Almacéns e Minaría de Datos

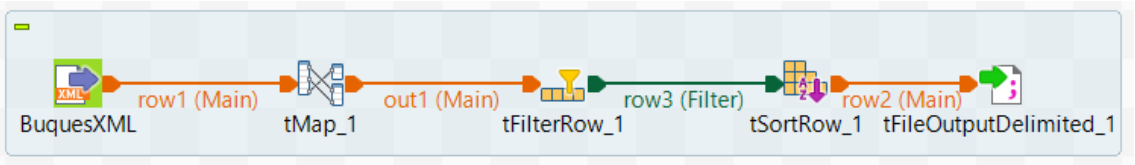
Práctica 2: Talend Open Studio

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

Índice

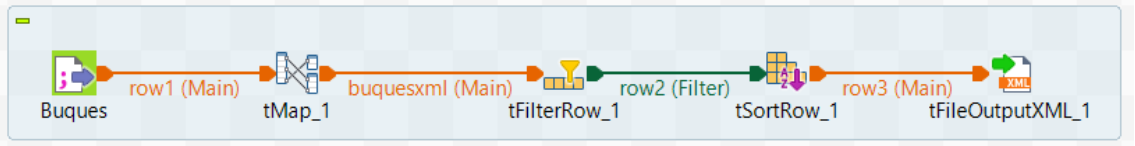
1-Conversión de arquivo XML en CSV e viceversa

En primeiro lugar, converteremos o arquivo XML a un arquivo CSV. Para isto, temos que crear, no apartado de *metadata*, metadatos para o "File XML", de forma que indiquemos o arquivo dende o cal se importarán os datos e a súa estrutura (columnas, nome das mesmas, etc...). En *schema*, teremos que elixir que campos queremos extraer do arquivo XML. Temos que indicar o *Xpath expression* e poñer -1 no *Loop limit*, para que colla tódolos datos correctamente.



A continuación, crearemos un *job* como o que se ve na imaxe. Temos como entrada un arquivo XML, e empregamos un *tMap* para mapear as columnas de entrada do arquivo coas columnas que exportaremos logo ó arquivo CSV. Neste punto, é importante que nos fixemos en posibles erros que teña o arquivo e que os tipos de datos coincidan cos que Talend asume automaticamente. No noso caso, temos que poñer a columna Folio como un tipo *String*, posto que por defecto virá como un tipo *Integer*. A saída deste *tMap* empregámola como entrada nun *tFilterRow*, no cal nos quedaremos só cos datos de 2020. A continuación, empregamos un *tSortRow* para ordenar os datos polo nome do barco, de forma ascendente. Por último, só nos queda conectar a saída do *tSortRow* coa entrada dun *tFileOutputDelimited*, o cal nos exportará os datos ó arquivo CSV que indiquemos.

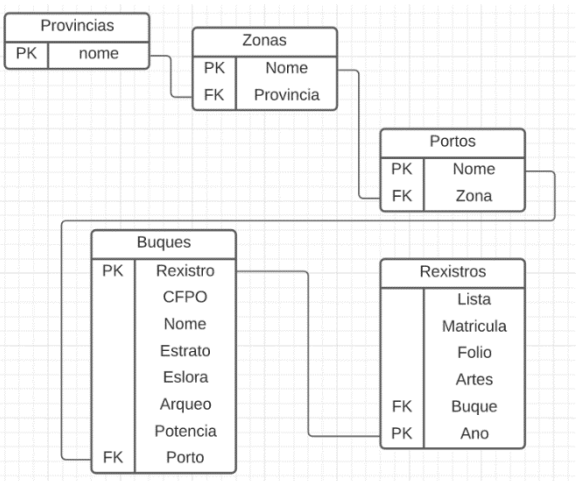
A segunda parte de exercicio consiste en converter o arquivo CSV descargado a un arquivo XML. Igual que antes, temos que crear metadatos para o "File delimited", indicando onde está o arquivo CSV e o tipo de separadores que ten (*field separator*, que no noso caso será un ';' ; e *row separator*, que será o '\n').



O *job* creado neste caso é moi similar ó anterior. Temos como entrada o arquivo CSV que definimos no anterior metadata, e co *tMap*, mapeamos os seus campos cos campos que queremos que teña o arquivo XML. Tamén, neste caso, temos que cambiar o tipo de dato da columna Folio de *Integer* (que vén por defecto) a *String*. A continuación, aplicamos un *tFilterRow* para quedarnos só cos datos de 2020, e un *tSortRow* para ordenar os datos, de forma análoga ó anterior apartado, por nome do buque. Unha vez feito isto, podemos empregar un *tFileOutputXML* para exportar os datos ó arquivo XML.

