ETL and Console of the Virtual Machine
User Manual

January, 2018

Abstract

This manual exposes techniques and guidelines for the implementation of a good data acquisition process to enrich the knowledge base that is the basis of several national (SiiMobility, http://www.sii-mobility.org) and European projects (Resolute, http://www.resolute-eu.org; Replicate, http://www.resolute-eu.org; Select4cities, http://www.select4cities.eu). The tools used are: Penthao Spoon, Karma, Apache Phoenix, MySQL database and Hadoop HBase. The main goal is to present a guided example. The user is invited to follow (where possible) the useful tips provided by tutors. The user is also invited to discuss with them for any doubt or question. The tutors themselves will be in charge of reviewing and validating the processes carried out by the user.

1 Introduction: ETL process (Extract, Transform, Load)

The data acquisition process is composed by several phases. A preliminary phase involves the study of the assigned dataset. It's needfull to understand the data to proceed in the best viable way to the following phases. Generally, two types of datasets are identified:

- ***static***: information about the type of service identified, which presumably does not change over time or that anyway have sporadic variations. For example the geographical location of a building, the identifier, the address, etc.
- ***real-time*** *(or periodic / dynamic)*: information that changes over time (in short intervals) therefore requires a higher acquisition frequency and usually has a static price

Example: Dataset that includes information on the state of traffic flow reported by various sensors located in the city of Florence.

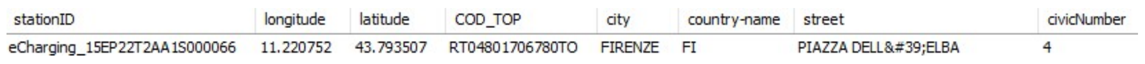
In this case there are both static data and dynamic data:

- ***static***: sensor position (coordinates) and related specifications (id, type of measurements, etc.)
- ***real-time*** *(or periodic / dynamic)*: number of cars / motorcycles / etc. crossing a stretch of road, etc.

Note that in this case the dynamic data is connected to the static one: it is necessary to know which sensor in Florence estimated the passage of N cars in a given time T.

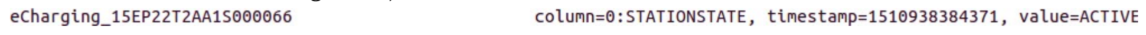
The processing of these two types of datasets requires different techniques. You need to create an ETL for static data and a different ETL to capture and process the dynamic data:

- Each ETL provides for the final upload of data on HBase tables.
- As previously mentioned, it is necessary to maintain the connection between static and related dynamic data (if present). The connection is made via a common column in the HBase tables of the two ETLs (static and dynamic). This column contains the identifier (of a sensor, of a hospital, other). Through this connection it will be possible to connect dynamic data to static information. In Figure 1 it's possible to appreciate an example.



stationID	longitude	latitude	COD_TOP	city	country-name	street	civicNumber
eCharging_15EP22T2AA1S000066	11.220752	43.793507	RT04801706780TO	FIRENZE	FI	PIAZZA DELL&#39;ELBA	4

Figure 1a): row of a HBase table related to static data



```
eCharging_15EP22T2AA1S000066                    column=0:STATIONSTATE, timestamp=1510938384371, value=ACTIVE
```

Figure 1b): Example of HBase tables of a dynamic data, connected to the static data of 1a)

Possibile kinds of datasets:

- static ONLY datasets: points of interest (POI, Point of Interest) of a city (churches, museums, etc.)
- dynamic ONLY dataset: earthquakes reports (they are real time and the epicenter of the earthquake is always different)

1.1 ETL input and output

The ETLs are mechanisms of data processing. They start from the ingestion (Extraction) and then continue with the re-elaboration of the data that includes phases of qualification and aggregation of data (Transforming) and the loading of data on the store (Load).
It has been decided to maintein:

- The final data:
  - in a NoSQL store (HBase)
  - in rdf triples (to aggregate data between themselves and connect them to KM4City multi-ontology)
- the input parameters of the ETLs in a MySQL table. For example: web server address to download the dataset, user license associated to the downloaded dataset, description of the dataset and /or data provider, directory where the rdf triples (file.n3) will be saved, the semantic category of KM4City associated with the data (class), etc. The table is called 'process_manager2', and is the same for all ETLs. The information to be included in this table are explained in detail in paragraph 4.

2 How to use the Virtual Machine and ETL tools

<u>Virtual machine (VM) access:</u>

The Virtual Machin can be executed with VMware player or workstation, with the following credentials:

- User: ubuntu
- Password: ubuntu

<u>Tools to use and related commands:</u>
The following tools are needed to manage the ETL in the VM.

1. Connect to HBase:
   - *Links:*
     - http://hbase.apache.org/
   - *Version*: Apache HBase ver. 1.2.5 (Database NoSQL), in uso come stand alone
   - *Commands:*
     - To run HBase (from any directory): "start-hbase.sh"
     - To stop HBase (from any directory): "stop-hbase.sh"
     - To check the execution: "jps"
     - To check the execution from web interface, visit the web page : http://localhost:16010/master.jsp
2. <u>Connect to Mysql:</u>
   - Links:
     - http://www.mysql.com

- Commands:
- Use browser: PhpmyAdmin interface http://127.0.0.1/phpmyadmin/ with credentials:
  - Username: testuser
  - Password: testpw
  - Administrator: root
  - Password: toor

