Ejercicio con Pentaho.

Jerónimo Carranza Carranza

Introducción

El ejercicio se ha realizado con la herramienta *Pentaho Data Integration* (*Kettle*) en su versión 7.1, ejecutándose sobre OpenJDK 64-Bit Server VM 1.8.0_131, en una máquina con Linux Fedora 24 y 4 Gb Ram.

El ejercicio consiste en un tarea ETL que realiza la descarga de datos de un servicio web a intervalos regulares y los transforma y almacena en una base de datos.

El servicio web es uno de los ofrecidos por la empresa JCDecaux en su página web para 27 ciudades en las que opera los servicios de bicicletas compartidas. Concretamente se accede al servicio de datos en tiempo real, a través del cual, se puede obtener para una ciudad en concreto los datos de cada una de las estaciones de bicicletas compartidas, proporcionando:

- Datos estáticos:
  - **number**: número de la estación. Único para cada contrato.
  - **contract_name**: nombre del contrato (Seville, en nuestro caso)
  - **name**: nombre de la estación
  - **address**: dirección de la estación
  - **position**: posición de la estación en coordenadas geográficas WGS84
  - **banking**: indica si la estación tiene un terminal de pago
  - **bonus**: indica si se trata de una estación bonificada
- Datos dinámicos:
  - **status**: indica si la estación está CLOSED o OPEN
  - **bike_stands**: número de estacionamientos operativos en la estación
  - **available_bike_stands**: número de estacionamientos disponibles en la estación
  - **available_bikes**: número de bicicletas operativas y disponibles en la estación
  - **last_update**: timestamp que indica última actualización en milisegundos desde Epoch

El servicio de datos en tiempo real requiere registro previo y en las peticiones hay que incluir una APIKey válida. En concreto la petición es de la forma

GET https://api.jcdecaux.com/vls/v1/stations?contract={contract_name}&apiKey={apiKey}

y la respuesta (exitosa) en la forma

[station_json, station_json, ...]

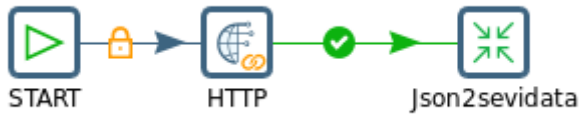
El servicio se actualiza en origen cada segundo. En nuestro caso cursamos peticiones cada minuto y los datos se almacenan en una base de datos sqlite (*stations.sqlite*).

La base de datos de almacenamiento tiene dos tablas; *seviesta* almacena los datos estáticos, que ya están almacenados y no son objetivo de este ejercicio, y *sevidata* que almacena los datos dinámicos a intervalos de 5 minutos (para el ejercicio se ha incrementado la frecuencia a 1 minuto).

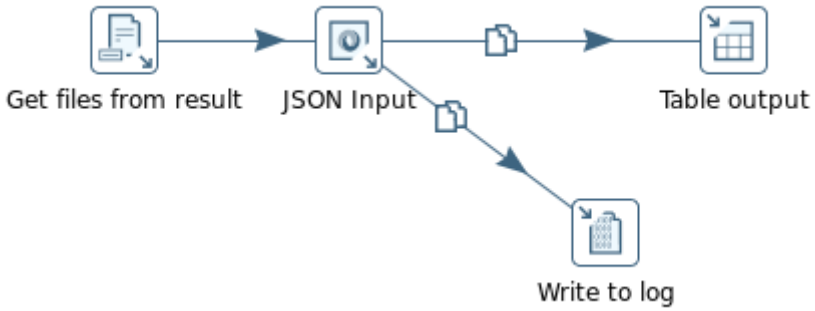
Implementación

Se implementa el ejercicio mediante una Tarea (*stations.kjb*) y una Transformación (*JsonInput_sevidata.ktr*). Los diagramas de los procesos, tarea y transformación, son los siguientes:

stations.kjb



JsonInput_sevidata.ktr



La tarea es una tarea repetitiva que se reinicia cada minuto, e incluye:

- La petición GET HTTP al servicio, cuyo resultado es el almacenamiento del fichero JSON de respuesta en la carpeta ***stations_tmp/*** con un patrón de nombre único que incluye el timestamp de su creación.
- La llamada al proceso de Transformación *Json2sevidata* que se almacena en *JsonInput_sevidata.ktr*.