Neste proceso de converter os arquivos, a diferencia máis salientábel que podemos atopar é a forma na que importamos os arquivos. No caso do arquivo CSV, é moi sinxelo, posto que só temos que indicar os separadores de columnas e filas; porén, no caso do XML, temos que indicar o XPath e o Loop Limit, así como os distintos campos do arquivo. Pola contra, traballar co arquivo XML directamente faise moito máis sinxelo, en tanto que podemos saber máis facilmente que información hai en cada unha das columnas do arquivo, simplemente lendo o arquivo. No caso do CSV, é preciso empregar a documentación dispoñible na web de descarga para isto, ou ben cotexalo co arquivo XML. Polo demais, no que respecta ós resultados, salvo a diferenza na estrutura interna dos arquivos, os datos son exactamente iguais en ambos casos.

2-Importación de datos a PostgreSQL



O primeiro paso que debemos facer para importar os datos a un esquema normalizado é crear dito esquema. Para normalizar, temos que empregar atributos atómicos en tódalas táboas, así como evitar ter datos duplicados ou repetidos. Desta forma, o esquema normalizado constará das 5 táboas que se ven na imaxe.

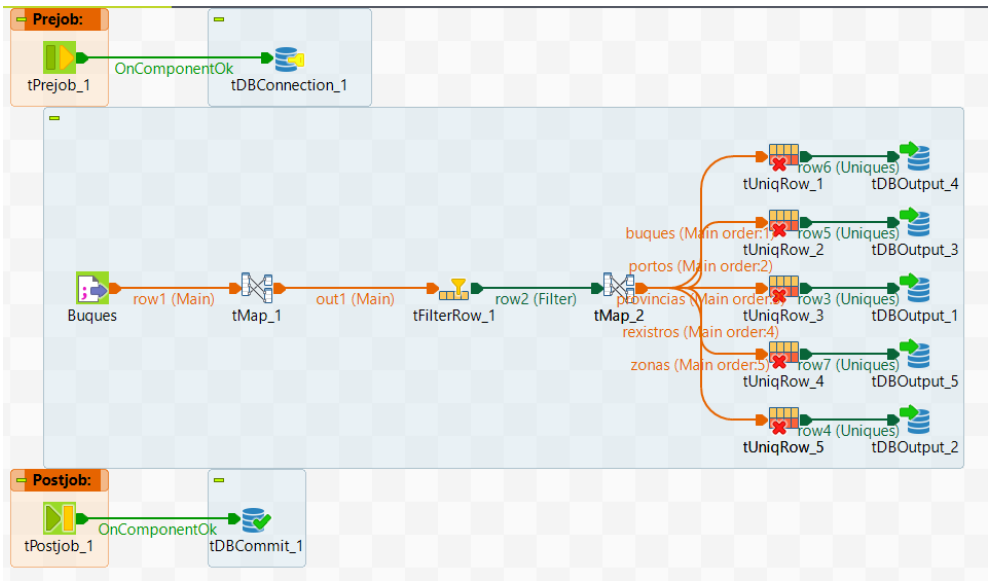
Neste caso, temos unha táboa na cal almacenaremos o nome das provincias. Na segunda táboa, almacenaremos o nome das zonas, e a provincia á cal pertence dita zona, mediante unha referencia á táboa provincias. A terceira táboa conterá información sobre os portos: o seu nome, e a zona na que se atopan (como referencia á táboa zonas). A táboa cuarta contén información sobre os buques; en particular, o seu número de rexistro (que é a chave primaria, posto que é único), o número do Censo da Flota Pesqueira Operativa (abreviado como CFPO), o nome do buque, o seu estrato, a eslora, o arqueo, a potencia, e o porto base do buque (como unha referencia á táboa Portos). Por último, na táboa Rexistros almacenamos información sobre cada rexistro de cada buque (que contén un rexistro por cada buque e ano): temos a lista, a matrícula, o folio, as artes, o ano (que é a chave primaria) e o buque (que é unha referencia á táboa buques).