3. <u>Use of Phoenix:</u>
   - *Links*:
     - https://phoenix.apache.org/language/datatypes.html
     - http://www.hadooptpoint.org/filters-in-hbase-shell/
     - https://forcedotcom.github.io/phoenix/
   - Commands:
     - To create a Phoenix tabel from command line:
       - go to the directory: "cd /home/ubuntu/Desktop/Utils/ create_table.sql"
       - run the command to open the file "nano create_table.sql"
       - run the command to execute the query in the file: "psql.py localhost create_table.sql"
     - To manage tables from command line:
       - sqlline.py 127.0.0.1:2181:/hbase
       - !tables
       - drop TABLE test_hbase;
       - select serviceID from Electric_vehicle_charging;
4. <u>Use of Karma data integration:</u>
   - *Links*:
     - http://usc-isi-i2.github.io/karma/
   - *Version*: 2.024
   - *Commands*:
     - To start Karma: run "mvn -Djetty.port=9999 jetty:run" from the directory "cd programs/Web-Karma-master/karma-web"
   - *Web interface*:
     - http://localhost:9999

5. <u>Penthao Kettle (in particular the graphic tool Spoon):</u>
   - *Links:*
     - www.pentaho.com/
     - https://wiki.pentaho.com/display/EAI/Pentaho+Data+Integration+Steps
     -
   - *Version:* Pentaho Data Integration (PDI) ver. 7.0

- *Commands*:
  - To start Spoon, run the following command from any directory on the virtual machine : spoon.sh

3 Nomenclature rules

Each ETL has a set of Nomenclature rules to be respected. In the more complex case, where a dataset has both the static and the realtime (or periodic / dynamic) parts, the rules are as follows:

- Main folder: datasetName_originalDataFormat
  - Folder **'Static'** (contains the ETL process of static data)
    - Folder **'Ingestion'** (mandatory)
      - **'Main.kjb'**
      - N_files.ktr (an undefined number of ktr files of Pentaho transformations)
      - M_files.kjb (an undefined number of kjb files of Pentaho jobs)
    - Folder **'QualityImprovement'** (optional)
      - **Main.kjb**
      - N_files.ktr (an undefined number of ktr files of Pentaho transformations)
      - M_files.kjb (an undefined number of kjb files of Pentaho jobs)
    - Folder **'Triplification'** (mandatory)
      - **Main.kjb**
      - N_files.ktr (an undefined number of ktr files of Pentaho transformations)
      - M_files.kjb (an undefined number of kjb files of Pentaho jobs)
  - Folder **'Realtime'** (contains the ETL process of real-time or periodic data)
    - Folder **'Ingestion'** (mandatory)
      - **Main.kjb**
      - N_files.ktr (an undefined number of ktr files of Pentaho transformations)
      - M_files.kjb (an undefined number of kjb files of Pentaho jobs)
    - Folder **'Triplification'** (optional**, ONLY in special cases, if required**)

Example:

- PisaBikeSharing_kmz
  - Static
    - Ingestion
      - Main.kjb
      - getAPI.ktr
      - Ingestion.kjb
      - Database.ktr

          - …
      - QualityImprovement
          - Main.kjb
          - Qi.ktr
          - ….
      - Triplification
          - Main.kjb
          - …
  - Realtime
      - Ingestion

4 Static data ingestion

The acquisition process for this type of data includes the following three phases:

- *Ingestion*: the data is downloaded, for example, from a web server or from a file and uploaded to HBase.
    - It is necessary to follow the nomenclature rules to store the data in the file system and on HBase. The file system storage serves only as a further control, the one on HBase allows the real management of data..
    - *Example of storing datasets (as they are made available by the providers) in the file system, or the path:*
        - Sources/Servizi/PisaBikeSharing_kmz/2017/12/19/15/3938 (for more details, see chapters 5 and 6)
    - *Example of HBase table*: "**PisaBikeSharing**" with family 'Family1'
- *Quality Improvement*: the data is improved and completed (the data just inserted in the HBase table is improved and saved on a new table, for example: streets with uppercase characters, additions of coordinates if missing, correction url / numeric numbers / phone numbers or other)
    - *Example of HBase table* : "**PisaBikeSharing_QI**" with family 'Family1'
- *Triplification*: the output is performed as triples in a file.n3, referring to the km4city ontology
    - It is necessary to follow the nomenclature rules to store the data in the file system..

4.1 Static data acquisition

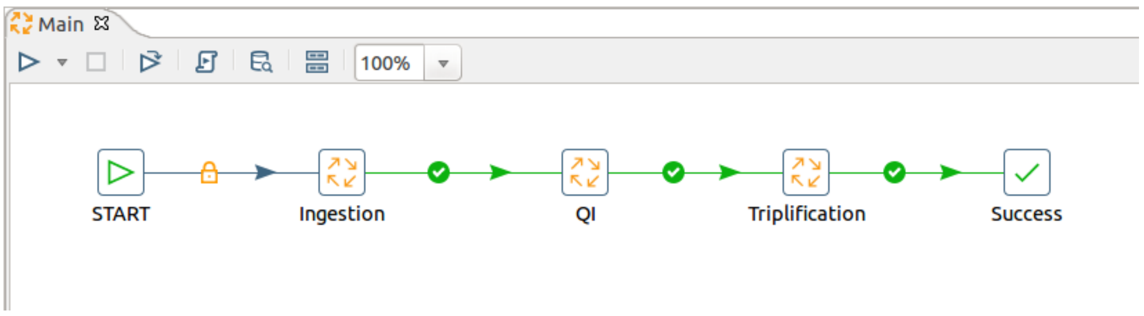


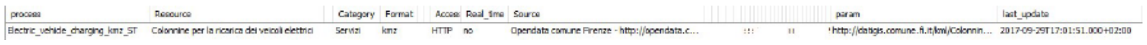
Figure 1: Main.kjb pipeline

In Figure 1, it is possible to appreciate the internal structure of the Main.kjb file which corresponds to a Spoon job. Jobs may contain other jobs, transformations or simple blocks. For example, the Ingestion job contains:

- The step 'start' (to start the ETL process)
- Three transformations (which correspond to the different data manipulation phases and which are described in detail in the following subsections):
    - Ingestion
    - QI
    - Triplification
- The step 'Success' (to terminate the ETL process)

This structure must be strictly manteined for all ETLs (this also facilitates the reuse of ETLs).

