precisaremos un *join*. Por tanto, da saída do "Select values", debemos sacar dous fluxos, que serán idénticos. Un será o que empreguemos para a subconsulta, mentres que o outro serán os "datos da táboa". A un destes fluxos, simplemente aplicaremos un "Sort rows", ordenando por ano, semana e país, xa que o compoñente "Merge join" requírenos que os datos estean ordenados. Por outra parte, o outro fluxo tamén o ordenaremos por ano, semana e país, e á saída aplicarémoslle un *group by*, agrupando por ano e semana e seleccionando o valor máximo do atributo *weeklyCount*. Estes dous fluxos serán os que poremos de entrada no *join*, onde o *first transform* será o do fluxo agrupado, e o *second transform*, o fluxo que só ordenamos. O *join* será de tipo INNER, e as chaves do mesmo serán as mesmas que tiñamos na consulta: ano (*year*), semana (*week*) e casos semanais (*weeklyCount*). Con isto, teríamos xa o país con máis casos cada semana. Porén, ó facer o *join*, este gardará dúas veces os datos do ano, semana e casos. Por tanto, ó gardar os resultados no csv, debemos elixir só os campos *country*, *year*, *week* e *weeklyCount*. Isto almacenarémolo no csv denominado `casesPerWeek.csv`, que tamén está adxunto a este informe.