Con isto, podemos crear unha base de datos en postgres, coas correspondentes táboas. Isto facémolo co código que se amosa nas seguintes imaxes.

```
create table Provincias(
    nomeProvincia varchar(10) NOT NULL,
    PRIMARY KEY (nomeProvincia)
);

create table Zonas(
    nomeZona varchar(30) NOT NULL,
    provincia varchar(10),
    PRIMARY KEY (nomeZona),
    FOREIGN KEY (provincia) REFERENCES Provincias(nomeProvincia)
);

create table Portos(
    nomePorto varchar(25) NOT NULL,
    zona varchar(30),
    PRIMARY KEY (nomePorto),
    FOREIGN KEY (zona) REFERENCES Zonas(nomeZona)
);
```

```sql
create table Buques(
    rexistro varchar(12) NOT NULL,
    cfpo integer,
    nome varchar(30),
    estrato varchar(55),
    eslora float,
    arqueo float,
    potencia float,
    porto varchar(25),
    PRIMARY KEY (rexistro),
    FOREIGN KEY (porto) REFERENCES Portos(nomePorto)
);

create table Rexistros(
    lista integer,
    matricula varchar(7),
    folio varchar(6),
    artes varchar(125),
    buque varchar(12),
    ano integer NOT NULL,
    PRIMARY KEY (ano),
    FOREIGN KEY (buque) REFERENCES Buques(rexistro)
);
```

Unha vez creada a base de datos, previo a importar os datos, temos que facer un cambio no formato dos números decimais do arquivo CSV, posto que estes veñen separados por ‘,’, e para que sexan un formato apto para Postgres, deben de estar separados por un ‘.’. Con isto, podemos comeza o *job* de Talend. En primeiro lugar, temos que definir un *prejob*, que consistirá en establecer a conexión coa base de datos. Para isto, é necesario engadir información sobre “Db connections” no apartado metadata, para poder empregar estes datos para conectarnos á base de datos de Postgres. Temos que definir o host como localhost, o porto como 5432, o nome da database que crearamos (neste caso, buques), o schema, que será public, e o usuario (postgres) e contrasinal). Pola súa parte, o *postjob* é simplemente un commit, para gardar os cambios realizados na base de datos, e o peche da conexión.



No que respecta ó *job* en si, en primeiro lugar, importamos os datos dende o arquivo CSV. É importante, como dixemos anteriormente, que teña cambiado o formato dos números decimais. Como estamos importando os datos directamente dende o CSV, antes de poder filtralos, temos que empregar un *tMap* para separalos nas distintas columnas, de forma análoga a como fixemos no primeiro exercicio. A continuación, si, podemos realizar o filtrado dos datos. Ademais de quedarnos só cos datos do ano 2020, neste caso pedíasenos rexeitar os datos de buques cunha eslora menor a 4 metros, así como escoller que se importen unicamente os datos dun único porto (no noso caso, escollemos Ferrol). Todas estas condicións debemos agrupalas cun condicional AND.

Unha vez que obtivemos e filtramos os datos, empregaremos outro *tMap* para separalos. Neste caso, temos como *input* a saída do *tFilterRow*, e teremos 5 saídas do *tMap*, cada unha delas correspondéndose coas 5 táboas que creamos na base de datos. No proprio *tMap*, podemos crear as distintas saídas, elixindo que datos queremos que se almacenen en cada unha das saídas. Posteriormente, empregaremos un *tUniqRow* antes de volcar os datos á base de datos, para evitar que teñamos datos duplicados. No noso caso, por exemplo, o que queremos evitar é que, se vamos importar 12 rexistros, se importe 12 veces á táboa das provincias o valor "A Coruña".

Antes de realizar o *tDBOutput*, temos que definir 5 "Generic schemas", un para cada unha das táboas. Neles, teremos que indicar o nome das columnas de entrada, o nome de cada unha das columnas tal e como aparece na base de datos, así como se dita columna é unha chave ou non, e se admite valores nulos. Por defecto, poremos que todas admitan valores nulos, agás aquelas que conteñan unha chave primaria. Con isto, temos todo o necesario para executar o *job* e importar os datos a PostgreSQL