The first task to be performed BEFORE being able to start working on the ETL, is to define the input data, i.e. to update the MySQL table called 'process_manager2' of the 'Elaborato_Sis_Distr' database. This table contains one row for each ETL.



process	Resource	Category	Format	Acces:	Real_time	Source			param	last_update
Electric_vehicle_charging_kmz_ST	Colonnine per la ricarica dei veicoli elettrici	Servizi	kmz	HTTP	no	Opendata comune Firenze - http://opendata.c...	:::	::	'http://datigis.comune.fi.it/kml/Colonnin...	2017-09-29T17:01:51.000+02:00

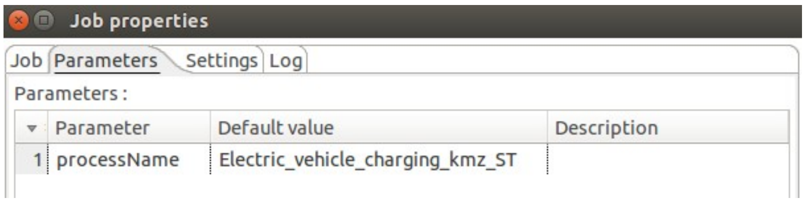
Figure 2: A row from the table *Elaborato_Sis_Distr.process_manager2*

As it possible to see in Figure 2, the key of the table is the 'process' column. The value assigned to this column is not random, it must show some information of the process itself such as: an identifier of the data, the format of the file and the type of the process. In this example we see "*Electric_vehicle_charging_kmz_ST*".

Other fields of primary importance are:

- *Resource: brief description of the data*
- *Category*: main directory (where the data will be downloaded and where the file.n3 will be saved)
- *Format*: file format (kmz, csv, xml, json, etc.)
- *Access*: protocol by which the file is obtained (es: ftp, http, etc.)
- *Real_time*: process type
- *Source*: file source
- *Param*: path to the file (es: http://dati.toscana.it/dataset/rt-oraritb/resource/ee55333c-fe53-4599-981d-389b13f28bb1)

In the "Main.kjb" job, it is necessary to set a processName as input parameter. This value must be the same of the key of the row inserted in the database (table 'process_manager2', column process). See Figure 3.



	Parameter	Default value	Description
1	processName	Electric_vehicle_charging_kmz_ST	

Figure 3: Main.kjb properties.

We proceed with the description of the single transformations that realize the static ETL: Ingestion, QI, triplification**.**

4.1.1 Ingestion

In this transformation we proceed in the following way: i) the dataset is acquired from the provider (for example downloading a file from the web, through ftp, doing crawling, etc.) and is saved in a file in the file system; ii) the data is read from the file just memorized; iii) extraction of relevant data; iv) storage of the data extracted on a HBase table. In Figure 4 it is possible to appreciate the pipeline of the

Main_Ingestion.kjb job, which contains other steps, jobs, transformations.

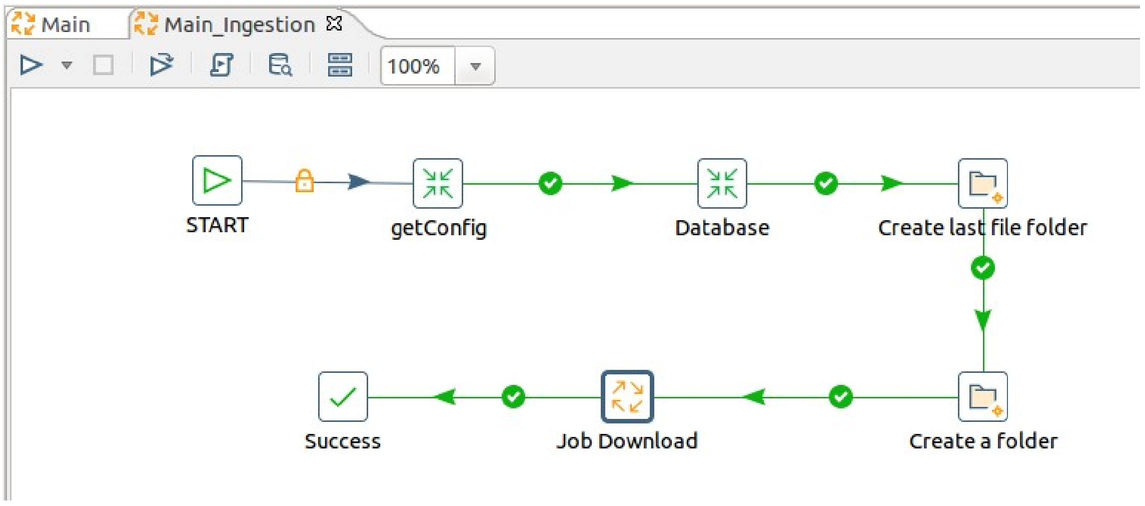


Figure 4: *Main_Ingestion.kjb pipeline*

The getConfig step simply loads a configuration file and sets global variables. These variables allow the use of relative rather than absolute paths. See Figure 5. The config.csv is located in the folder "/home/ubuntu/Desktop/Trasformazioni/".

