Smart City: data ingestion and mining

Parte 12 (2014-2015) of course on Collaborative and protection systems

Laurea in Ingegneria

Mariano di Claudio, Giacomo Martelli, Paolo Nesi, Nadia Rauch

DISIT Lab, Dipartimento di Ingegneria dell'Informazione, DINFO
Università degli Studi di Firenze
Via S. Marta 3, 50139, Firenze, Italy
tel: +39-055-2758515, fax: +39-055-2758570

http://www.disit.dinfo.unifi.it ***alias*** http://www.disit.org
Prof.Paolo Nesi, paolo.nesi@unifi.it

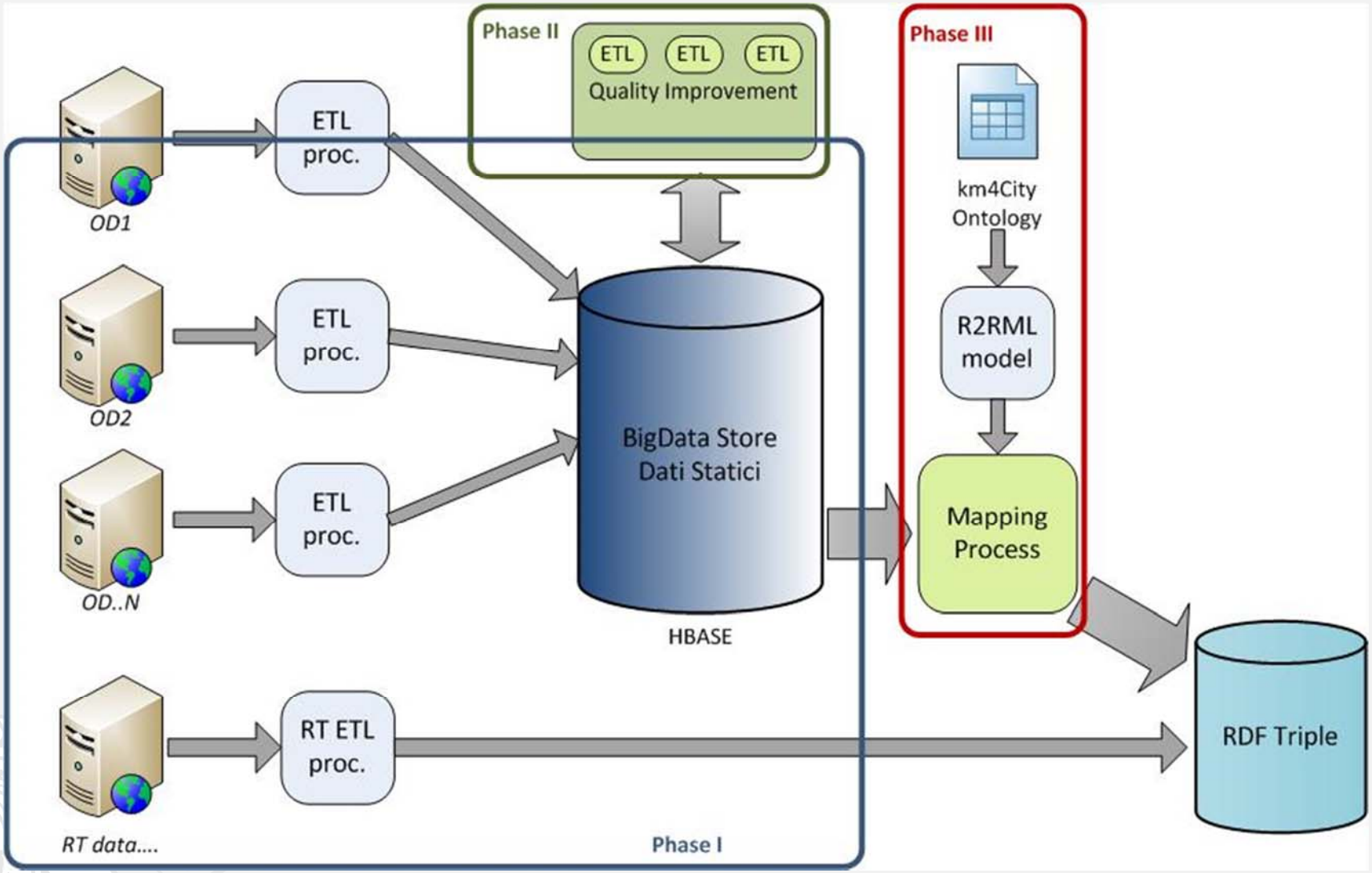
Index

1. From Open Data to Triples
2. ETL process
3. ETL tool: Pentaho Data Integration (PDI)
   - Features
   - Key concepts
   - Examples
4. SiiMobility Project

1

From Open Data to Triples

Data Engineering Architecture



Phase I: Data Ingestion

- **Ingesting a wide range of OD/PD**: open and private data, static, quasi static and/or dynamic real time data.
- **Static and semi-static data** include: points of interests, geo-referenced services, maps, accidents statistics, etc.
  - files in several formats (SHP, KML, CVS, ZIP, XML, etc.)
- **Dynamic data** mainly data coming from sensors
  - parking, weather conditions, pollution measures, bus position, etc..
  - using Web Services.
- Using **Pentaho - Kettle** for data integration (Open source tool)
  - using specific ETL Kettle transformation processes (one or more for each data source)
  - data are stored in HBase (Bigdata NoSQL database)

Phase II: Data Quality Improvement

- **Problems kinds:**
  - Inconsistencies, incompleteness, typos, lack of standards, multiple standards, ..
- **Problems on:**
  - CAPs vs Locations
  - Street names (e.g., dividing names from numbers, normalize when possible)
  - Dates and Time: normalizing
  - Telephone numbers: normalizing
  - Web links and emails: normalizing
- **Partial Usage of**
  - Certified and accepted tables and additional knowledge

Phase III: Data mapping

- Transforms the data from HBase to RDF triples
- Using **Karma Data Integration tool, a mapping model** from SQL to RDF on the basis of the ontology was created
  - Data to be mapped first temporary passed from Hbase to MySQL and then mapped using Karma (in batch mode)

- The mapped data in triples have to be uploaded (and indexed) to the **RDF Store (OpenRDF – sesame with** OWLIM-SE)

2

ETL Process

Useful tools

Pre-processing data to RDF triples generation: ETL (Extract, Transform and Load)

– Process used in database and data warehousing that involves three phases.

– Useful tools to prepare data for the following data analysis phase and eventual translation into RDF.

– Translation in **RDF Triples** is based on use of a specific **ontology**.

- Definition of mapping models from SQL to RDF
- The triples generated are loaded on OWLIM RDF store.

ETL Process

The three phases are:

- **Extracting** data from outside sources (**Ingestion** phase).

- **Transforming** it to fit operational needs, which can include quality levels (**Data Quality Improvement** phase).

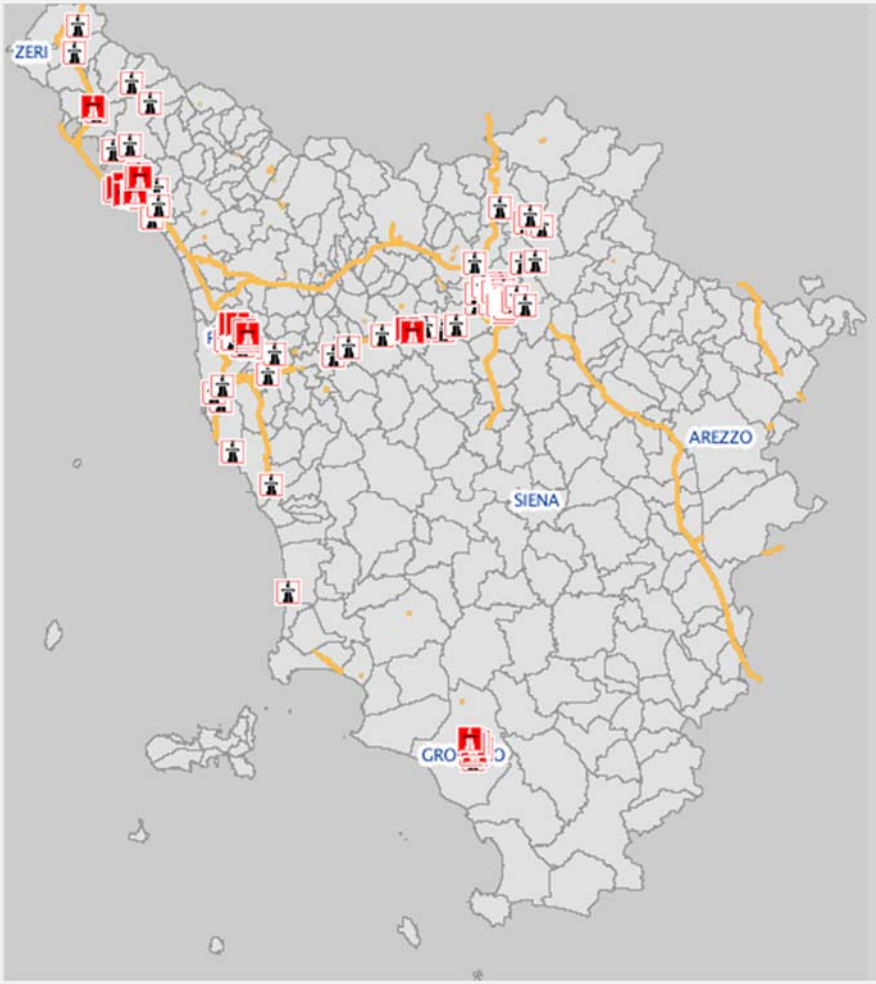
- **Loading** it into the end target (database, operational data store, data warehouse, data mart, etc…..). So the data can be translated in **RDF triples using a specific ontology.**

Datasets

- **Mobility data** are made available by MIIC (Mobility Information Integration Center)
  - MIIC is a service of the Tuscany regional authority that deals with the collection of infomobility data (from federal authorities) and their distribution via web services.
  - Web services expose data about: traffic, parking, AVM (Automatic Vehicle Monitoring), emergencies and weather information.

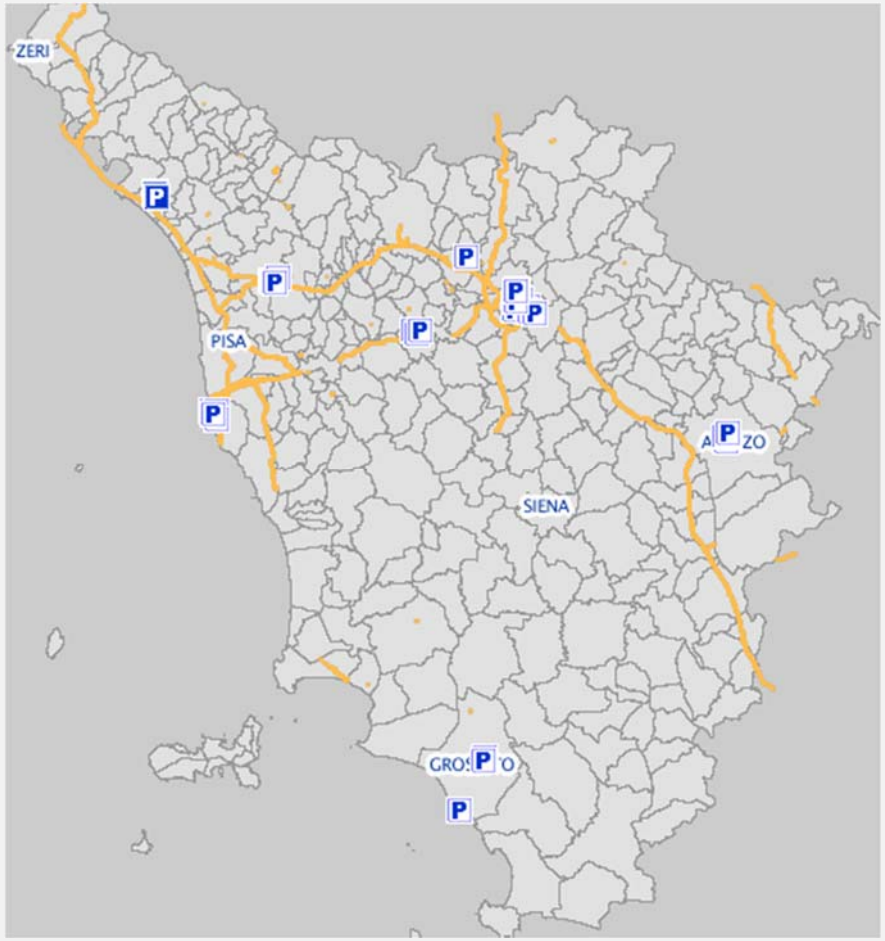
Example of Dataset: MIIC

- **Traffic sensors**: data about the situation of the traffic report from sensor detection systems operators
  - The measurements include data such as average distance between vehicles, average speed of transit, percentage of occupancy of the road, transit schedules, etc.
  - The sensors are divided into groups identified by a catalog code that is used when invoking the web service
  - A group is a set of sensors that monitors a road section
  - The groups produce a measurement every 5 or 10 minutes



Example of Dataset: MIIC

- **Parking**: data on the status of occupancy of the parking from parking areas operators.
  - The status of a parking is described by data such as the number of places occupied, the total number of vehicles in and out, etc.
  - The parking are divided into groups identified by a catalog code that is used when invoking the web service
  - A group corresponds to the collection of parking owned by a municipality
  - The situation of each parking is published approximately every minutes



Example of Dataset: MIIC

- **AVM:** real-time data about local public transport of Florence metropolitan area equipped with AVM devices
  - Monitors the status of active rides in the territory, where a ride is the path that a vehicle runs from a start point to the end
  - The data provided are related to delay or advance state of a vehicle, location vehicle in GPS coordinates, information about the last stop made and programmed, etc.
  - The web service is invoked passing the identification code of the race as a parameter.
  - AVM devices send two types of messages, one at a programmed time, usually every minute, and one at a major event like the arrival at a stop, departure from a stop or interruption of service.

Example of Dataset: MIIC

- **Static data**: data updated infrequently that can enrich those in real time.
  - Are provided by the regional observatory for mobility and transport through a portal with graphical user interface.
  - Positional information about parking and sensors surveyed by MIIC .
  - Additional details on public transport network: description and details of lines, routes and stops, Geolocation with Gauss-Boaga coordinates of stops

Example of Dataset

- **Weather forecasts** provided by LaMMA
  - XML Format
  - Information about current day
  - Weather on the current day and the next 4 days
  - Forecast on five times of the day: morning, afternoon ...
- **Services** of the Tuscany region
  - CSV Format
  - Various services: banks, schools, food, hospitals, shops, theatres, museums and. ..
  - Geolocation by address (Street, house number) and municipality of belonging
  - Contains the service name, address, city, State, type of service, phone number, email …

Example of Dataset

- **Statistics** on the Florence municipality
  - CSV Format
  - Contain information on the town and on the streets of Florence: crashes, tourist arrivals, circulating vehicles, etc..
  - The statistics shall cover the last five years
- **Tram line**
  - KMZ Format
  - Contains the KML file format used for geospatial data managing in Google earth and Google maps
  - Contains the coordinates of the path covered by tram line

3

ETL tool: Pentaho Data Integration (PDI)

FEATURES

Pentaho Data Integration (PDI)

- **Pentaho** is a framework that contains several packages integrated to allow complete management:
  - ***Business Intelligence problems;***
  - ***Data Warehouse problems;***
  - ***Big Data problems.***

- **Kettle** is the ETL component Pentaho for data transfer and processing.

Pentaho Data Integration (Kettle)

- Free, **open source** (LGPL) ETL (Extraction, Transformation and Loading) tool.
  - It is available also in **enterprise version.**

- **Developed in Java**, therefore is guaranteed the compatibility and portability with the major operating systems (Windows, Linux, OS X..).

- **Powerful** Extraction, Transformation and Loading (ETL) capabilities.

Pentaho Data Integration (Kettle)

- **Scalable**, standards-based architecture.
- Opportunity to interfacing with the main NoSQL Databases (Hbase, Cassandra, MongoDB, CouchDB…).
- It uses an innovative, **metadata-driven** approach.
- Graphical, **drag and drop** design environment.

Pentaho Data Integration (Kettle)

Main strengths:

- Collect data from a **variety of sources** (extraction);
- Move and modify data (transport and transform) while cleansing, denormalizing, aggregating and enriching it in the process;
- Frequently (daily) store data (loading) in the final target destination, usually a **large** dimensionally modeled **database** (or **data warehouse**).

Pentaho Data Integration (Kettle)

Main weakness:

- Kettle is not able to transform data into RDF triples, therefore it is necessary use other tools at a later stage (Karma).

Kettle's 4 main programs

- **Spoon:** graphically oriented end-user tool to model the **flow of data** from input through transformation to output (**transformation**).
- **Pan** is a **command line tool** that executes transformations modeled with Spoon.
- **Chef**: a graphically oriented **end-user tool** used to model **jobs** (transformations, FTP downloads etc. placed in a flow of control).
- **Kitchen** is a **command line tool** to execute jobs created with Chef.