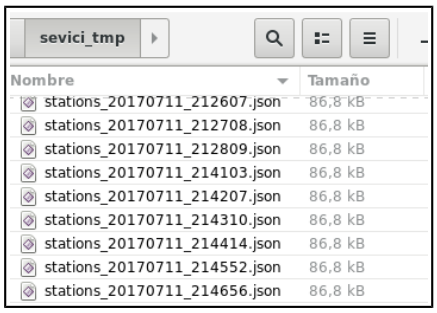
La transformación incluye:

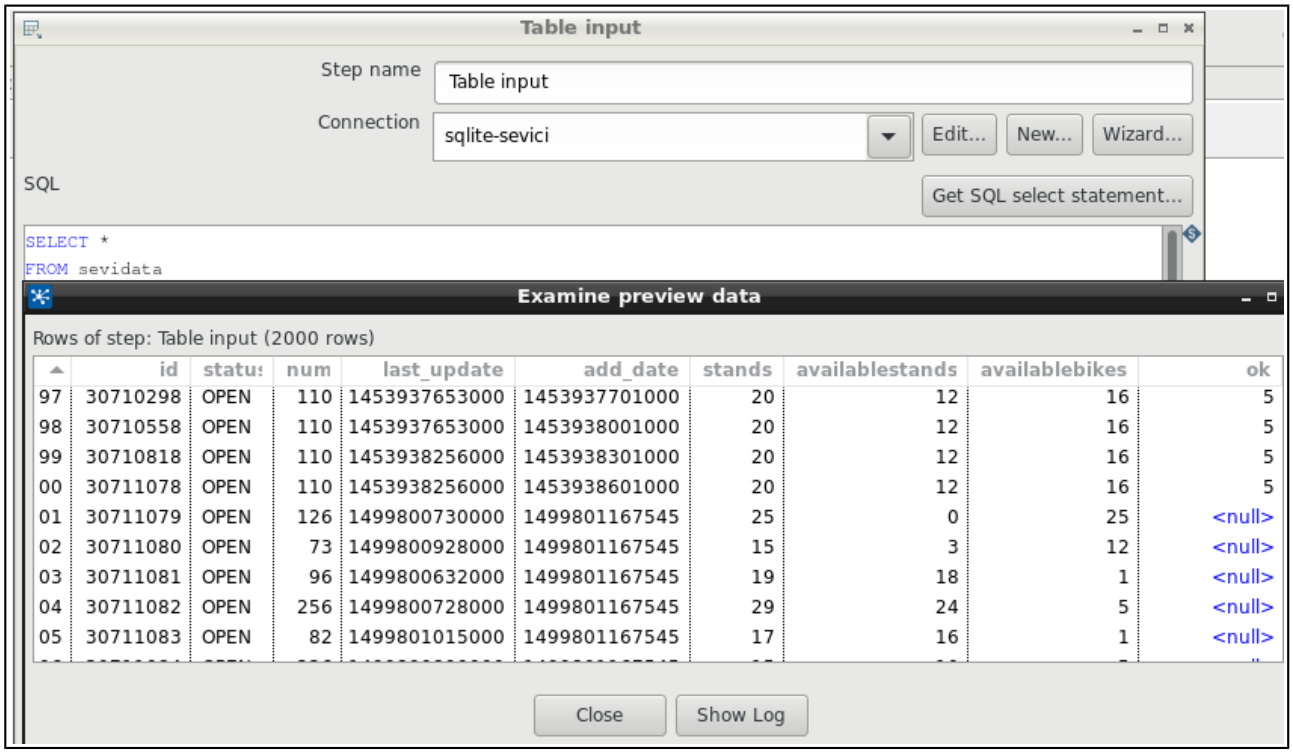
- La recuperación de metadatos del último fichero incorporado a resultados.
- El tratamiento del referido fichero como entrada JSON.
- La salida de los datos de interés a tabla en BD SQLite y registro en el log.

Resultados

Los resultados del proceso son:

- Colección de ficheros json descargados:
- Base de datos *stations.sqlite* con tabla *sevidata* actualizada:





	id	statu:	num	last_update	add_date	stands	availablestands	availablebikes	ok
97	30710298	OPEN	110	1453937653000	1453937701000	20	12	16	5
98	30710558	OPEN	110	1453937653000	1453938001000	20	12	16	5
99	30710818	OPEN	110	1453938256000	1453938301000	20	12	16	5
00	30711078	OPEN	110	1453938256000	1453938601000	20	12	16	5
01	30711079	OPEN	126	1499800730000	1499801167545	25	0	25	<null>
02	30711080	OPEN	73	1499800928000	1499801167545	15	3	12	<null>
03	30711081	OPEN	96	1499800632000	1499801167545	19	18	1	<null>
04	30711082	OPEN	256	1499800728000	1499801167545	29	24	5	<null>
05	30711083	OPEN	82	1499801015000	1499801167545	17	16	1	<null>