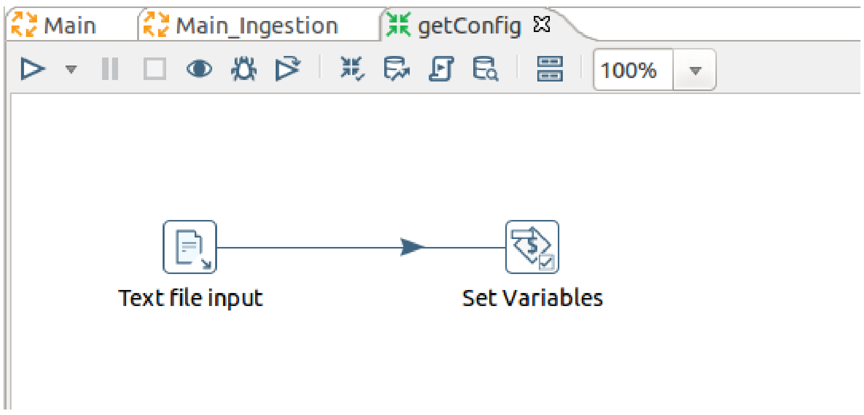


Figure 5: *getConfig.ktr pipeline*

The Database step (referring to Figure 4) is a Transformation Executor and calls the Database.ktr transformation which has the structure shown in Figure 6.

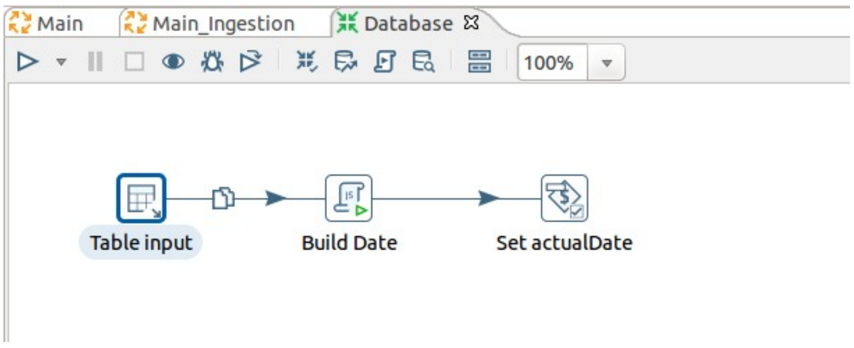


Figure 6: *Database.ktr* pipeline

The table input step (always referring to figure 6), retrieves from the MySQL table process_manager2 all the fields related to the running process through a SQL query.

The Build Date step (in figure 6) is a block in which it is possible to write Javascript code. In this case, starting from the current date are created some fields to construct the path in which it is possible to store the downloaded file. Then in the SetactualDate step the fields defined above are saved into variables.

In the Create last file folder and Create a folder steps (see Figure 5) the folders that will host the source file and the '1LastFile' folder are created. In ‘1LastFile’ a copy of the latest version of the downloaded file will be stored.

Finally, Figure 7 shows the Job Download.