Kettle's 4 main programs

- Interesting feature: Kettle is **model-driven**.
- **Spoon** and **Chef** have a graphical user interface to define the ETL processes on a **high level**.
- **Pan** and **Kitchen** can read and interpret the models created by Spoon and Chef respectively.
- Models can be saved to a particular **XML format**, or they can be stored into a relational database (**repository**).
- Handling many models with repository: models are stored in a structured manner, arbitrary queries can be written against the repository.

3

ETL tool: Pentaho Data Integration (PDI)

KEY CONCEPTS

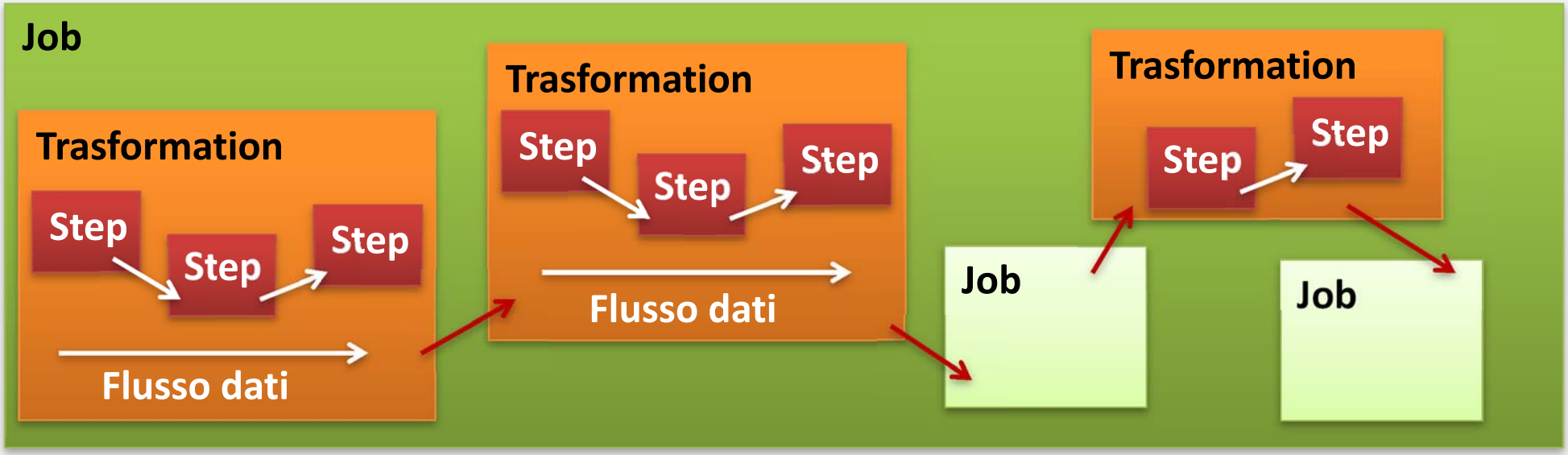
Kettle: Concepts

- Kettle is based on two key concepts (from operating point of view):
  - **Job** (with extension ".kjb");
  - **Transformation** (with extension ".ktr"), composed of several **steps**.

- Kettle's key components are:
  - **Spoon** for ETL process modeling;
  - **Pan** to execute the transformations from command line;
  - **Kitchen** to execute the Job from command line.

Kettle: Operational structure

Kettle operating components are organized as follows:



- The data are seen as rows flow from one step to another one.
- The steps are parallel executed on separated threads and there is no necessarily a beginning or end point of transformation.
- A job manages the sequential execution of lower-level entities: transformations or other jobs.

Spoon Concepts: Steps and hoops

- One **step** denotes a particular kind of **action** that is performed **on data**.
- **Hops** are links to connect steps together and allow data to pass from one step to another.
- Steps are easily created by **dragging** the icon from the treeview **and dropping** them on the graphical model view.
- Kettle provides a lot of different step types, and can be **extended with plugin**.

Type of Steps in Spoon (1/2)

Three different kinds of steps: **input**, **transform**, **output**.

- **Input steps** process some kind of 'raw' resource (file, database query or system variables) and create an output stream of records from it.

- **Output steps** (the reverse of input steps): accept records, and store them in some external resource (file, database table, etc.).

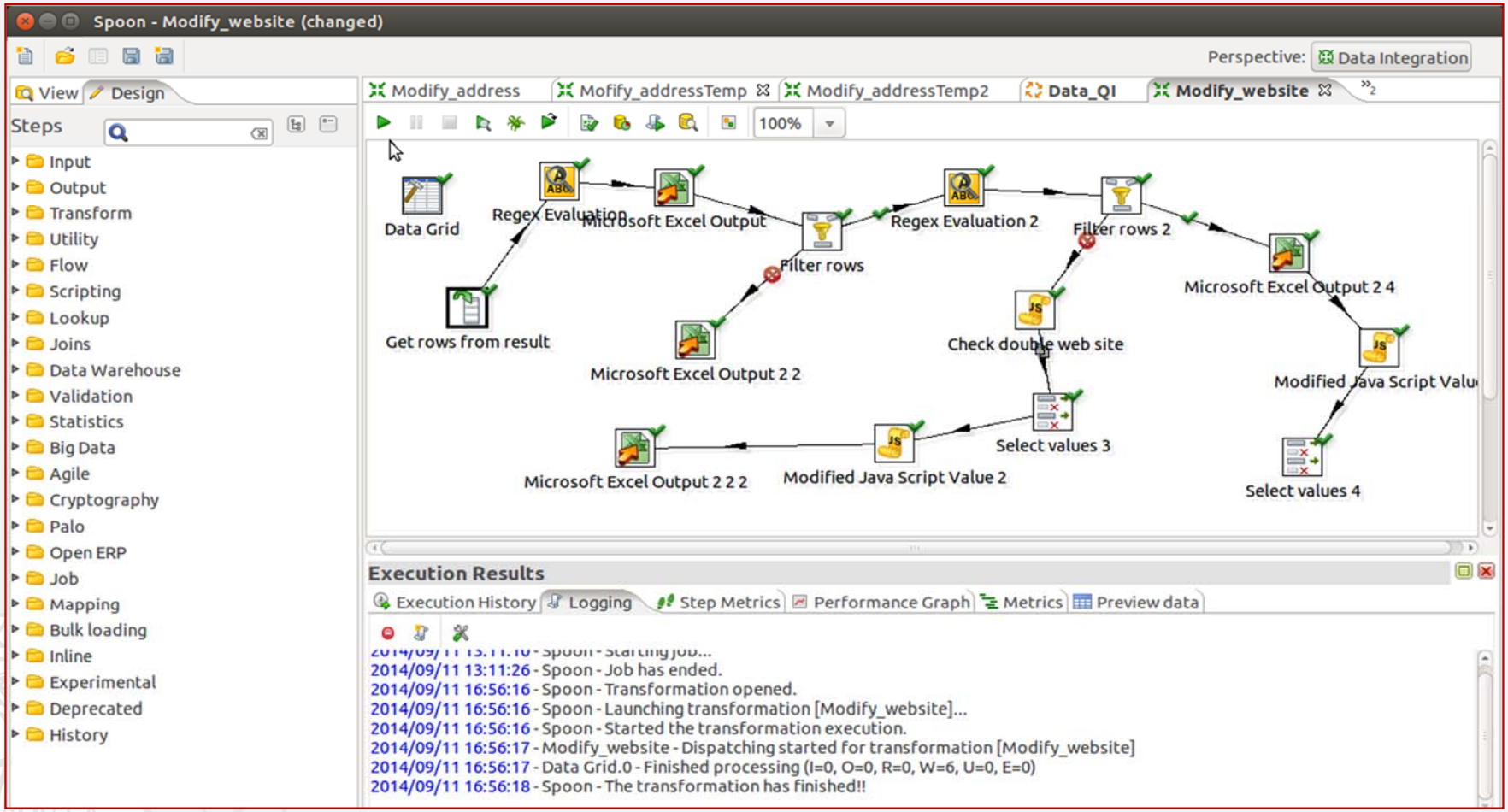
Type of Steps in Spoon (2/2)

- **Transforming steps** process input streams and perform particular action on it (adding new fields/new records); This produce one or more output streams. Kettle offers many transformation steps out of the box, very simple tasks (renaming fields) and complex tasks (normalizing data, maintaining a slowly changing dimension in a data warehouse).

- **Main.kjb** is usually the primary job.

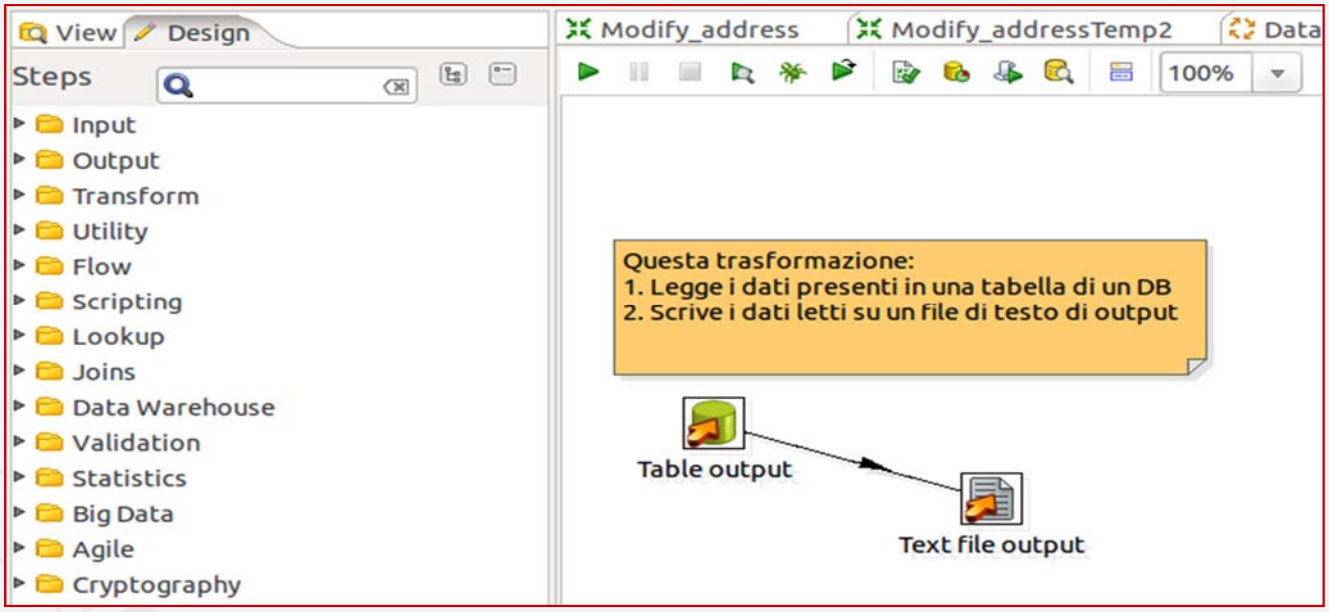
Kettle: Spoon

To run Spoon, just launch the instruction **./spoon.sh** from command line.



Kettle: Transformations

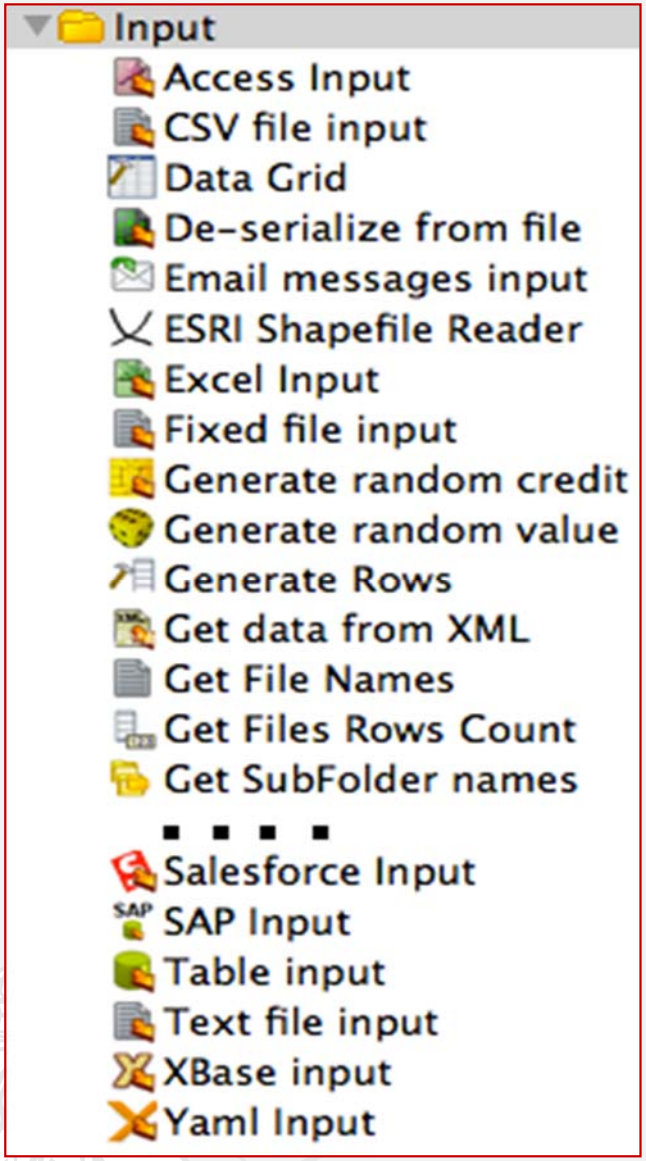
- Transformations define how the data must be collected, processed and reloaded.
- Consist of a series of step connected by links called Hop.
- Typically a transformation has one **input step**, one or multiple **transformation steps** and one or more **output step**.



Kettle: Transformations

- There are several possible steps organized by type: ***Input, Output, Utility, Scripting, Flow, Validation, Lookup, Statistics***…etc.
- Each type of step, in turn, offers several of possibilities.
- For example, an input step can take data from different sources:
  - ***From a table of a relational database;***
  - ***From CSV file;***
  - ***From MS-Excel sheet.***
- The Hops between two steps don't define the execution sequence but represent the data flow and allow passing the content of a Field from one step to the next one.

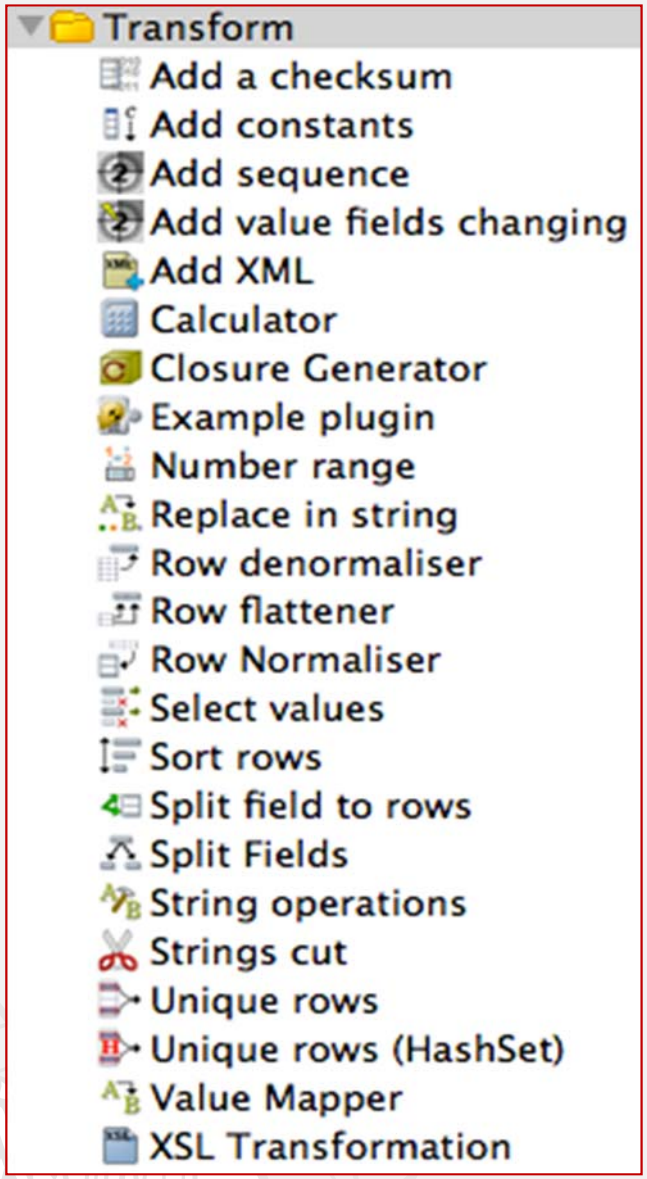
Kettle: Transformations



Input

Collection of step dealing with input data management. They are present in various types.