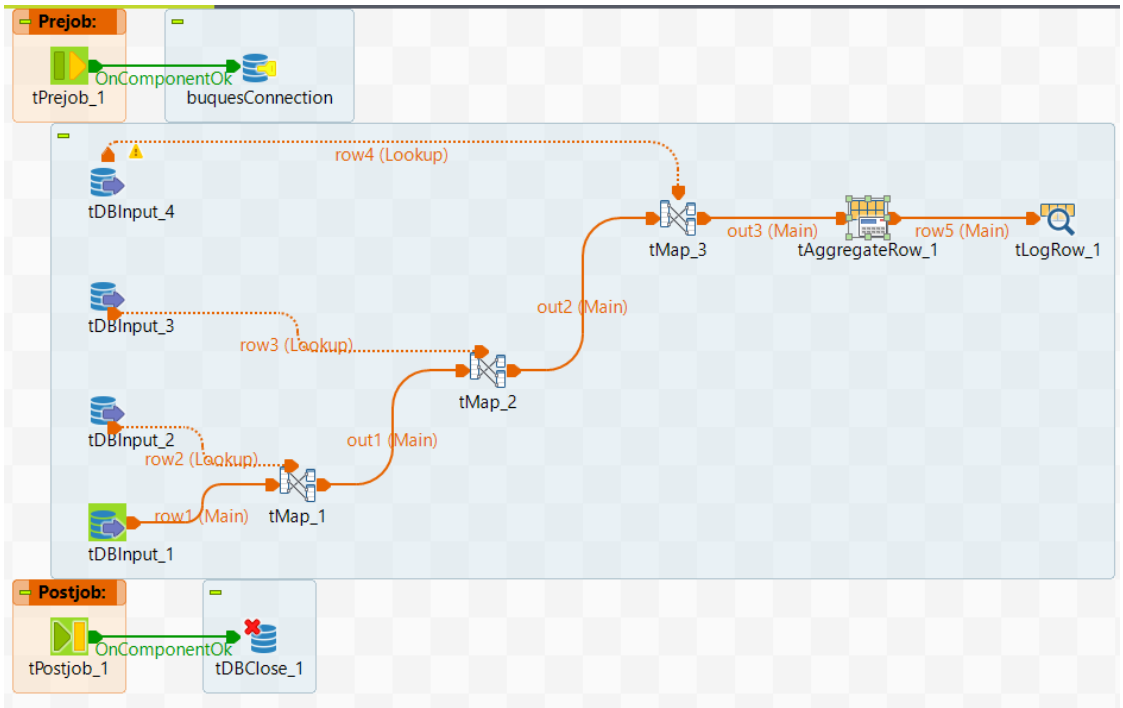
3-Cálculo do arqueo total por provincia

En primeiro lugar, vamos a calcular o arqueo total con Postgres. Para isto, é preciso executar a seguinte consulta SQL. Nela, temos que coller os datos de 4 táboas distintas: o nome das provincias está almacenado na táboa provincias, pero o arqueo na táboa de buques. Como non hai ningunha vinculación directa entre as provincias e os buques, temos que seleccionar o nome da provincia da táboa das provincias, e chegar á táboa de buques obtendo o nome das zonas asociadas a cada provincia, dos portos asociados a cada zona e dos buques asociados a cada porto. Agrupamos por provincias e calculamos a suma do arqueo, obtendo o resultado que se mostra na imaxe.

```
SELECT pr.nomeprovincia, SUM(bu.arqueo) as Arqueo

FROM provincias as pr, zonas as zo, portos as po, buques as bu

WHERE pr.nomeprovincia=zo.provincia

  AND zo.nomezona=po.zona

  AND po.nomeporto=bu.porto

GROUP BY pr.nomeprovincia
```

nomeprovincia [PK] character varying (10)	arqueo double precision
A Coruña	51157.95999999993
Lugo	23848.31
Pontevedra	80551.49000000012

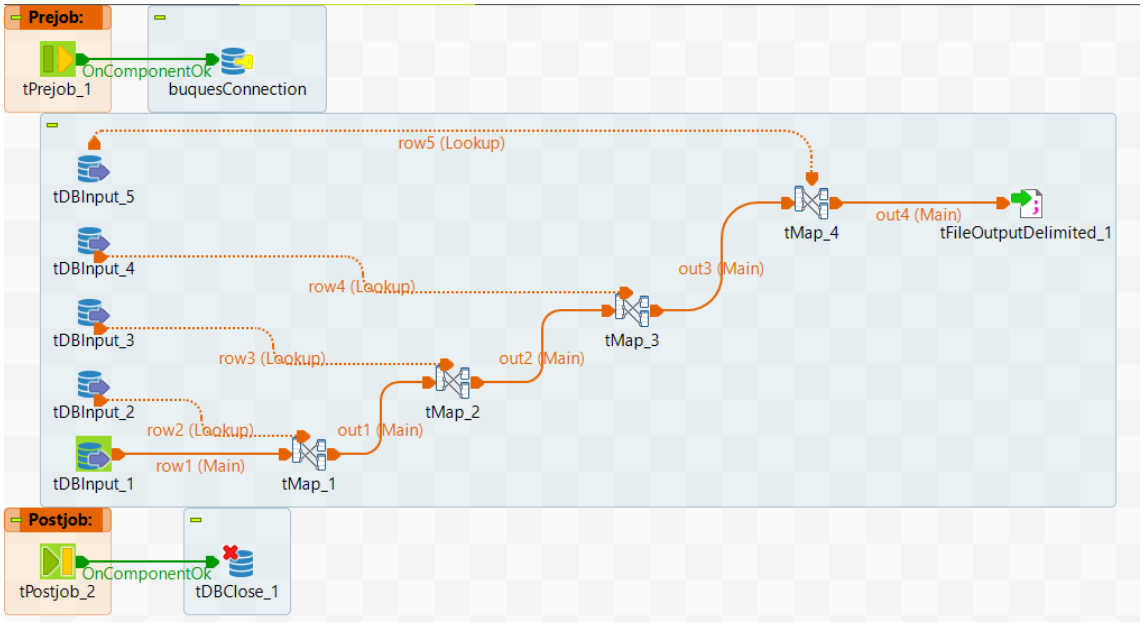
A continuación, vamos calcular o arqueo total por provincia empregando Talend. Como fixemos no apartado anterior, definimos un *prejob*, no cal estableceremos a conexión á base de datos. Para isto, empregaremos a mesma "Db connection" que antes. O *postjob*, neste caso, non será un *commit*, senón simplemente pechar a conexión á base de datos.