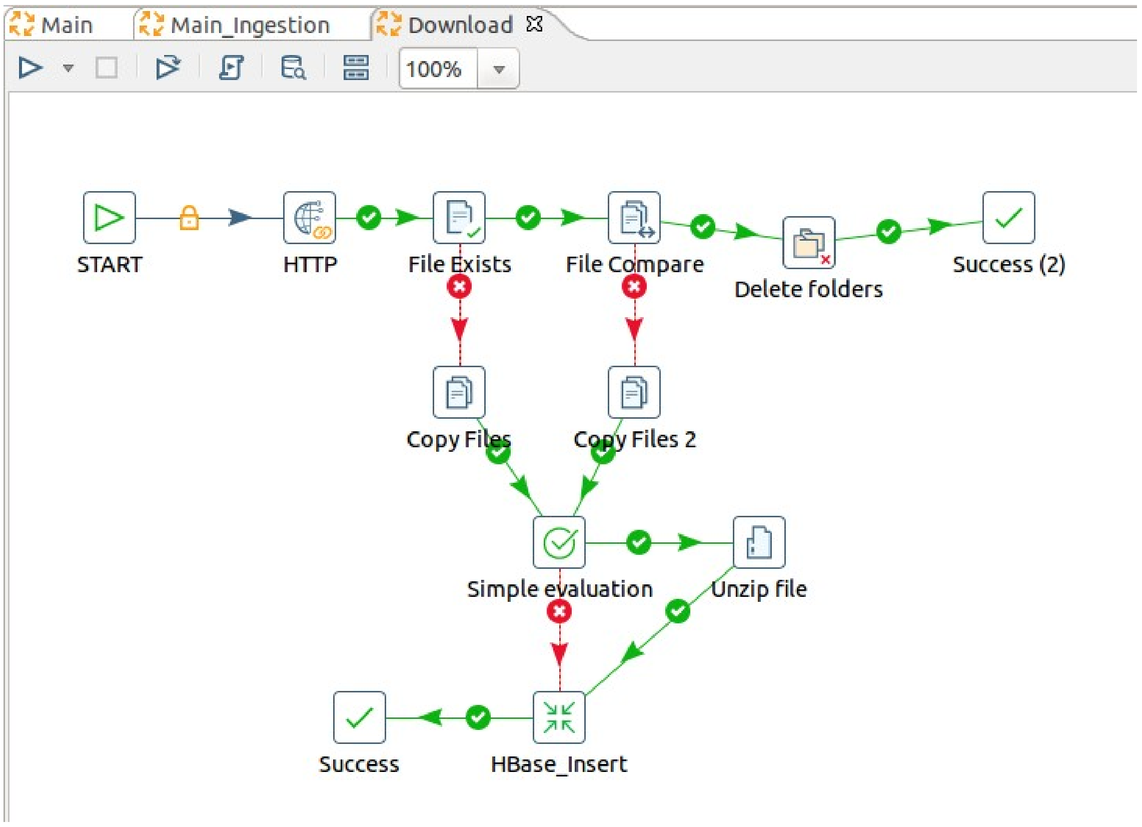


Figure 7: *Download.kjb pipeline*

In the HTTP step the fields are set: 'URL', as the address from which to download the file of interest; 'Target', as a path to save this file (obviously the paths are relative, i.e. the previously set variables are used). Then a check is made: if the source file containing the data just downloaded already exists within the **1LastFile** folder, a comparison is made with the one just downloaded (compare the current data with those previously downloaded, in case the ETL has already been started).
If the two files are identical, the newly created folders are deleted and the job ends in Success 2, this means that the data made available by the Data Provider have not changed and that we already have the updated data (so there is NOT need to analyze them again). Otherwise the copy within the 1LastFile folder is updated with the new data. In the example we proceed by unzipping the file and inserting the data into a HBase table.
Figure 8 shows the transformation pipeline. The data in this case is read from a file.kml, it is "cleaned up" and finally it is inserted in HBase through the HBase Output step. This table must be called *Electric_vehicle_charging_ST*.

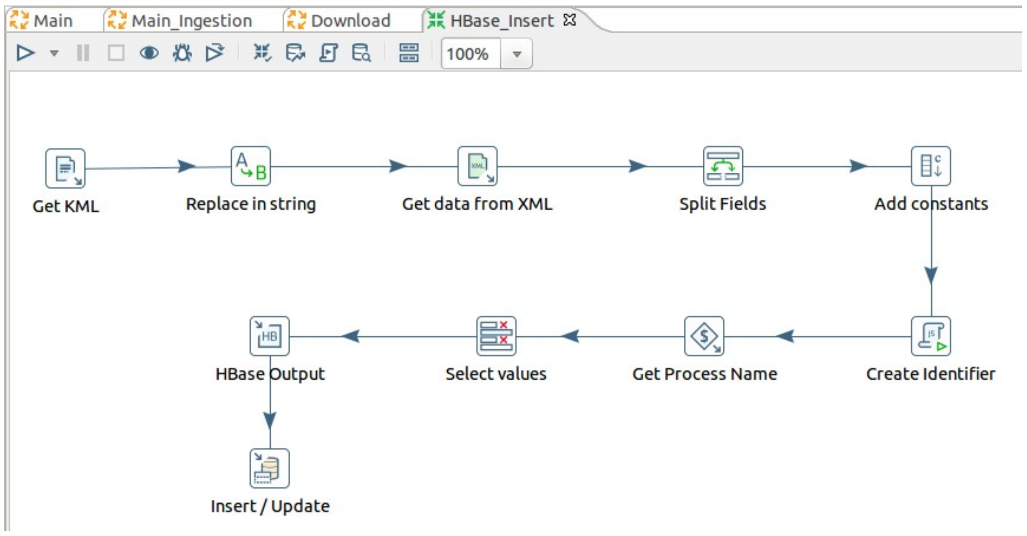


Figure 8: *HBase_Insert.ktr pipeline.*

4.1.2 Quality Improvement

The Quality Improvement process aims to improve the quality of the data acquired in the ingestion phase, making appropriate changes to the data and storing it in a new HBase table. The structure of the Job Main_QI is shown in Figure 9.

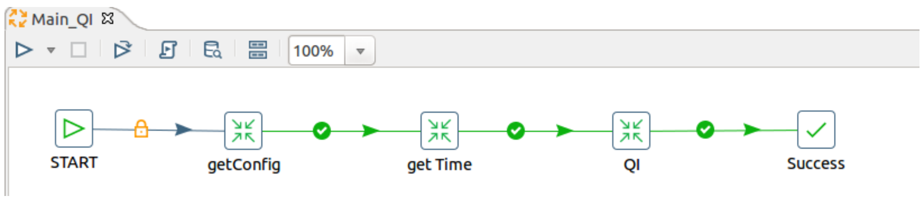


Figure 9: *Main_QI.kjb pipeline*

The QI.ktr transformation is in charge of add fundamental properties to the data. A SPARQL query extracts information such as: coordinates (latitude and longitude), streets, house numbers, cities, zip codes, but above all the toponym. See Figure 10.

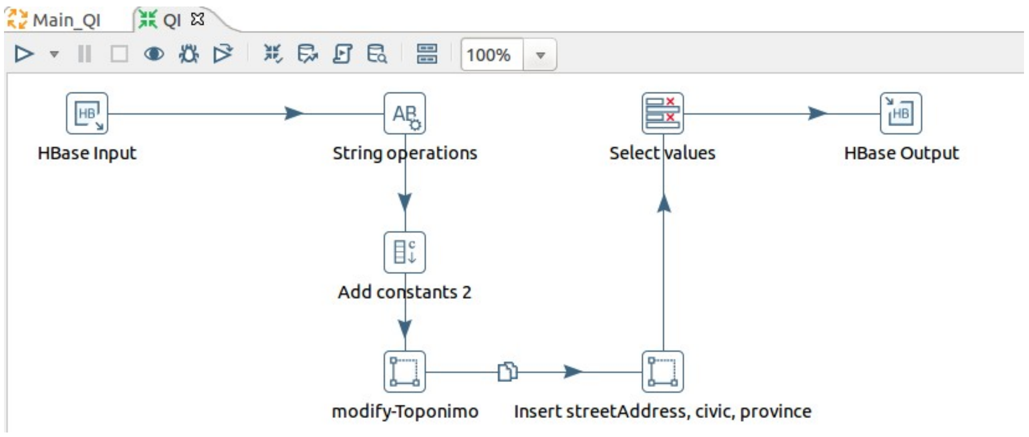


Figure 10: *QI.ktr pipeline*

The data is read from the Electric_vehicle_charging_ST table and is completed with the missing information. Through the Select Value step, the data set is cleaned that is the fields that are relevant to the system are selected. So, the data is ready to be saved on a new HBase table (ex: Electric_vehicle_charging_ST_QI, always with family equal to 'Family1').