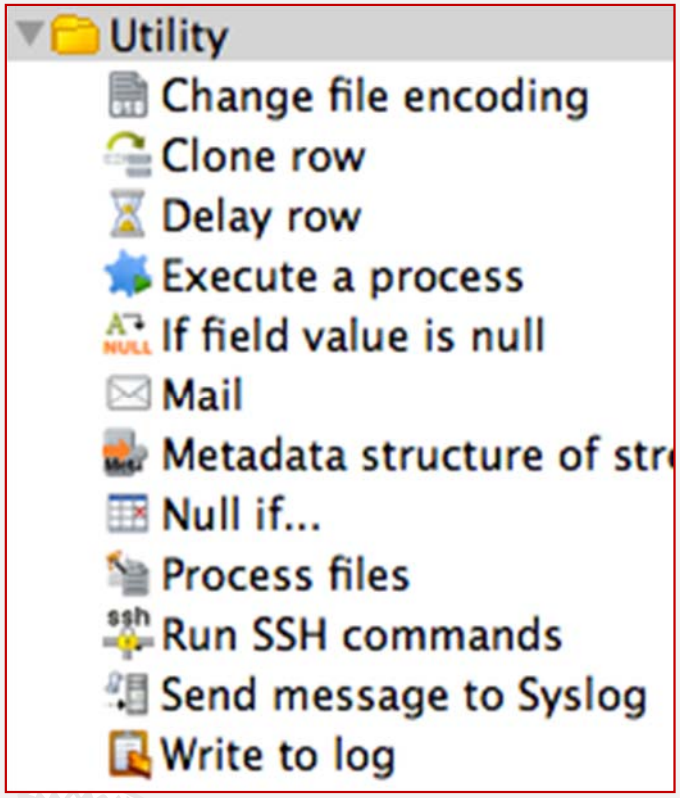
Kettle: Transformations



Transform

collection of step dealing with realize data transformation: i.e. trim, fields separation, strings truncation, rows sorting, etc.....

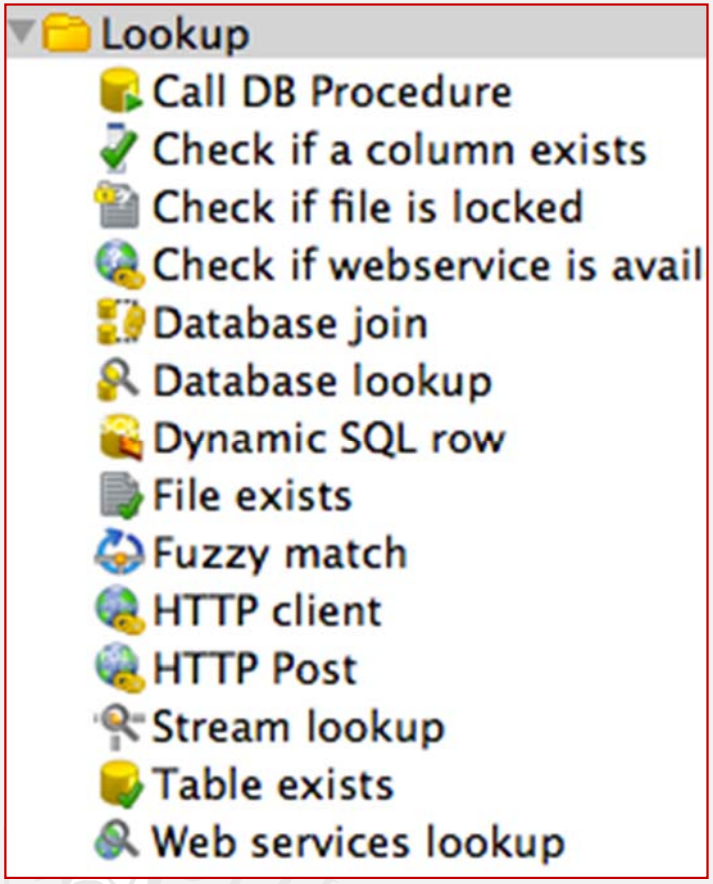
Kettle: Transformations



Utility

collection of step with advanced or supporting features: i.e. log writing, check if a field is null, rows deletion, etc….

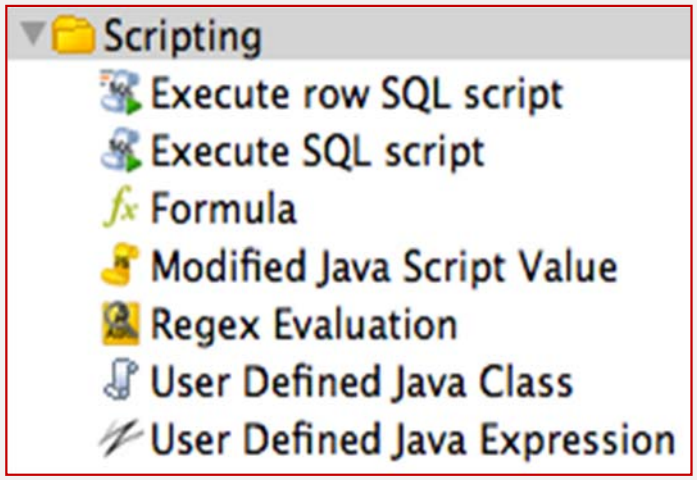
Kettle: Transformations



Lookup

collection of step allowing data consultation operations on specific solutions or on data already extrapolated and kept in temporary structures (to increase speed and reactivity).

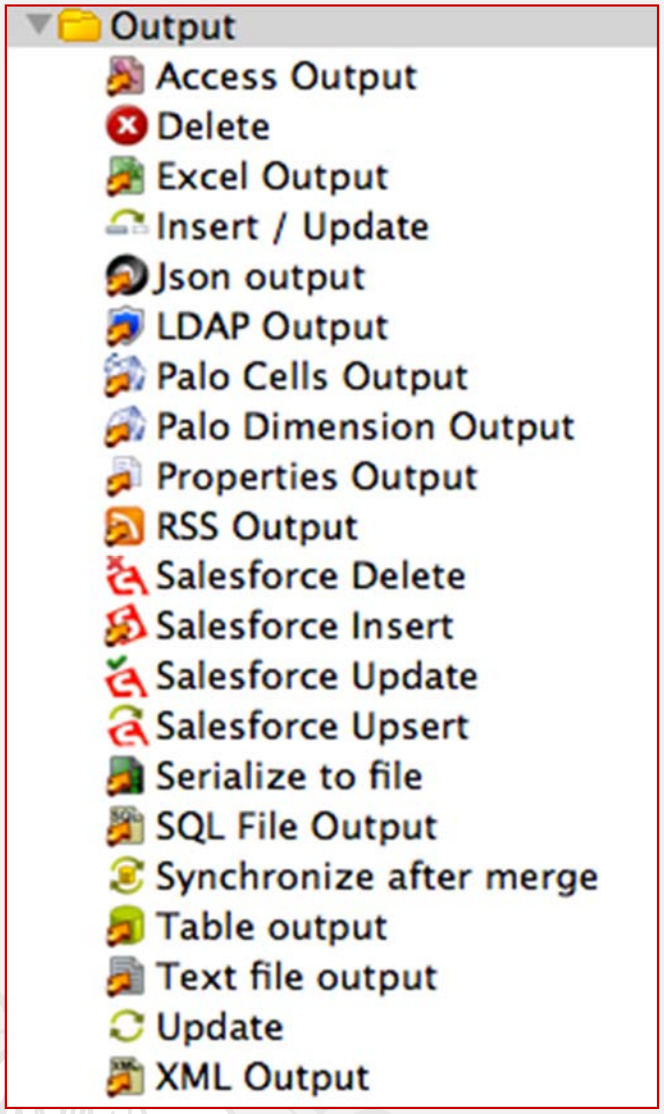
Kettle: Transformations



Scripting

set of step in which you can define scripts in different languages (SQL, JavaScript, etc…).

Kettle: Transformations



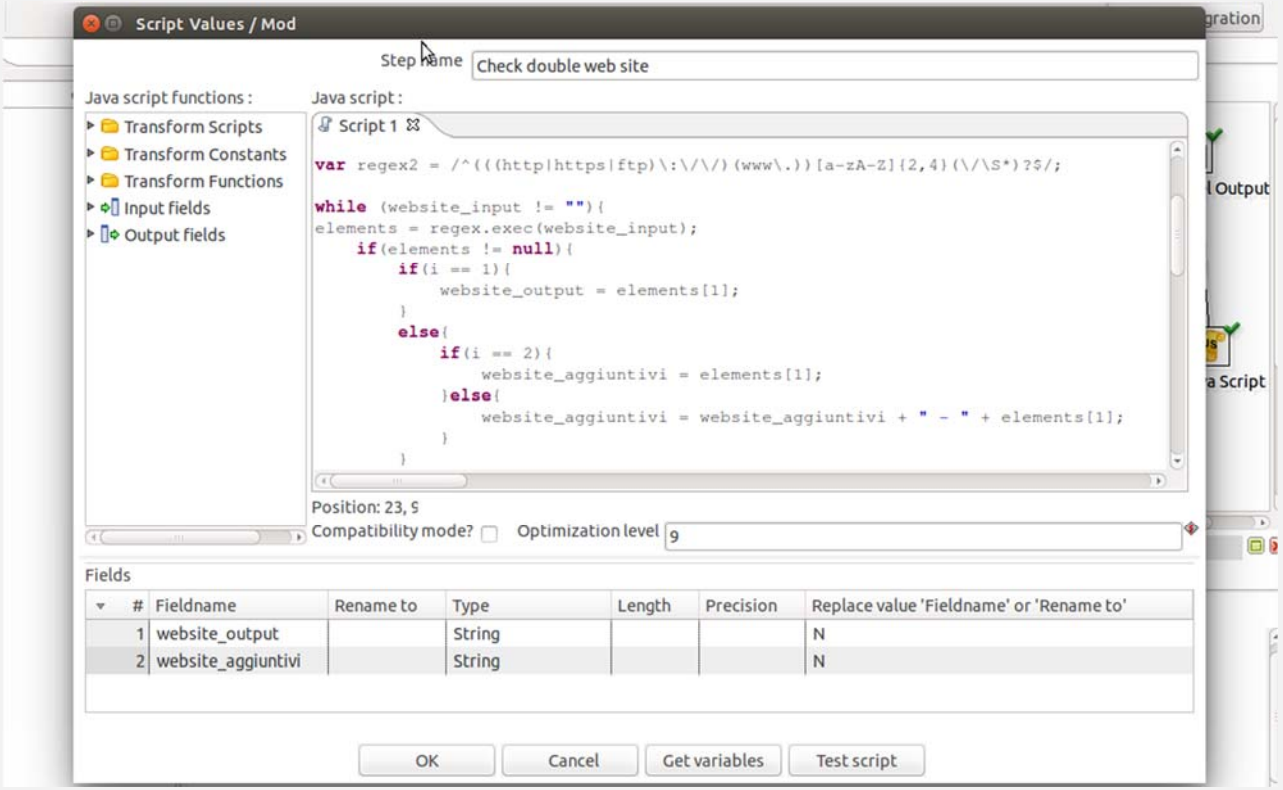
Output

Collection of step dealing with ouput data management. They are present in various types.

Kettle: Transformations

Kettle offers many types of steps to execute various data operations, also it offers:

- ***possibility of use and add some JavaScript code.***
- ***possibility of use regular expressions.***



Kettle: Pan e Kitchen

- The Transformations made with Spoon can be executed with Pan from command line (similarly Kitchen for the Job).

```
/usr/local/pdi/pan.sh -file /home/pentaho/repos/LetturaDati.ktr
```

```
# Lancia il job ogni sabato alle sei di mattina...
6 6 * * 6 /usr/local/pdi/kitchen.sh -file /home/pentaho/repo/Aggiorna1.kjb >> /tmp/cron1.log 2>&1
```

- The output is typically recorded on a log in order to analyze it in case of problems.

```
INFO  07-06 06:10:02,486 - Using "/tmp/vfs_cache" as temporary files store.
INFO  07-06 06:10:02,712 - Pan - Start of run.
INFO  07-06 06:10:02,902 - Lettura dati per DWH - Dispatching started for transformation [Lettura dati per DWH]
INFO  07-06 06:10:02,929 - Lettura dati per DWH - This transformation can be replayed with replay date: 2011/06/07 06:10:02
INFO  07-06 06:10:03,233 - DB DWH - Connected to database [Self DB] (commit=100)
INFO  07-06 06:10:03,599 - DB AS_UTIL - Finished reading query, closing connection.
INFO  07-06 06:10:03,614 - DB AS_UTIL - Finished processing (I=27, O=0, R=0, W=27, U=0, E=0)
INFO  07-06 06:10:03,625 - DB DWH - Finished processing (I=0, O=27, R=27, W=27, U=0, E=0)
INFO  07-06 06:10:03,626 - Pan - Finished!
INFO  07-06 06:10:03,627 - Pan - Start=2011/06/07 06:10:02.713, Stop=2011/06/07 06:10:03.126
INFO  07-06 06:10:03,627 - Pan - Processing ended after 0 seconds.
INFO  07-06 06:10:03,627 - Lettura dati per DWH -
INFO  07-06 06:10:03,627 - Lettura dati per DWH - Step DB AS_UTIL.0 ended successfully, processed 27 lines. ( - lines/s)
INFO  07-06 06:10:03,628 - Lettura dati per DWH - Step DB DWH.0 ended successfully, processed 27 lines. ( - lines/s)
```

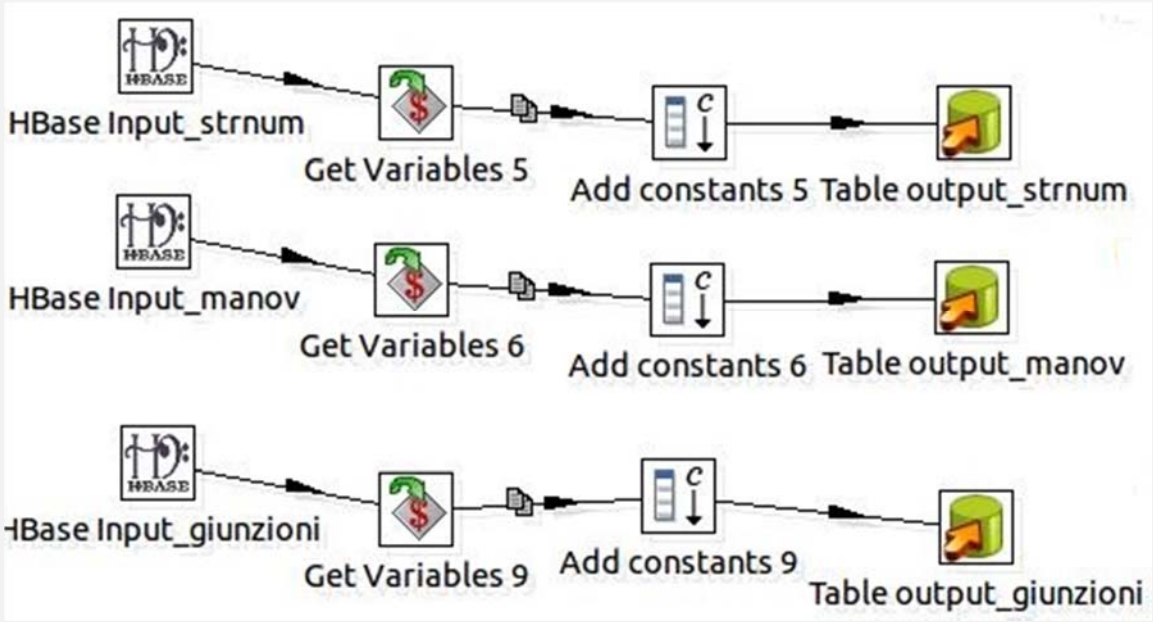
Sequential Execution

These steps (transformations) are executed sequentially (there is a single flow execution).

```
void main(){
        int a;
        f1(a);
        f2(a+2);
}
```

The statements are executed sequentially.

Parallel Execution



- Unlike before there are multiple streams of execution that are executed in parallel (simultaneously).
- Like in a multi-threading programming, multiple thread (portions of the running program) can virtually run independently and in parallel.

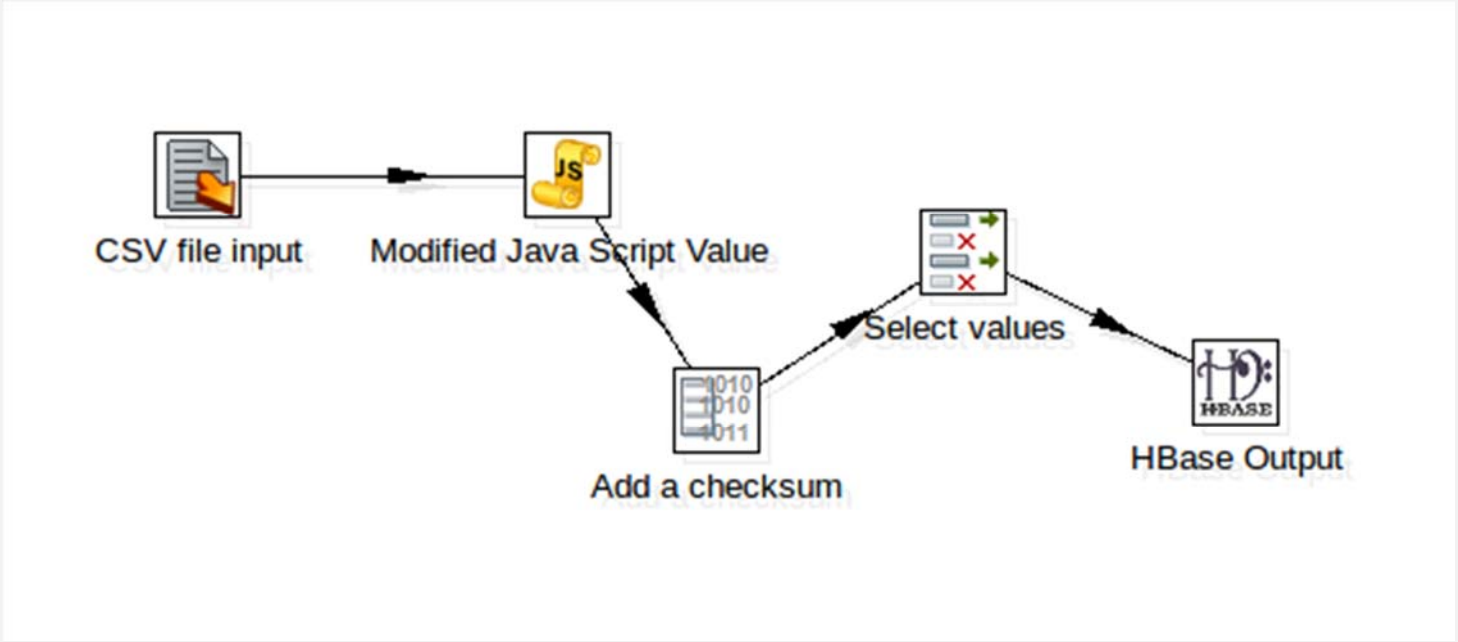
3

ETL tool: Pentaho Data Integration (PDI)

EXAMPLES

Transformation Hbase Ouput

- This transformation takes the museums file in CSV format and defines a key to load data into the HBase table “monuments”.