No que respecta ó *job* en si, temos 4 inputs, un por cada táboa que temos que importar (igual que explicamos no caso de Postgres). Nos *tMap*, o que facemos é o mesmo que nas cláusulas *where* da consulta: unir tódalas táboas, para obter unha na que teñamos os arqueos e a correspondente provincia. Unha vez que temos tódolos datos nunha táboa, empregamos un *tAggregateRow*, no cal agrupamos os datos por provincia e facemos a suma do arqueo. Por último, co *tLogRow*, mostramos o resultado por pantalla.

4-Esquema desnormalizado simulando o arquivo orixinal

Para crear o esquema desnormalizado, temos que seguir un proceso similar ó anterior. Creamos un *prejob* no que establecemos a conexión á base de datos, e un *postjob* no que pechamos a conexión coa base de datos. No *job*, temos que poñer 5 inputs, cada un deles correspondéndose con cada unha das táboas que ten a nosa base de datos normalizada. Posteriormente, con *tMap*s, podemos ir creando o noso arquivo desnormalizado. Así, no primeiro *tMap*, unimos os nomes das provincias cos nomes das zonas; no seguinte, unimos as zonas cos portos; no terceiro, unimos os portos cos buques; e no último, os buques cos rexistros. Desta forma, podemos conectar o *output* do cuarto *tMap* á entrada dun *tFileOutputDelimited*, e exportar a base de datos a un arquivo desnormalizado.



No que respecta a executar o traballo dende liña de comandos, podemos xerar facilmente un arquivo binario facendo clic dereito no traballo, e seleccionar onde queremos crealo. Talend fará un arquivo .zip, no cal poderemos atopar o traballo para executalo con diferentes extensións (.jar, .bat, .ps1 ou .sh). Só temos que facer dobre clic sobre, por exemplo, o arquivo .bat, e abrirase unha terminal co seguinte *output*, e crearase o arquivo CSV no directorio elixido no *job*.

```
C:\Users\hugo0\Downloads\exportDesnorm_0.1\exportDesnorm>C:

C:\Users\hugo0\Downloads\exportDesnorm_0.1\exportDesnorm>cd C:\Users\hugo0\Downloads\exportDesnorm_0.1\exportDesnorm\

C:\Users\hugo0\Downloads\exportDesnorm_0.1\exportDesnorm>java -Dtalend.component.manager.m2.repository="C:\Users\hugo0\D
ownloads\exportDesnorm_0.1\exportDesnorm/../lib" -Xms256M -Xmx1024M -cp .;../lib/routines.jar;../lib/log4j-slf4j-impl-2.
13.2.jar;../lib/log4j-api-2.13.2.jar;../lib/log4j-core-2.13.2.jar;../lib/log4j-1.2-api-2.13.2.jar;../lib/commons-collect
ions-3.2.2.jar;../lib/jboss-marshalling-river-2.0.12.Final.jar;../lib/jboss-marshalling-2.0.12.Final.jar;../lib/advanced
PersistentLookupLib-1.3.jar;../lib/dom4j-2.1.3.jar;../lib/slf4j-api-1.7.29.jar;../lib/trove.jar;../lib/postgresql-42.2.1
4.jar;../lib/talendcsv-1.0.0.jar;../lib/crypto-utils-0.31.12.jar;exportdesnorm_0_1.jar; amd_p2.exportdesnorm_0_1.exportD
esnorm --context=Default
```