4.1.3 Triplification

The Triplification task aims to generate a file.n3 containing a set of RDF triples starting from the data acquired and processed in the two previous phases. Each piece of information collected is linked to KM4City ontology (http://www.disit.org/km4city/schema), using the Karma tool.
Karma provides:

- The creation of a model that maps data on one or more ontologies (in our case in the KM4City multi-ontology). The mapping is done by linking the fields of ontologies with the data contained in MySQL table (therefore to generate the Karma model it is necessary to copy data from HBase to MySQL).
- The application of the model to the data through the tool Kitchen.
- The architecture (pipeline) of this ETL process is shown in Figure 11.

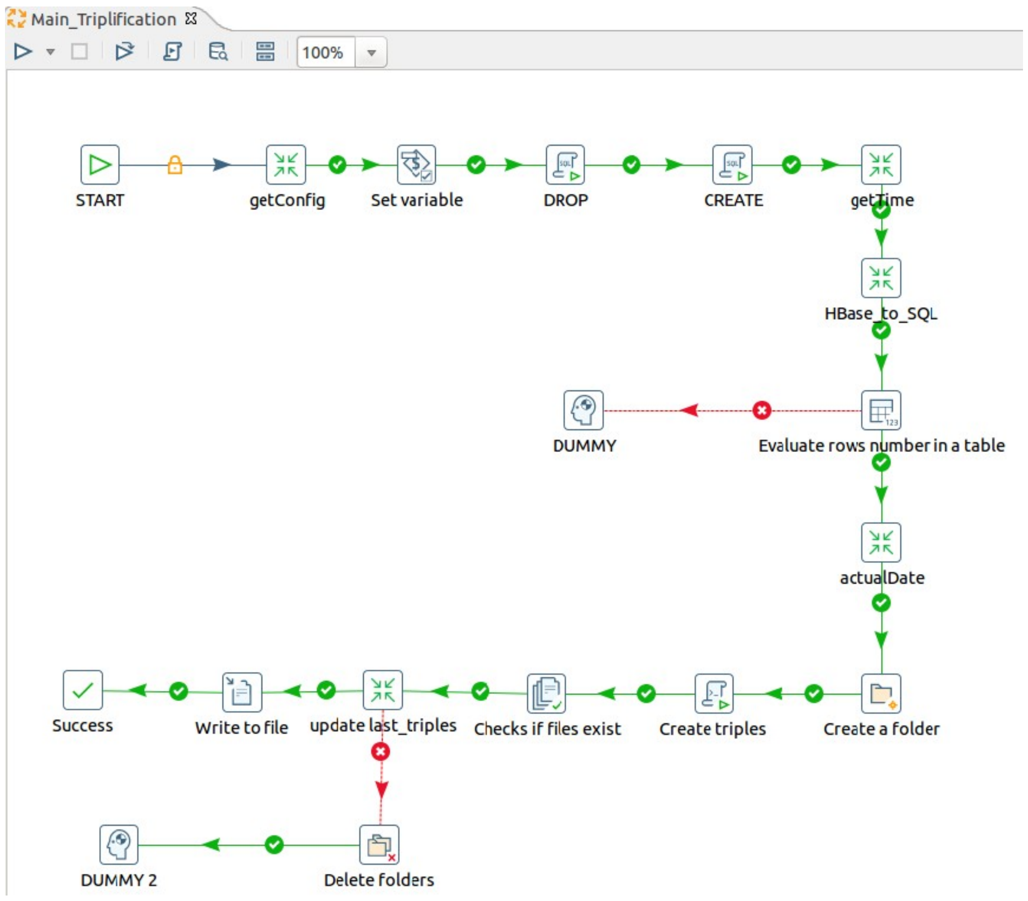
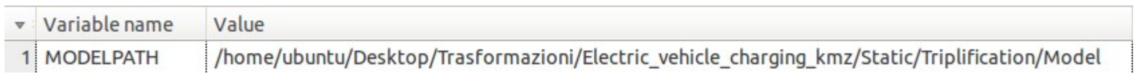


Figure 11: *Main_Triplification.kjb* pipeline.

The 'Set variable' step creates the MODELPATH variable that keeps the path to the model (the file containing the Karma model). The model must be saved in a folder called Model inside the Triplification folder. See Figure 12.



	Variable name	Value
1	MODELPATH	/home/ubuntu/Desktop/Trasformazioni/Electric_vehicle_charging_kmz/Static/Triplification/Model

Figure 12: *Set variable*

In the following steps, the information contained in the HBase table Electric_vehicle_charging_ST_QI is temporarily copied to another MySQL table Electric_vehicle_charging_kmz_ST. This operation is necessary because it was decided to use the Karma tool to realize the rdfs triples, and Karma works only with SQL.
The 'Create a folder' step creates the folder that will contain the file containing the triples. It is necessary to follow the nomenclature guidelines :

- the folder containing the file.n3 must have the same name as the process. This folder must be maintained in a group of nested directory where the root folder is called as the process category ('Category' of the MySQL table 'process_manager2'). The other folders must keep track of the date on which the triples were generated (i.e. Category/year/month/day/hour/second/processName).