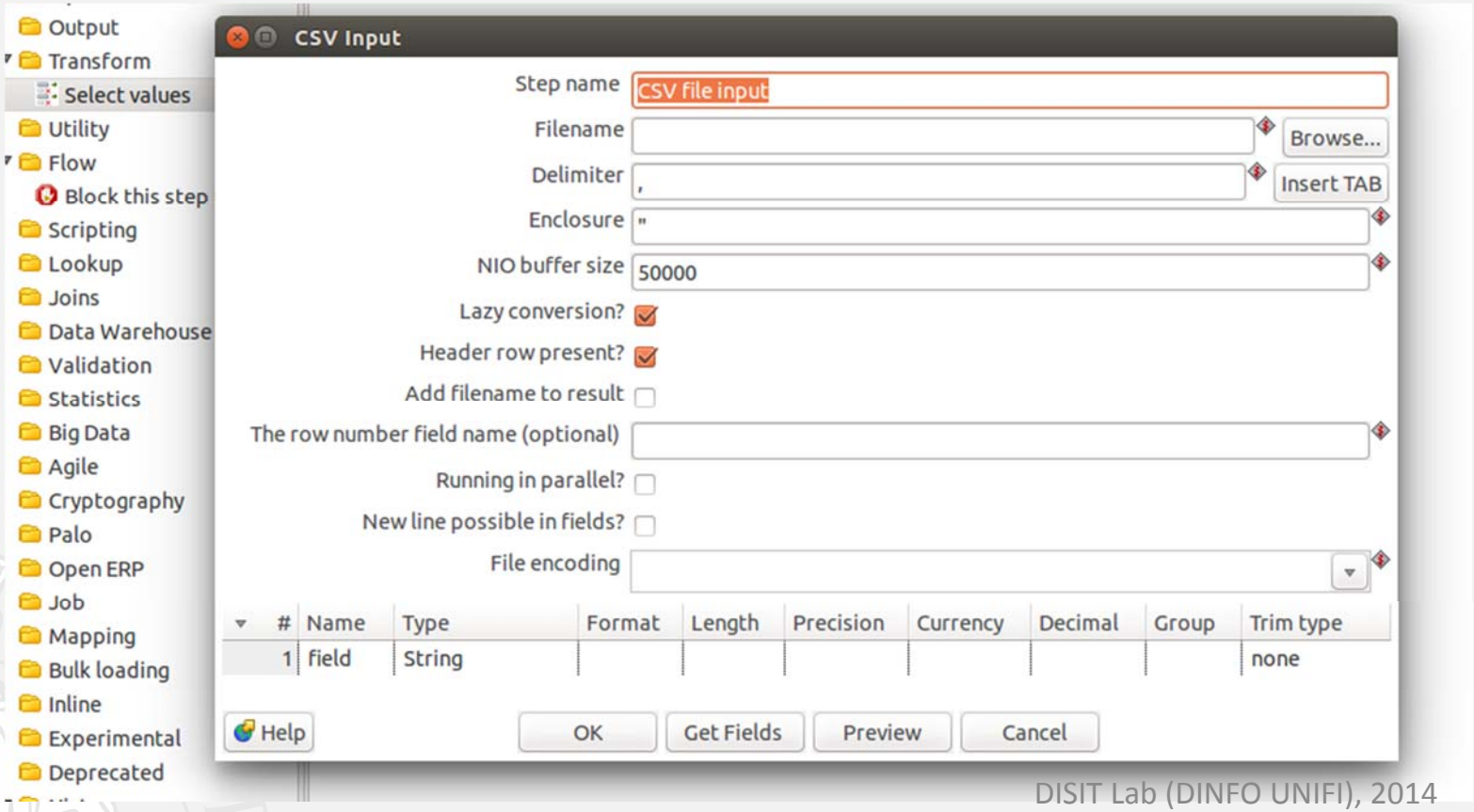


- It is composed of five steps.
- The sequence is given by data flow.

Transformation Hbase Ouput

CSV file input

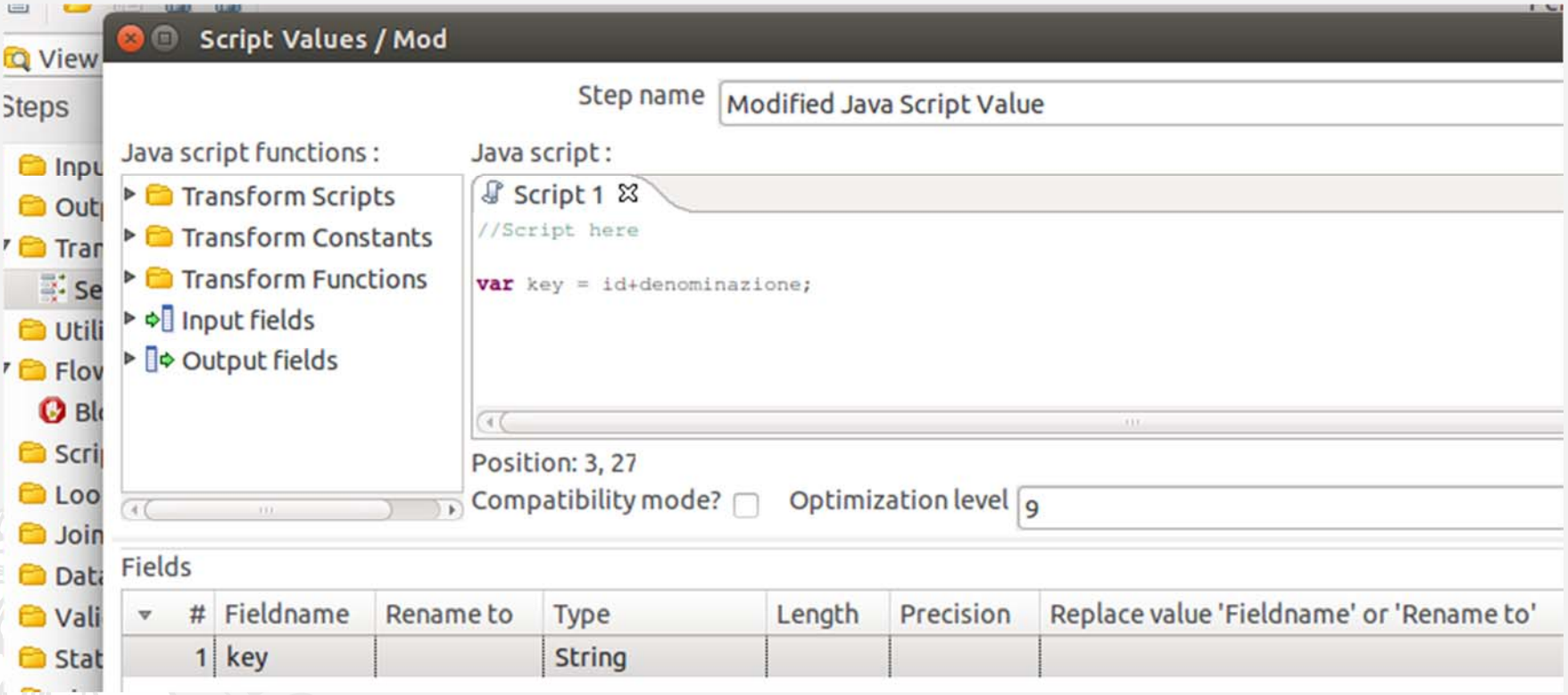
- In this step you select the CSV file, you can choose the separator type used and select the fields to be imported using the Get field button (also you can determine the type and other parameters).



Transformation Hbase Ouput

Modified Java Script value

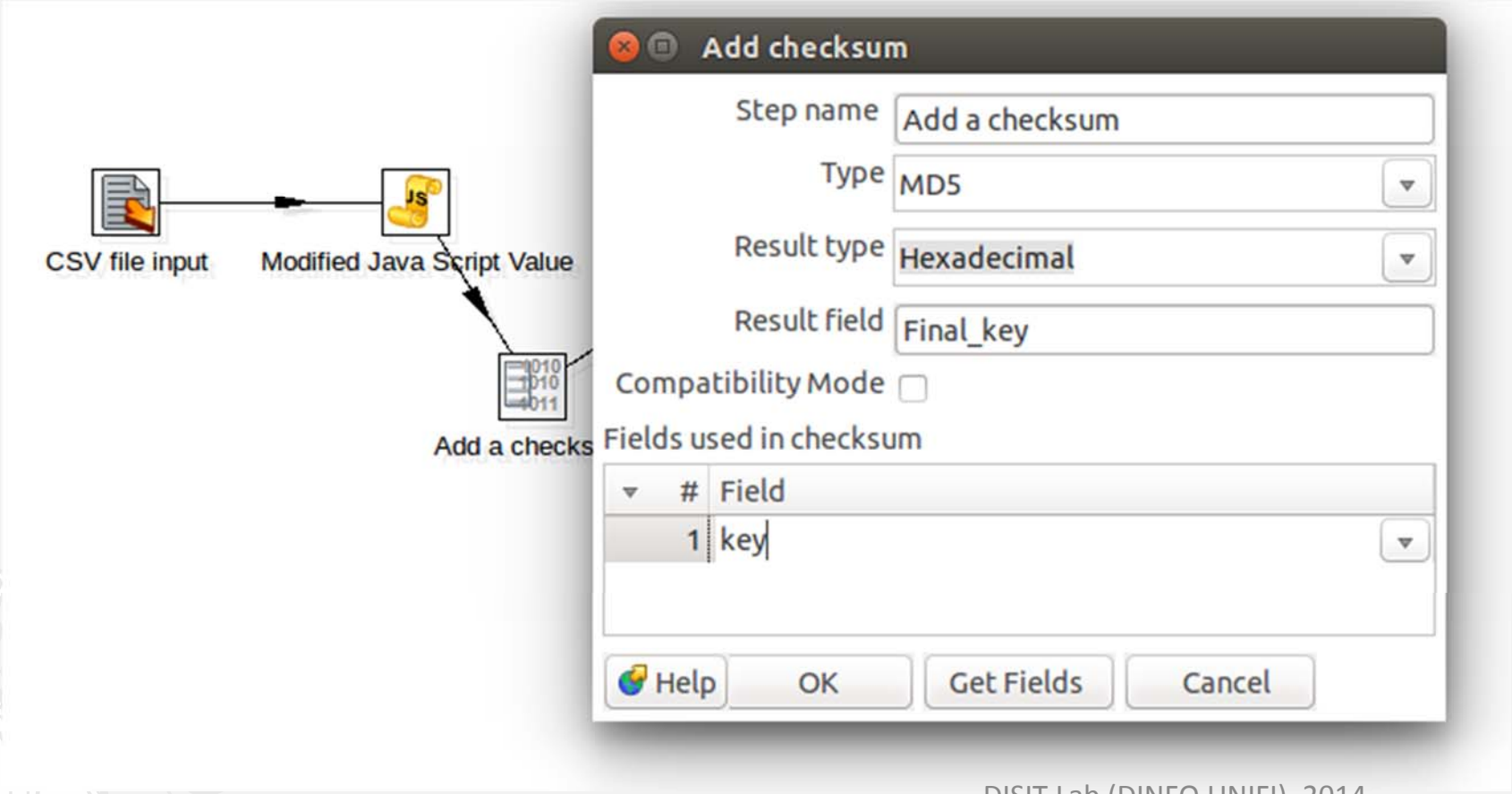
- In this step you can add JavaScript code. It is defined a variable by concatenating two input fields and at the end the same variable is used to define an output field.



Transformation Hbase Ouput

Add a Checksum

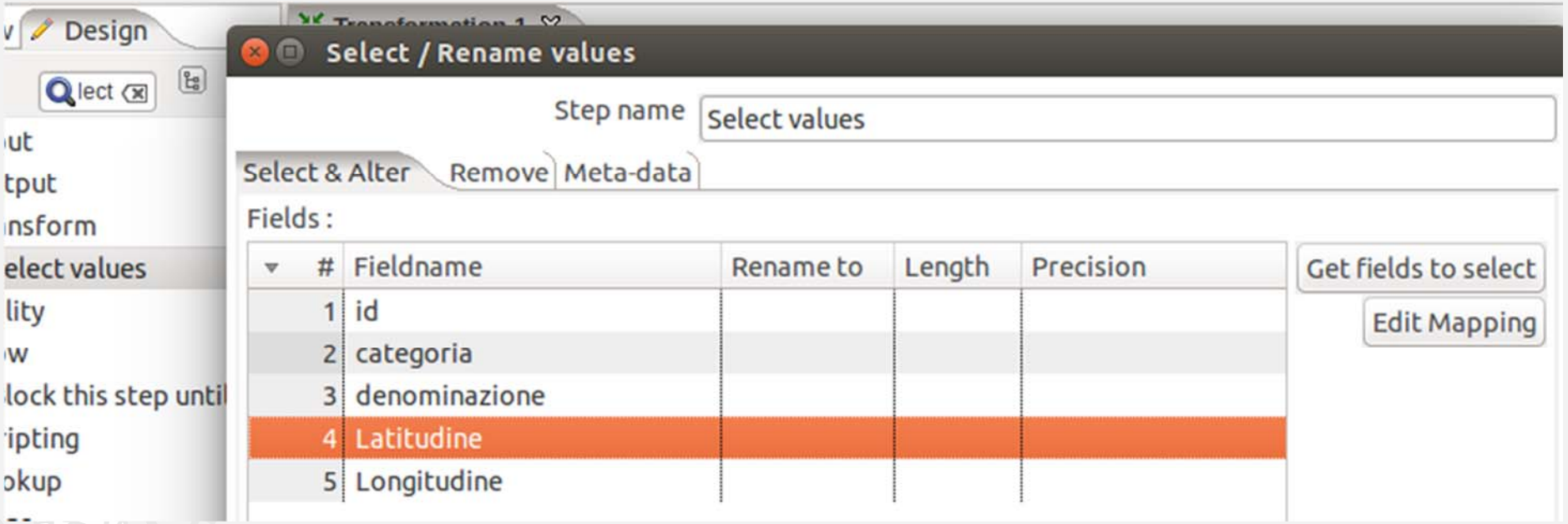
- This step allows you to choose which algorithm (MD5, CRC32) to use to encode a field (usually the key) and define the new name in output.



Transformation Hbase Ouput

Select Values

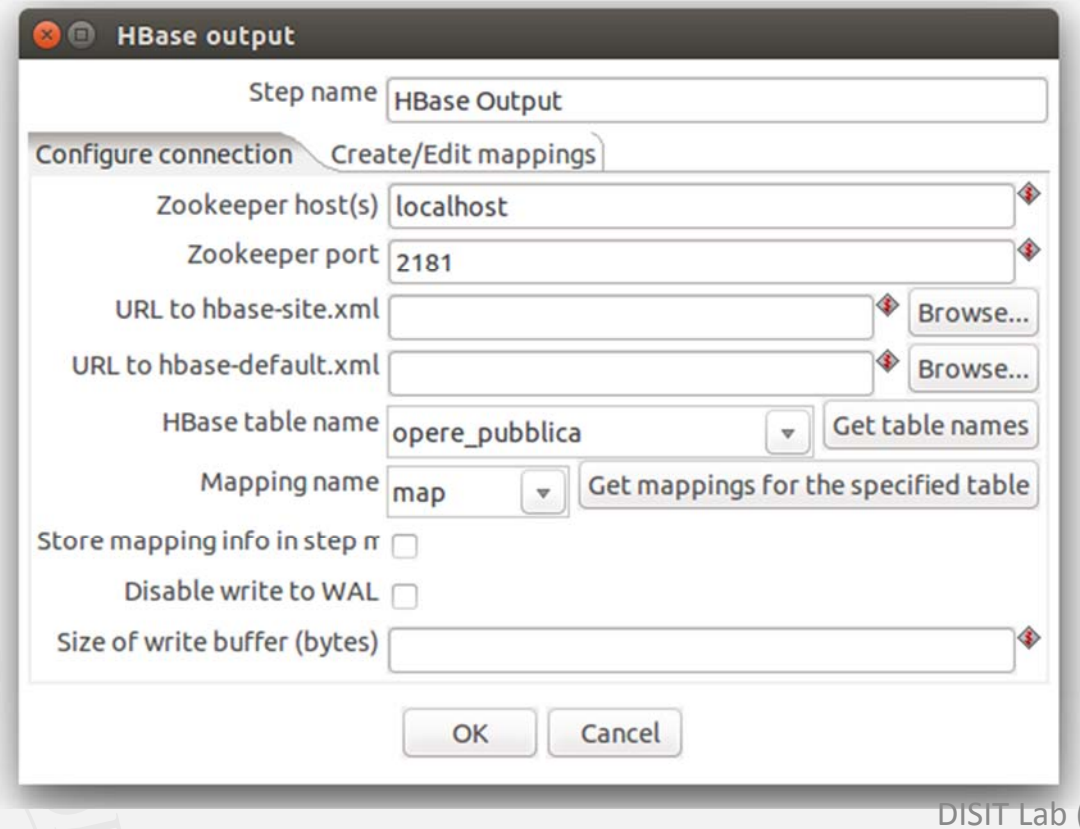
- In this step you can select the fields (one or more) that you want to pass to next step. Furthermore, you can also use the Remove option to select the fields you want to block.



Transformation Hbase Ouput

Hbase Output

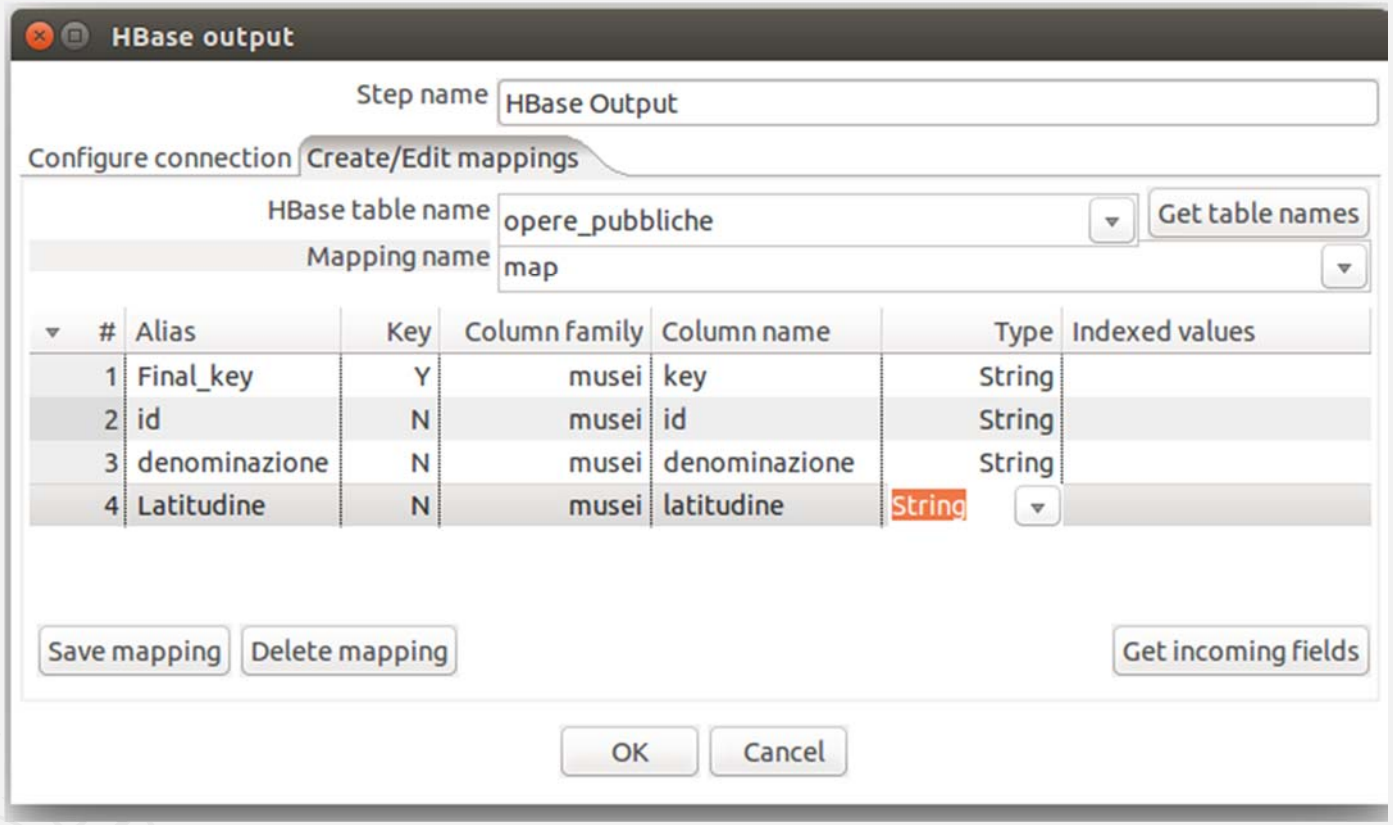
- In this step you set the parameters to load data into a table HBase.
- In the first tab you define the port (2181) and the IP address of the machine that hosts the database.



Transformation Hbase Ouput

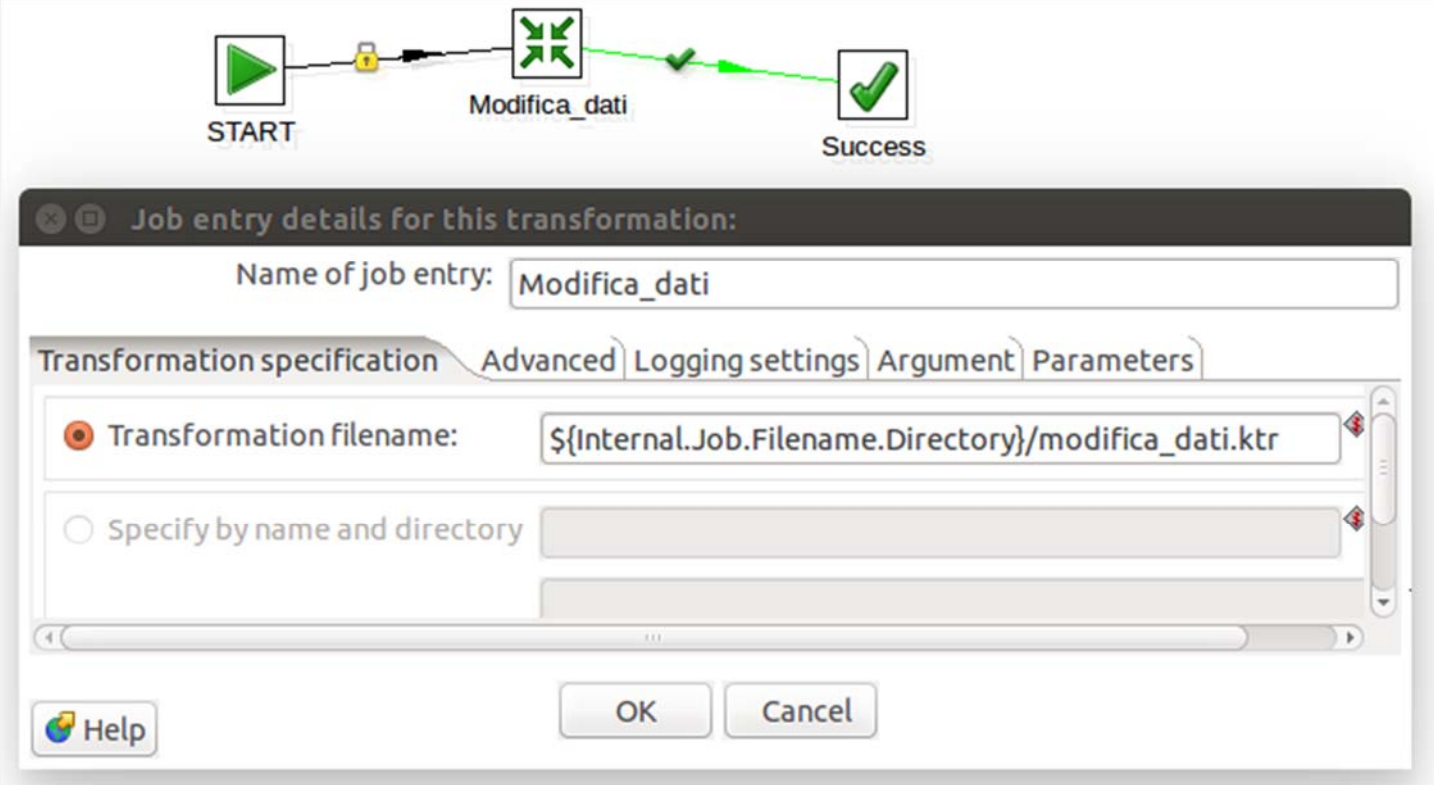
Hbase Output

- In the second tab you select the table, the mapping and the fields to be loaded.



Job

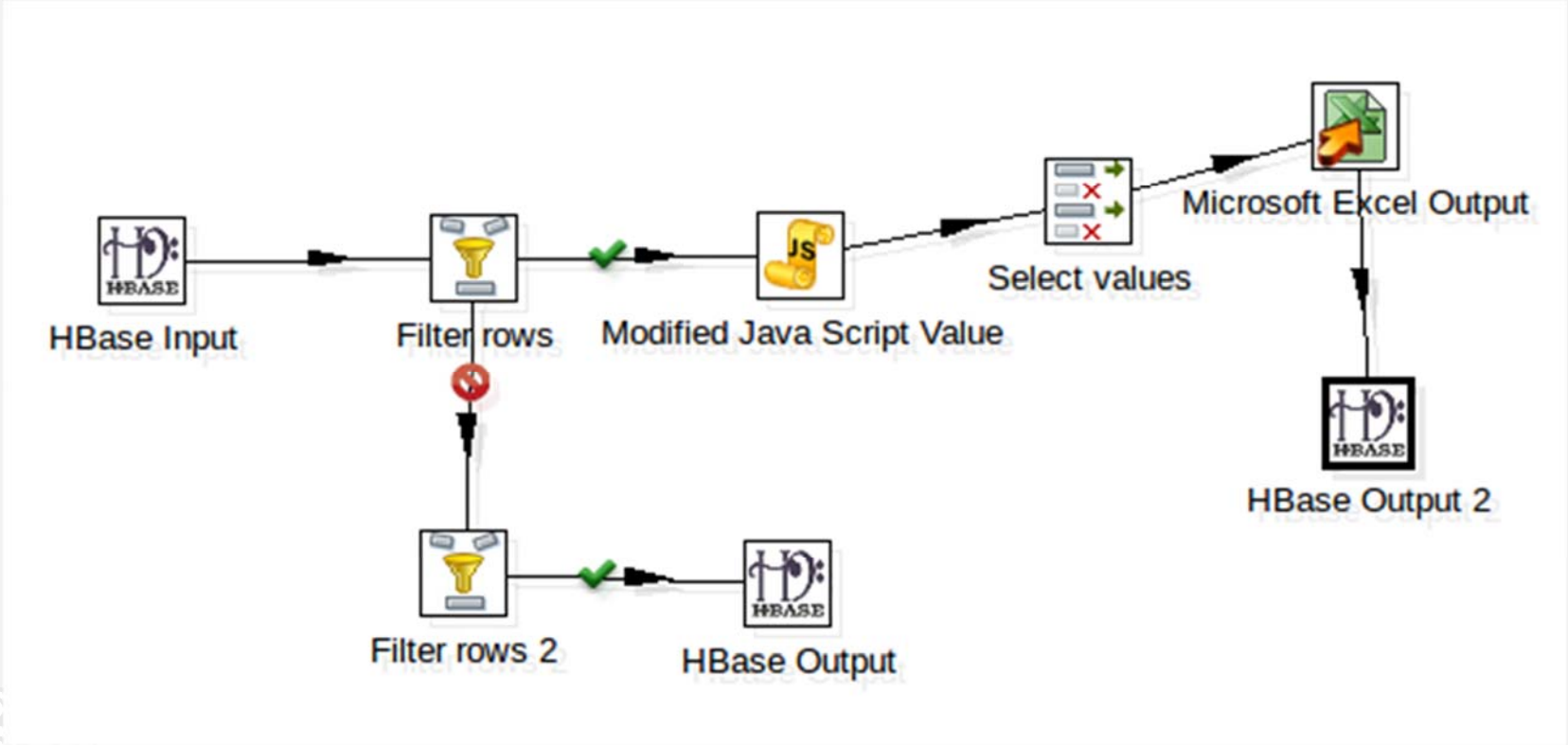
- Create a Job that contains the start step, the "Modifica_dati" step (selected transformation) and the final step (Success).



- The Hop between the different steps of a Job represent the control flow and define the sequence of steps to perform.

Transformation Mod_dati

- This Transform loads data from HBase through the HBase Input Step. Then, it applies the Filter rows and Modified Java Script Value Step to perform some data cleaning before storing again on HBase.

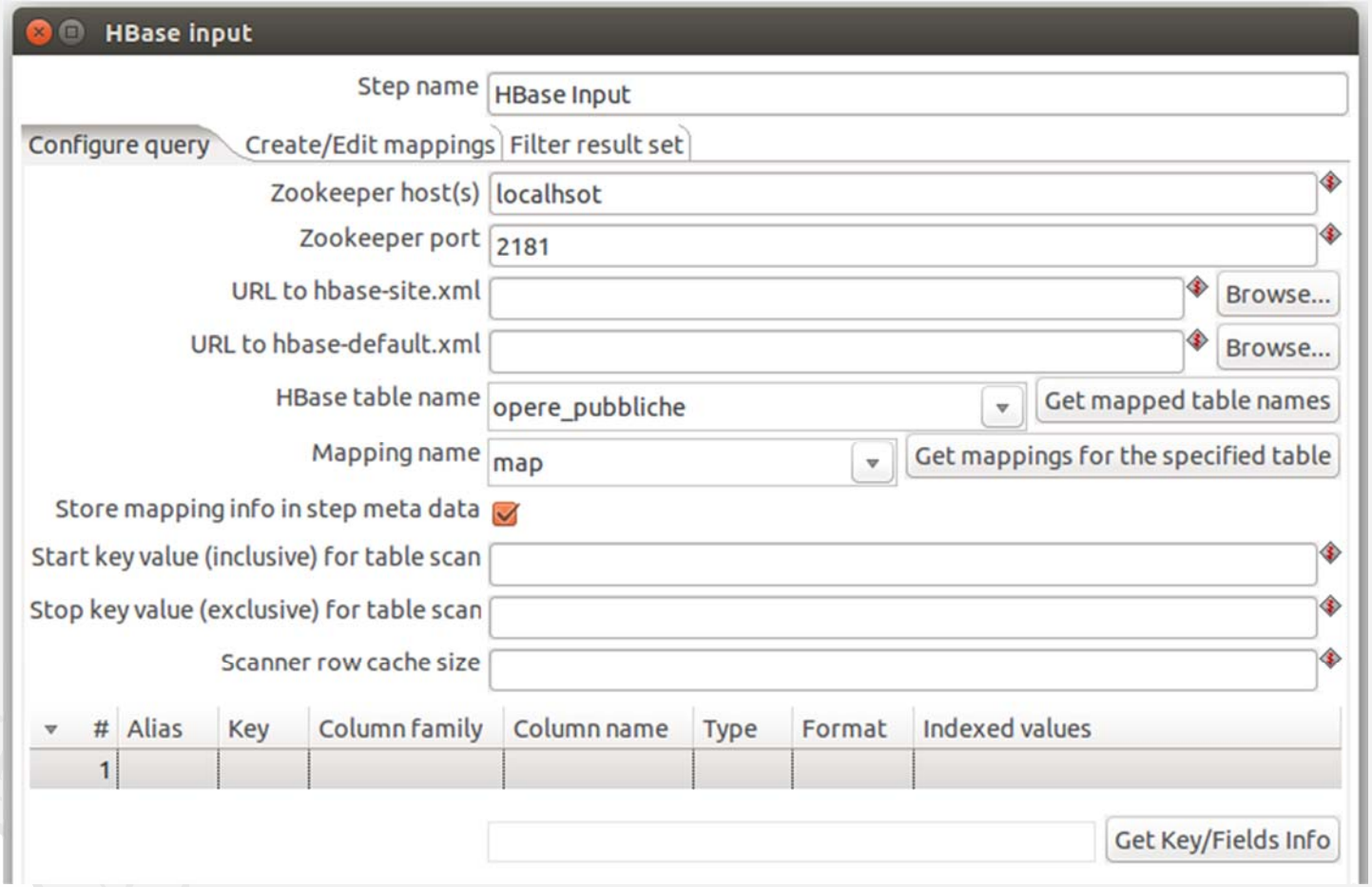


- In this case the data flow is splitted into two different smaller data streams based on specific condition.