- Example: TPL/Bus_ataflinea/2017/12/19/17/0001

The step '*Create triples*' realizes the triples while the following steps carry out some checks and validations.

4.2 RealTime data acquisition (periodic/dynamic)

The acquisition process includes a single phase during which the data is downloaded, processed and inserted into a HBase table. In Figure 13 it is possible to observe the internal structure of the Main.kjb file (the Min pipeline), which must be strictly maintained.

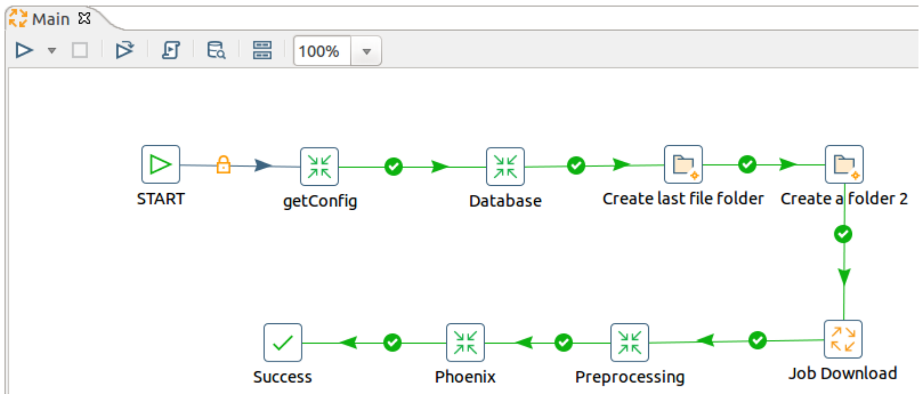
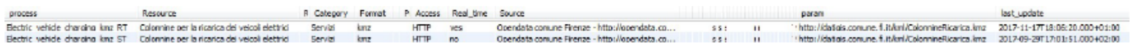


Figure 13: Main.kjb pipeline

As for the static data, the first operation to perform is to update the MySQL table 'process _manager2' of the Elaborato_Sis_Distr database. See Figure 14.



process	Resource	R	Category	Format	P	Access	Real_time	Source			param	last_update
Electric_vehicle_charging_kmz_RT	Colonnine per la ricarica dei veicoli elettrici		Servizi	kmz		HTTP	yes	Opendata comune Firenze - http://opendata.co...	s s :	i i	http://datigis.comune.fi.it/kml/ColonnineRicarica.kmz	2017-11-17T18:06:20.000+01:00
Electric_vehicle_charging_kmz_ST	Colonnine per la ricarica dei veicoli elettrici		Servizi	kmz		HTTP	no	Opendata comune Firenze - http://opendata.co...	s s :	i i	http://datigis.comune.fi.it/kml/ColonnineRicarica.kmz	2017-09-29T17:01:51.000+02:00

Figure 14: Some rows extracted from the table *Elaborato_Sis_Distr.process_manager2*

Also for this ETL, it is necessary to set the processName parameter within the job. Its value must be equal to the key of the row inserted in the database, i.e Electric_vehicle_charging_kmz_RT. The next steps are almost identical and have the same responsibilities. Figure 16 shows the structure of the job Download.kjb.

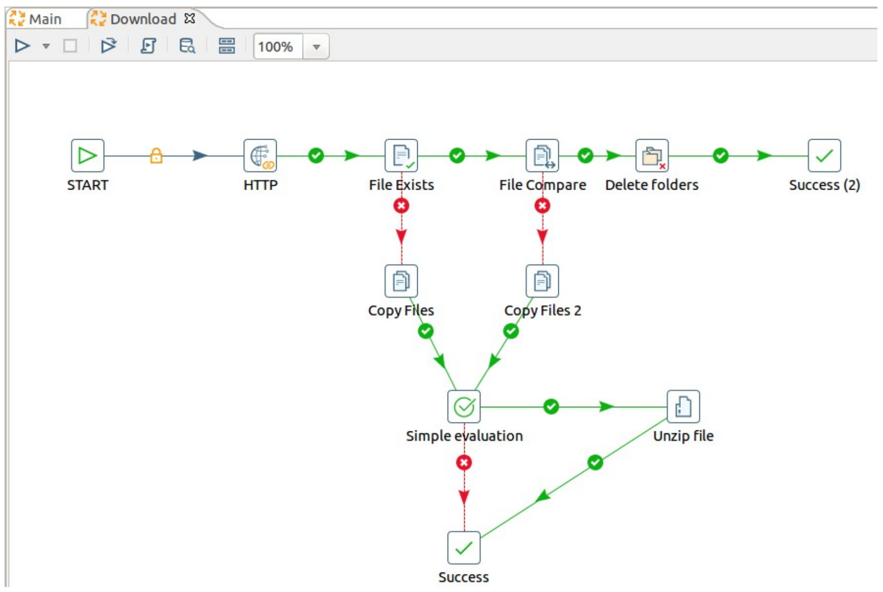


Figure 15: *Download.kjb pipeline*

The Preprocessing step in this example is in charge of selecting the RealTime information and setting an identifier for each row of the dataset. This identifier must be equal to the identifier used during the acquisition of the static data. **This field is fundamental because it connects the static and the RealTime data.** See Figure 16: the output is copied to the stream.