Transformation Mod_dati

Hbase Input

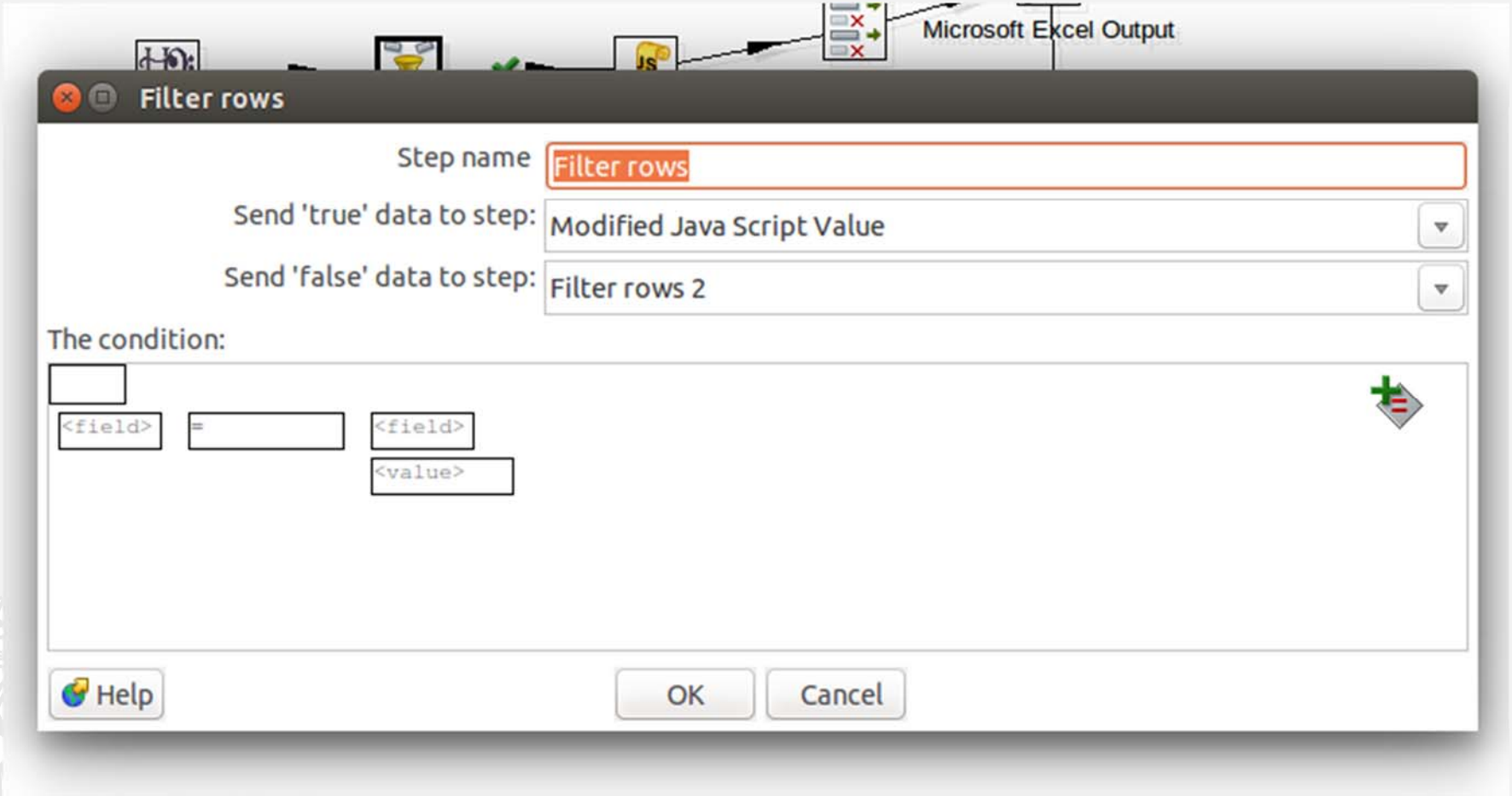
- In this step you set the parameters to retrieve the data from Hbase.
- You can perform a filtering by setting some conditions in the tab Filter result set.



Transformation Mod_dati

Filter rows

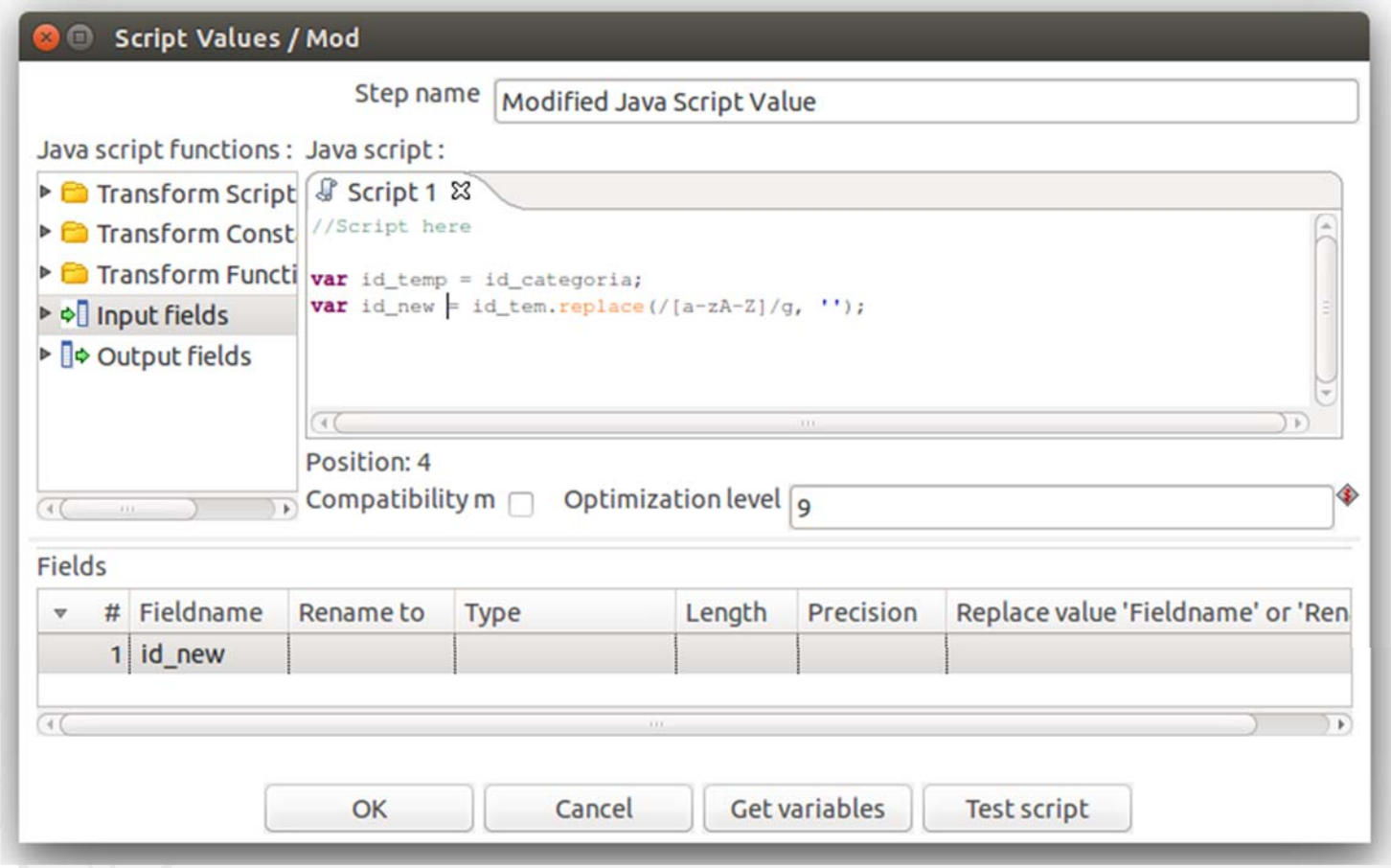
- It's another way to filter the data flow specifying a condition. In output this step creates a fork of the data flow.



Transformation Mod_dati

Modified Java Script Value

- In this step you will use a regular expression inside the Javascript code. The goal is to replace all literals characters within a given field, leaving only numeric ones.

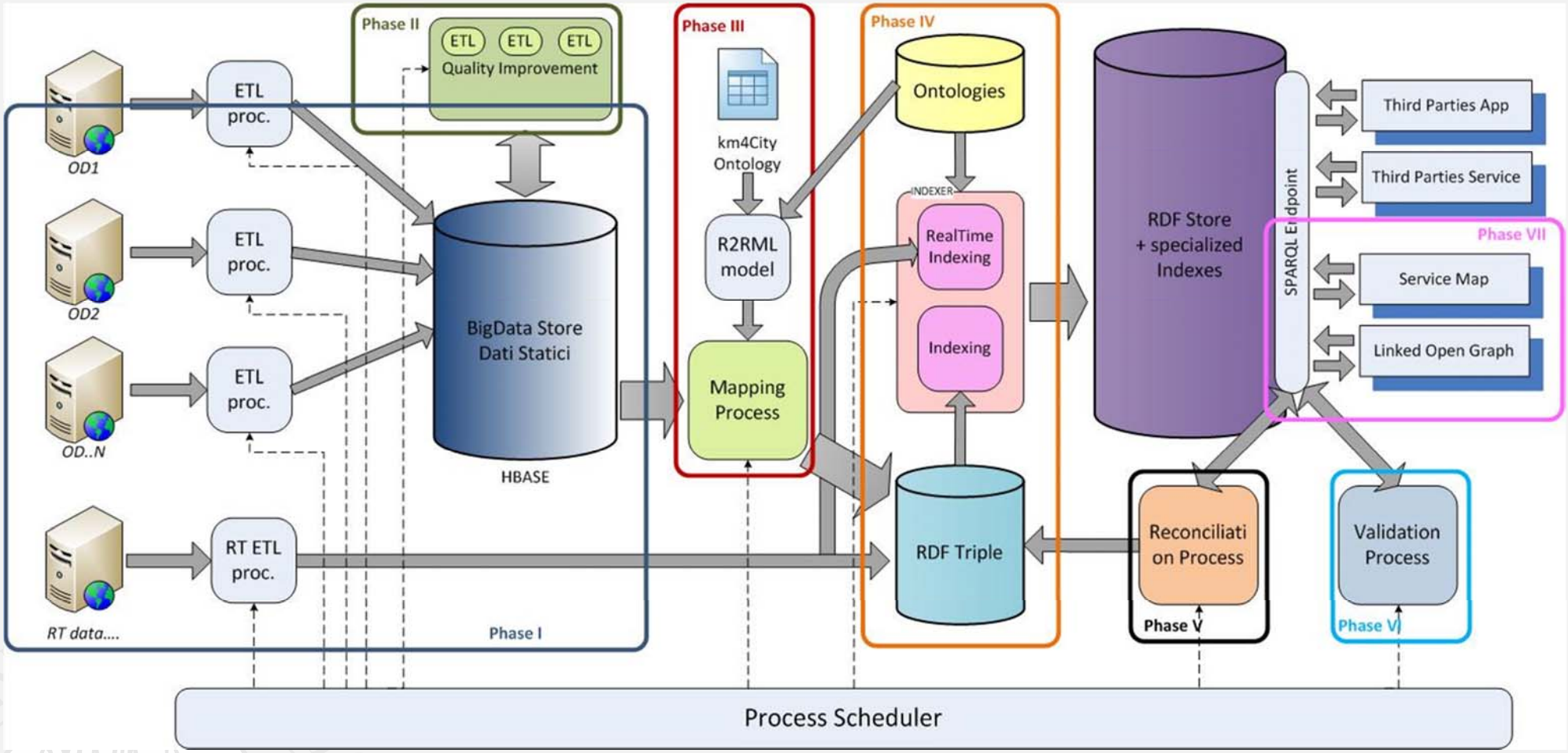


4

Sii-Mobility Project

Sii-Mobility project (a part)

http://www.sii-mobility.org



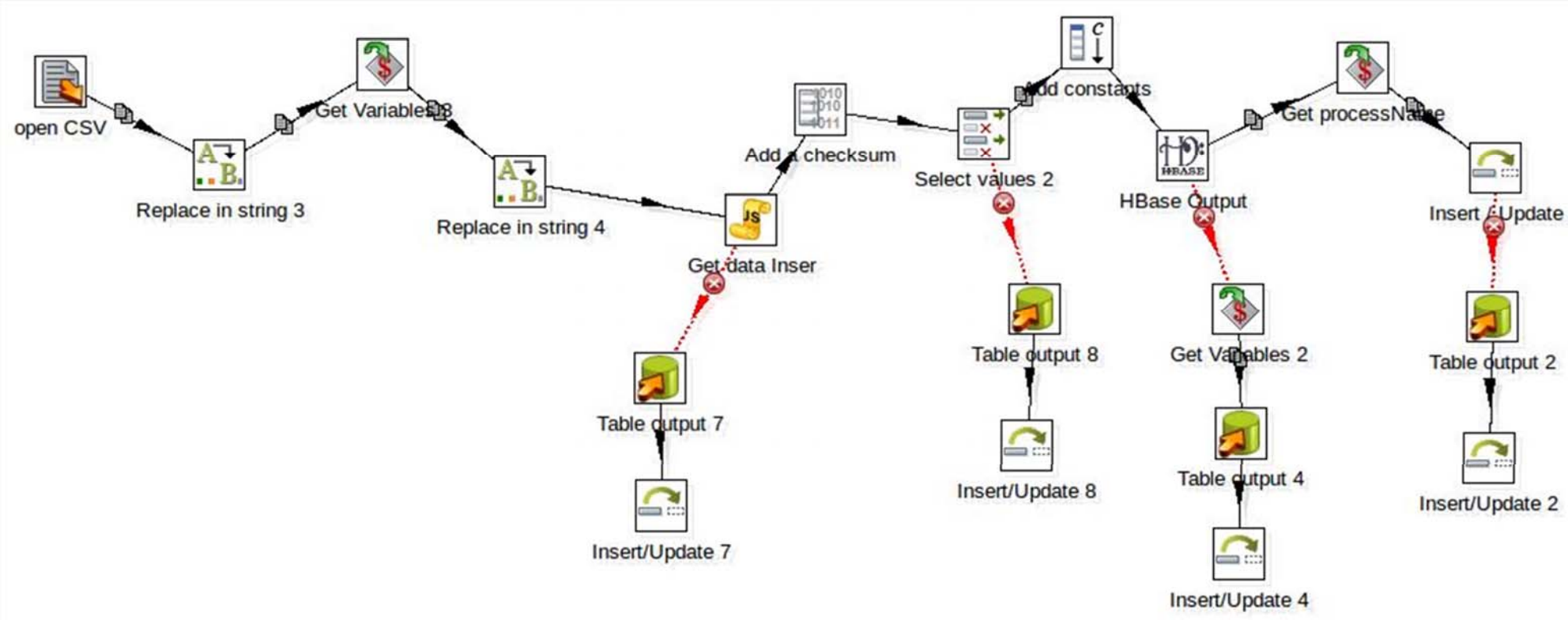
Transformation Service Data

- To **process the service data** for Sii-Mobility project.
  - **static data** from Tuscan region.

- **3** phases:
  - **INGESTION** phase;
  - **QUALITY IMPROVEMENT (QI)** phase;
  - **TRIPLES GENERATION** phase.

Ingestion phase

To importing and storage data (in a database) for later use.

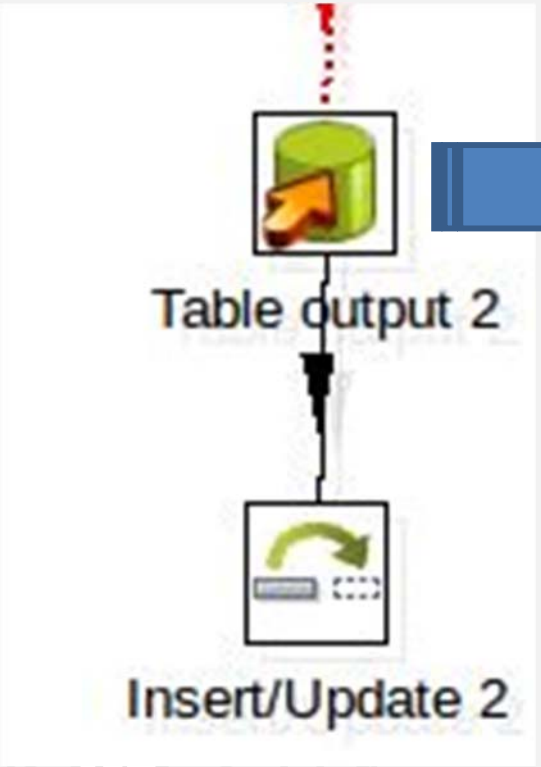


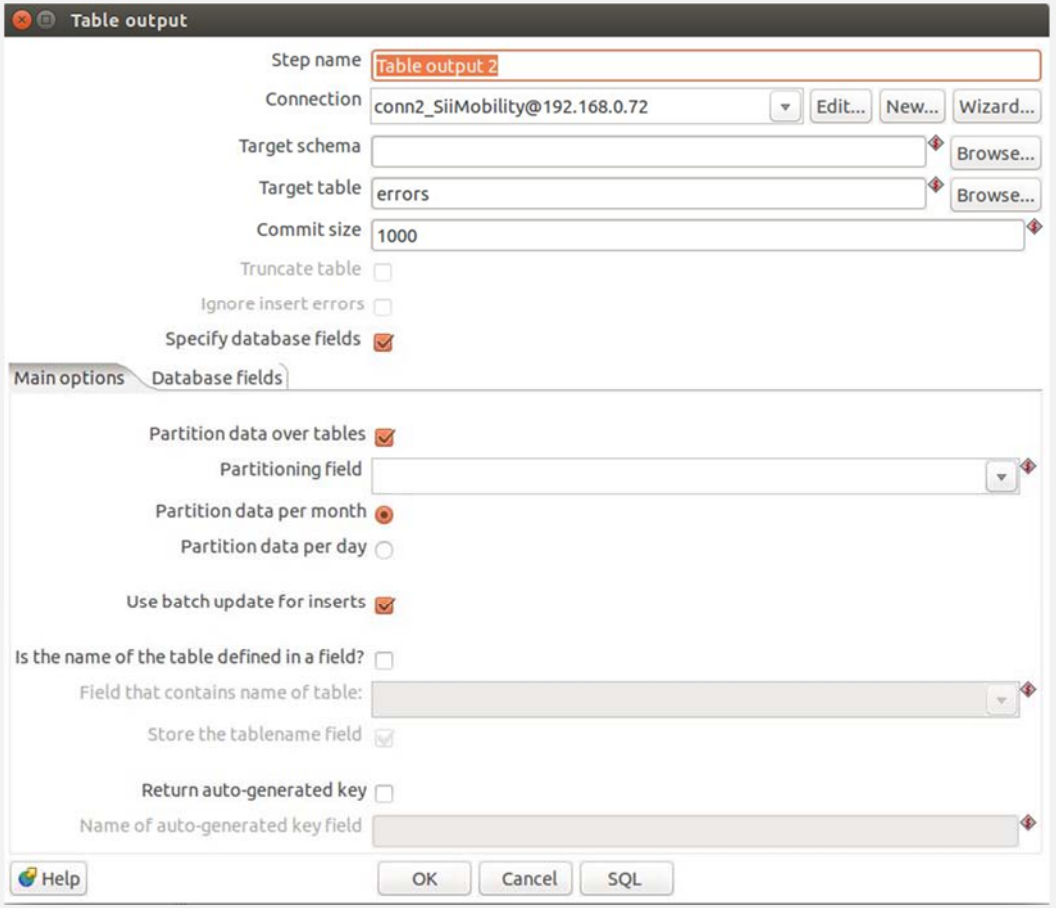
The sequence is given by data flow.

Ingestion phase

Table output

- In this step you load data into a database table (for example a MySQL table).

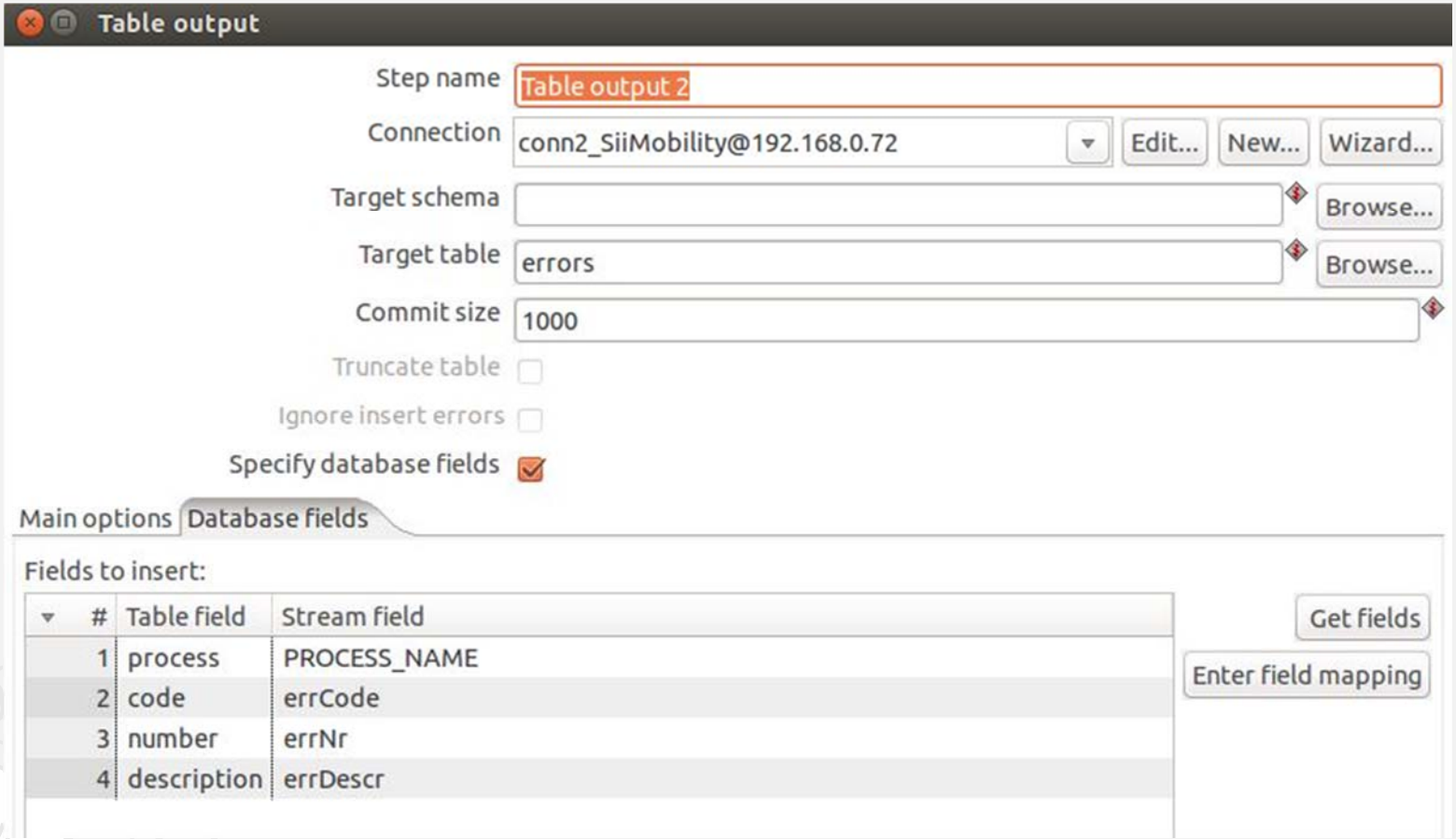




Ingestion phase

Table output

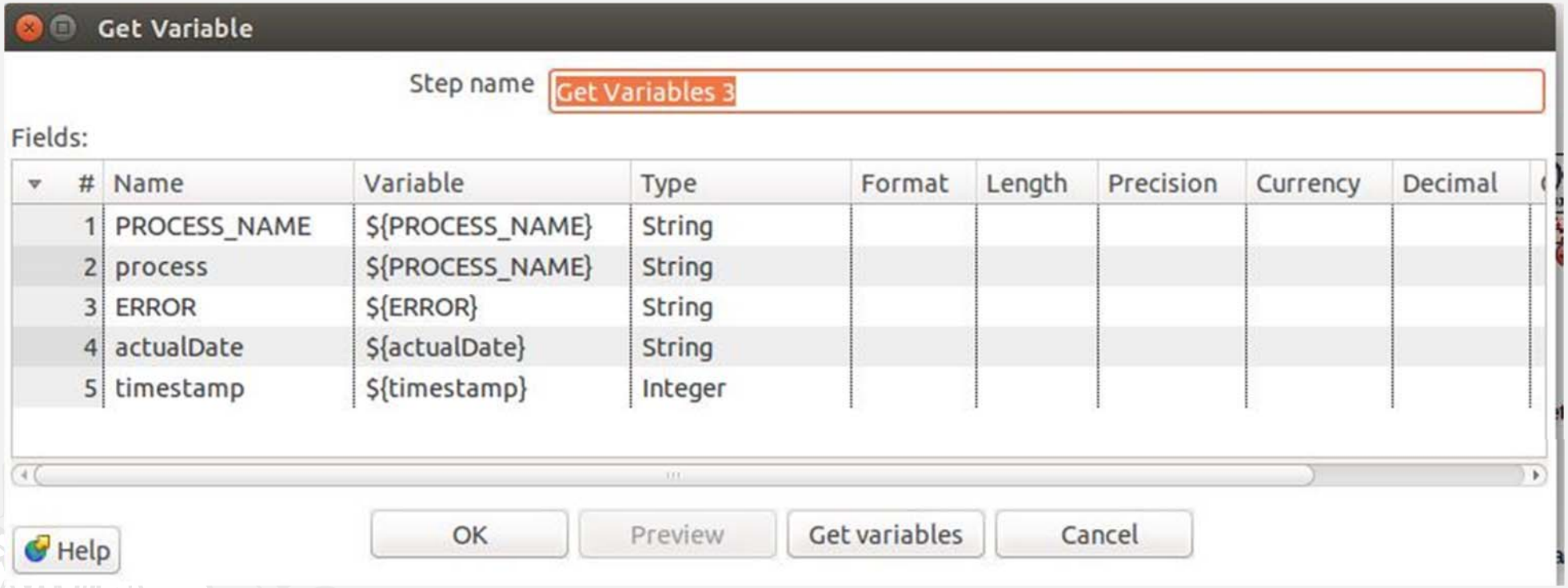
- There are several options to set this step: database connection, table name, specify the fields in the Database fields tab, etc….



Ingestion phase

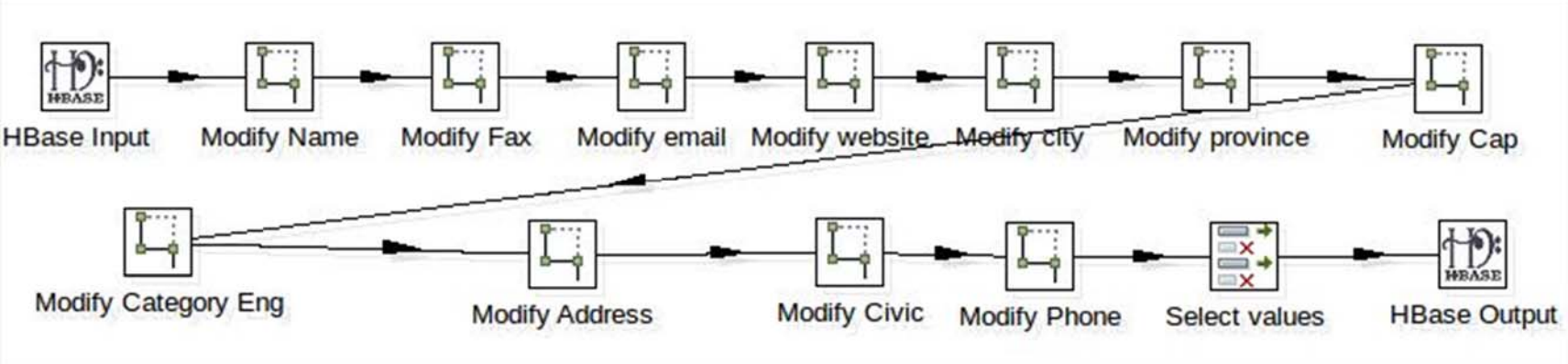
Get Variable

- This step allows you to get the value of a variable.
- You must specify the complete variable specification in the format ${variable}.



QI phase

To enhance the quality of raw data and to produce reliable and useful information for next applications.



In this case every Step is a transformation.

What is the **Data quality** ?

QI phase

Data quality's aspect:

- **Completeness**: presence of all information needed to describe an object, entity or event (e.g. Identifying).
- **Consistency**: data must not be contradictory. For example, the total balance and movements.
- **Accuracy**: data must be correct, i.e. conform to actual values. For example, an email address must not only be well-formed nome@dominio.it, but it must also be valid and working.

QI phase

- **Absence of duplication:** tables, records, fields should be stored only once, avoiding the presence of copies. Duplicate information involve double handling and can lead to problems of synchronization (consistency).
- **Integrity** is a concept related to relational databases, where there are tools to implement integrity constraints. Example a control on the types of data (contained in a column), or on combinations of identifiers (to prevent the presence of two equal rows).

QI phase

Mapping

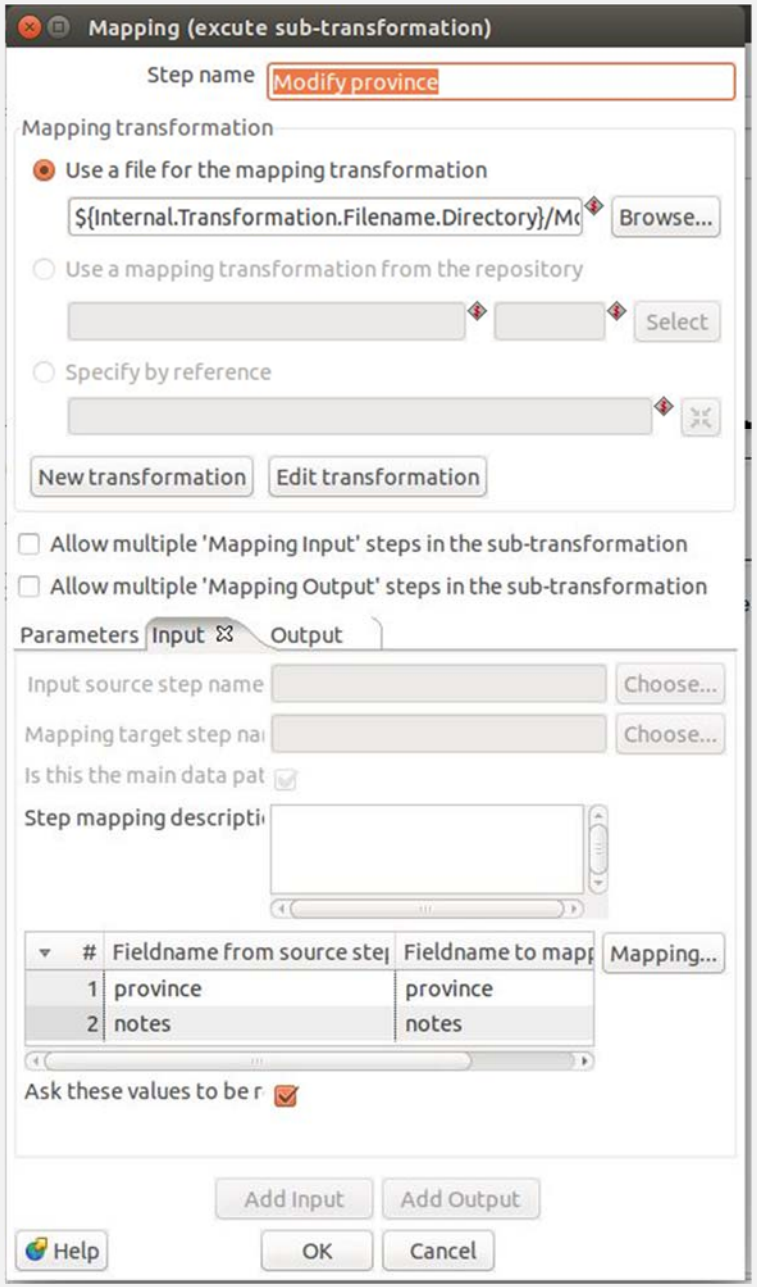
- A mapping is the Kettle solution for transformation re-use.
- For example, if you have a complex calculation that you want to re-use everywhere you can use a mapping.

- These interface steps define the fields structure of the incoming and returning rows. So when a parent transformation calls a sub-transformation the parent row fields are mapped to the fields that the sub-transformation accepts as input. A similar mapping happens when the processed rows are returned to the parent.

QI phase

Mapping

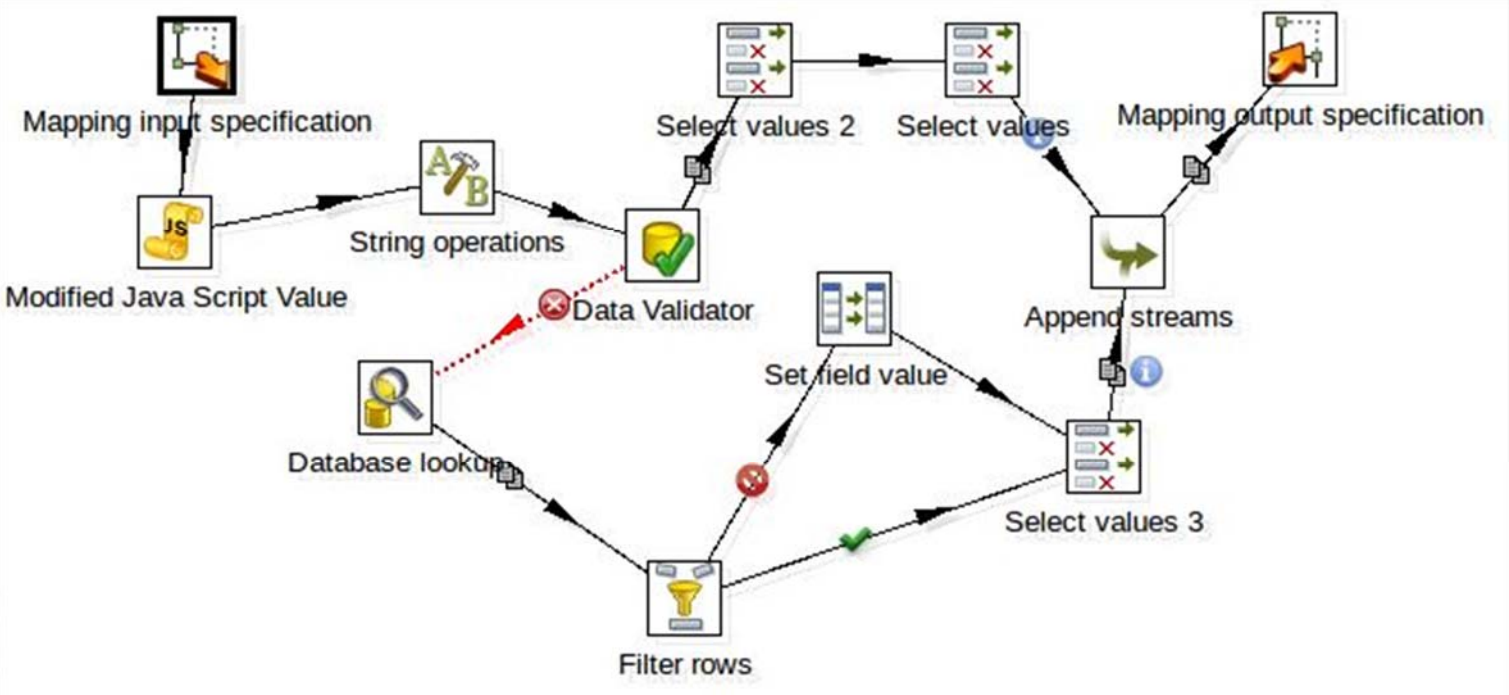
- To re-use a transformation you:
  - specify the sub-transformation to execute;
  - can define or pass Kettle variables down to the mapping;
  - can specify the input fields that are required by your sub-transformation;
  - can specify the ouput fields that are required by your sub-transformation.
- You can see this how a function that returns output values calculated on a specific input data.