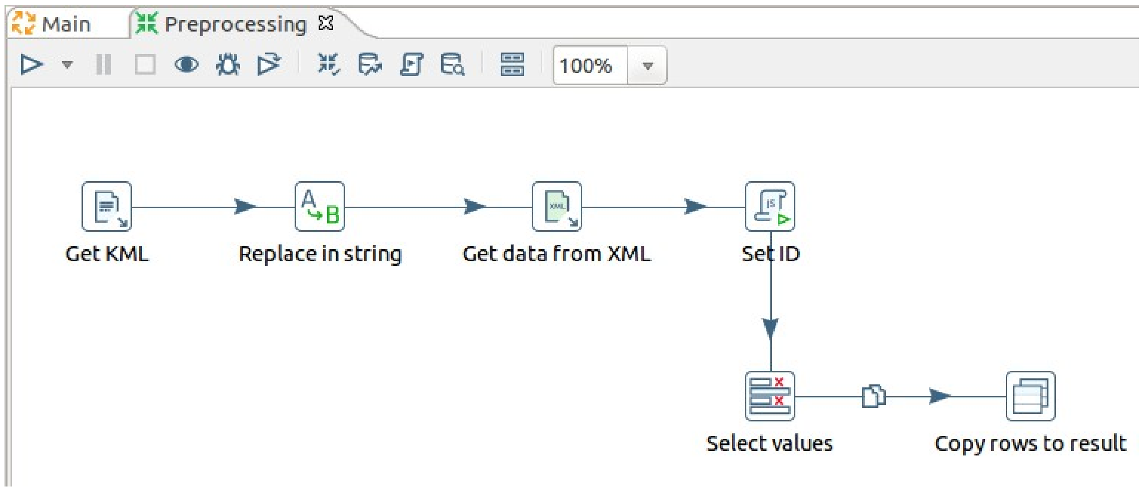
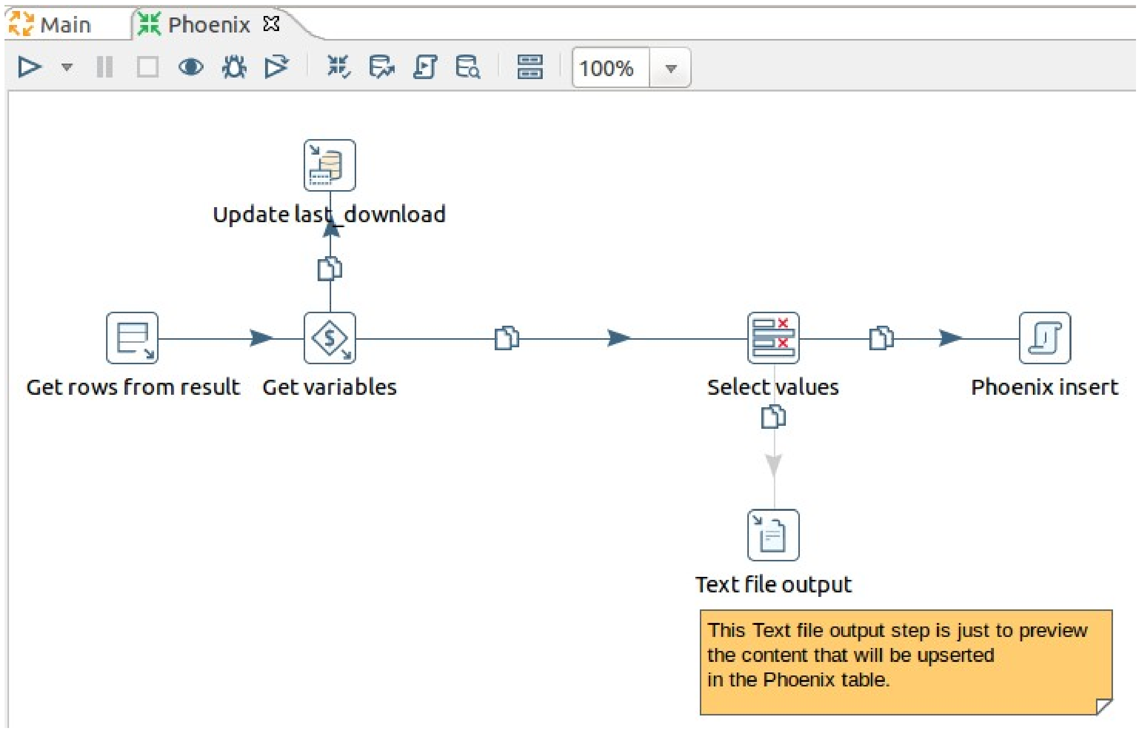


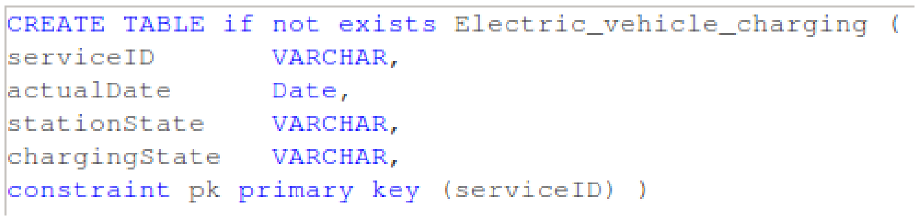
Figure 16: *Preprocessing.ktr pipeline*

The data is read from the stream and through the Phoenix insert step is inserted (using the upsert function) into a HBase table. See Figure 18. This table must be created from the command line. In Figure 18 an example can be observed.



This Text file output step is just to preview the content that will be upserted in the Phoenix table.

Figura 17: *phoenix.ktr pipeline*



```
CREATE TABLE if not exists Electric_vehicle_charging (
serviceID       VARCHAR,
actualDate      Date,
stationState    VARCHAR,
chargingState   VARCHAR,
constraint pk primary key (serviceID) )
```

Figure 18: Create the HBase table via command line