QI phase

Mapping

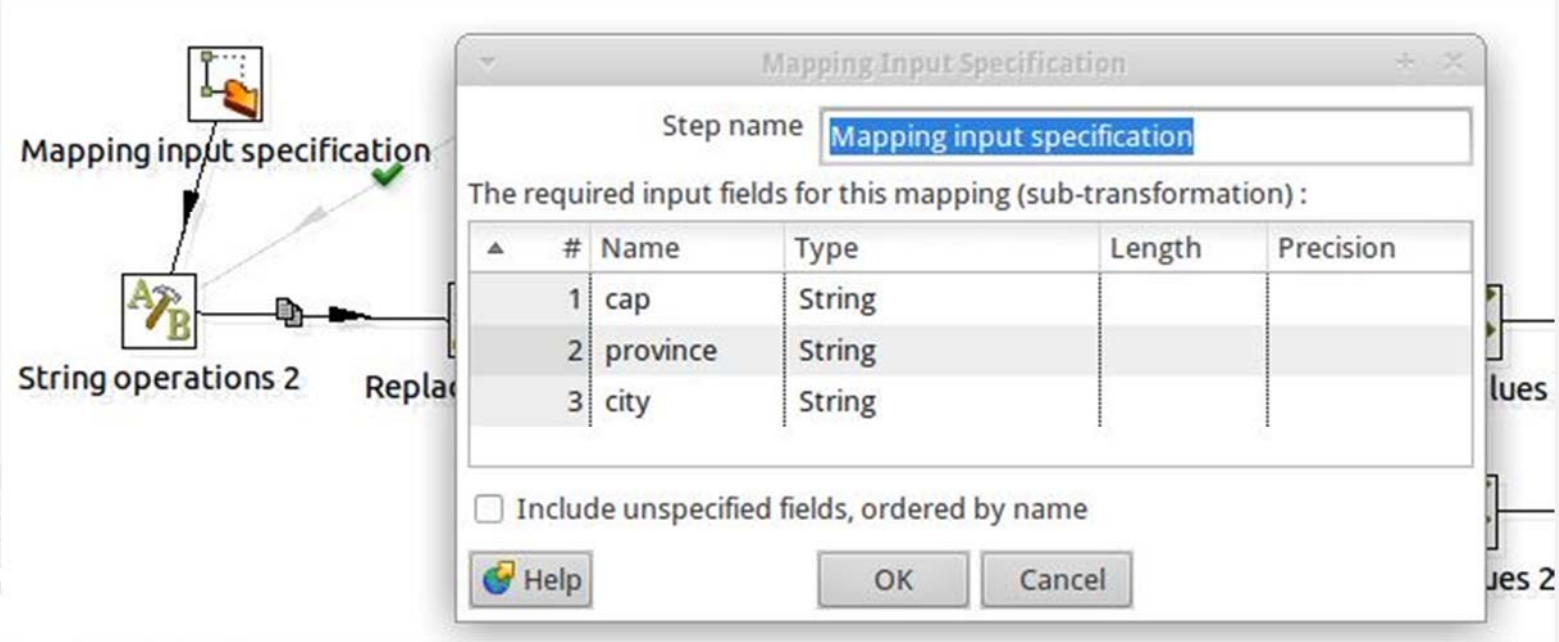
- A mapping is also called a sub-transformation because it is a transformation just like any other with a couple of key differences.



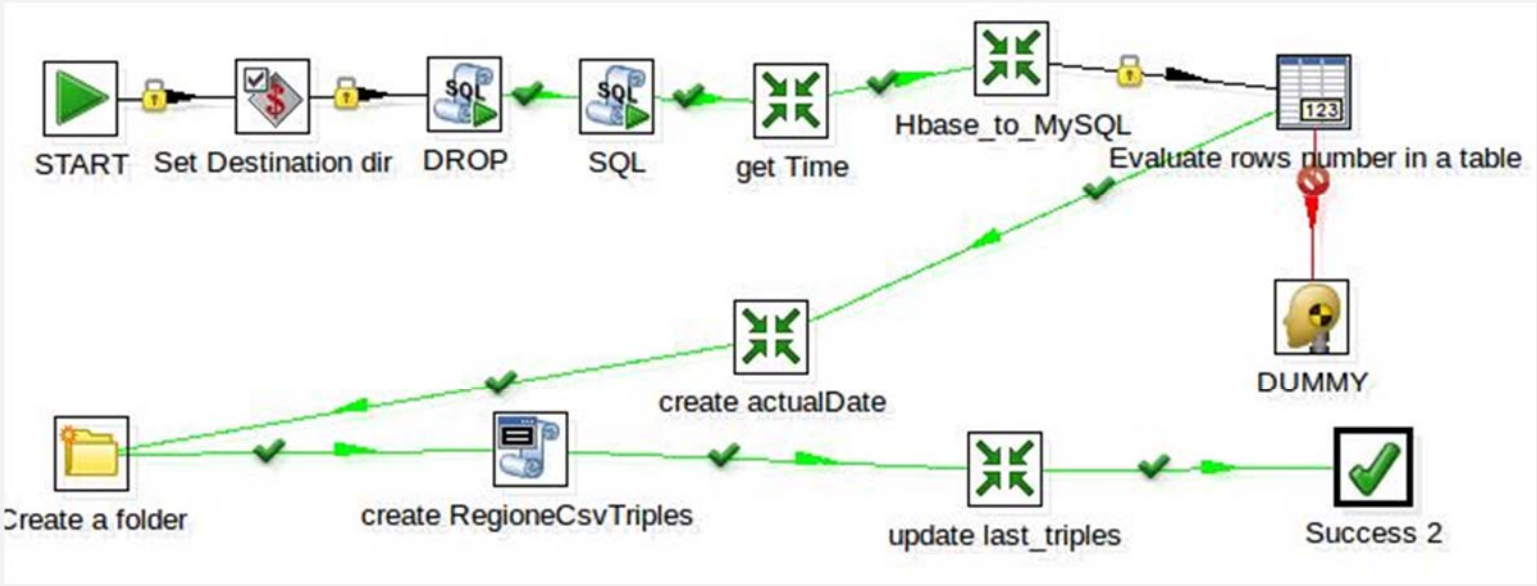
QI phase

Every mapping needs

- a Mapping Input step to define the fields that are **required** for the correct mapping execution.
- a Mapping Output step to define the fields that are **generated** by the mapping.



RDF Triples generation phase

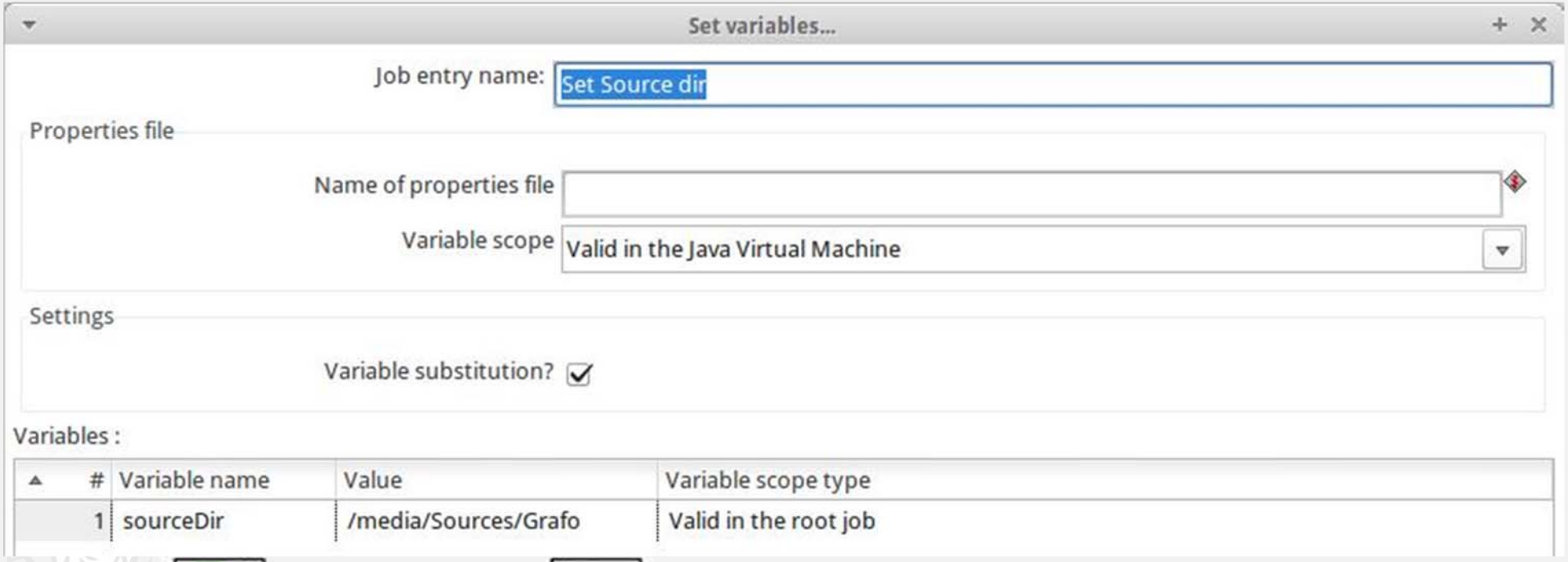


The triples are generated with the **km4city** ontology and then loaded on OWLIM RDF store.

RDF Triples generation phase

Set variables

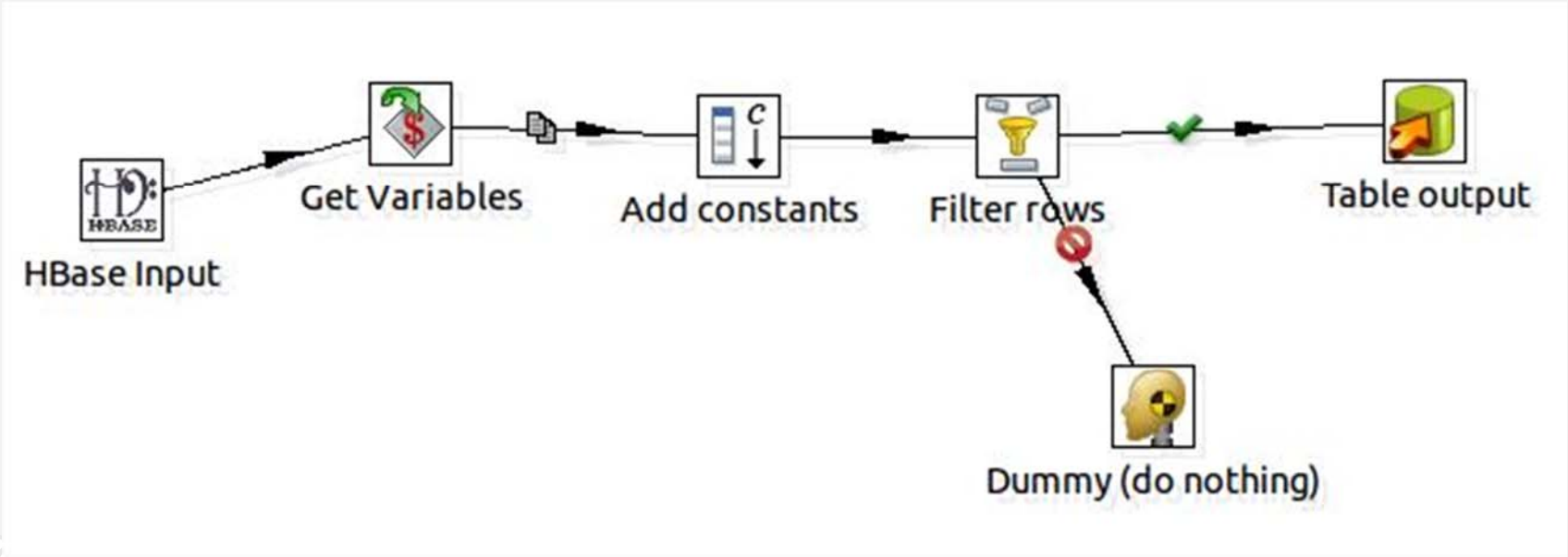
- This step copies in an environment variable the name of subfolder in which operations are performed. You can then reference it by using the command ${sourceDir}.



Get rows from result Set Variables

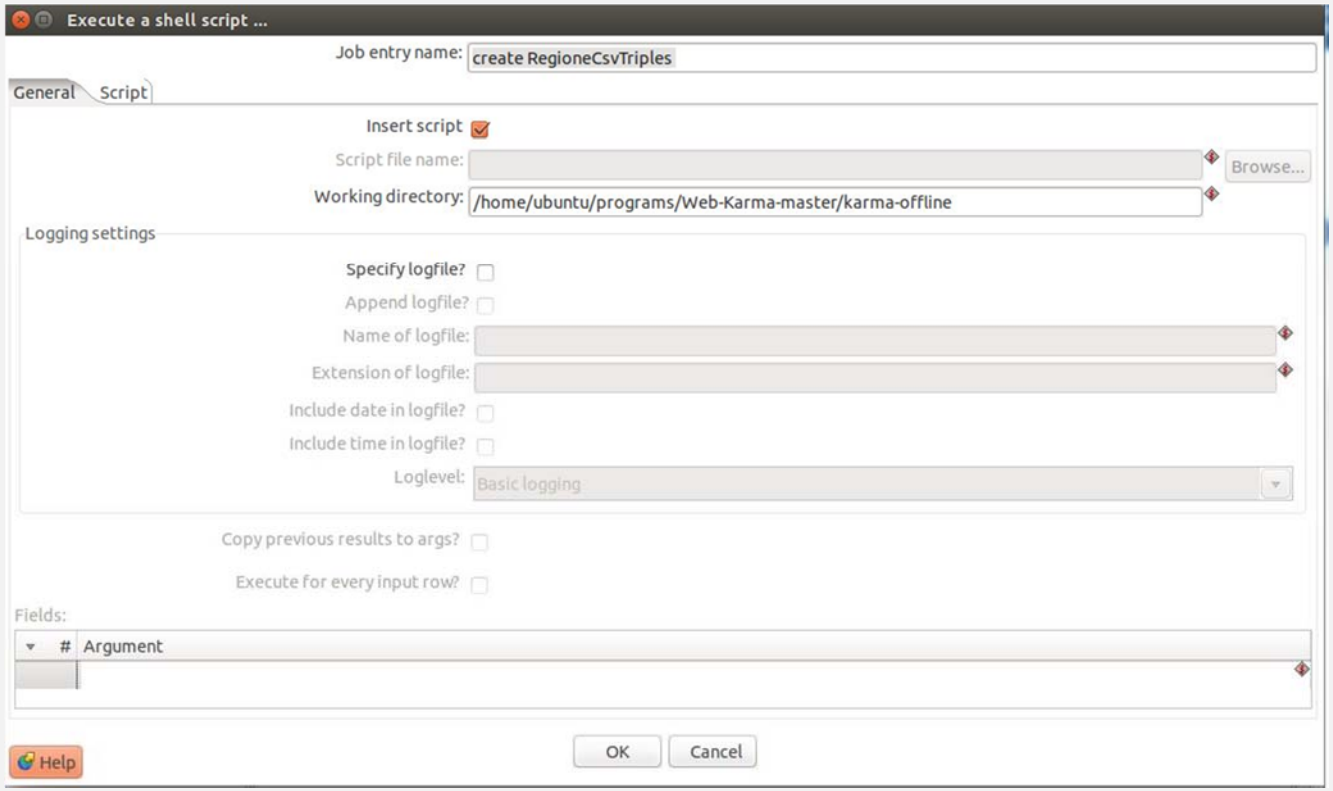
RDF Triples generation phase

1. Load data from Hbase in order to copy them into a MySQL table.



RDF Triples generation phase

2. Create triple RDF from data loaded through a specific script.

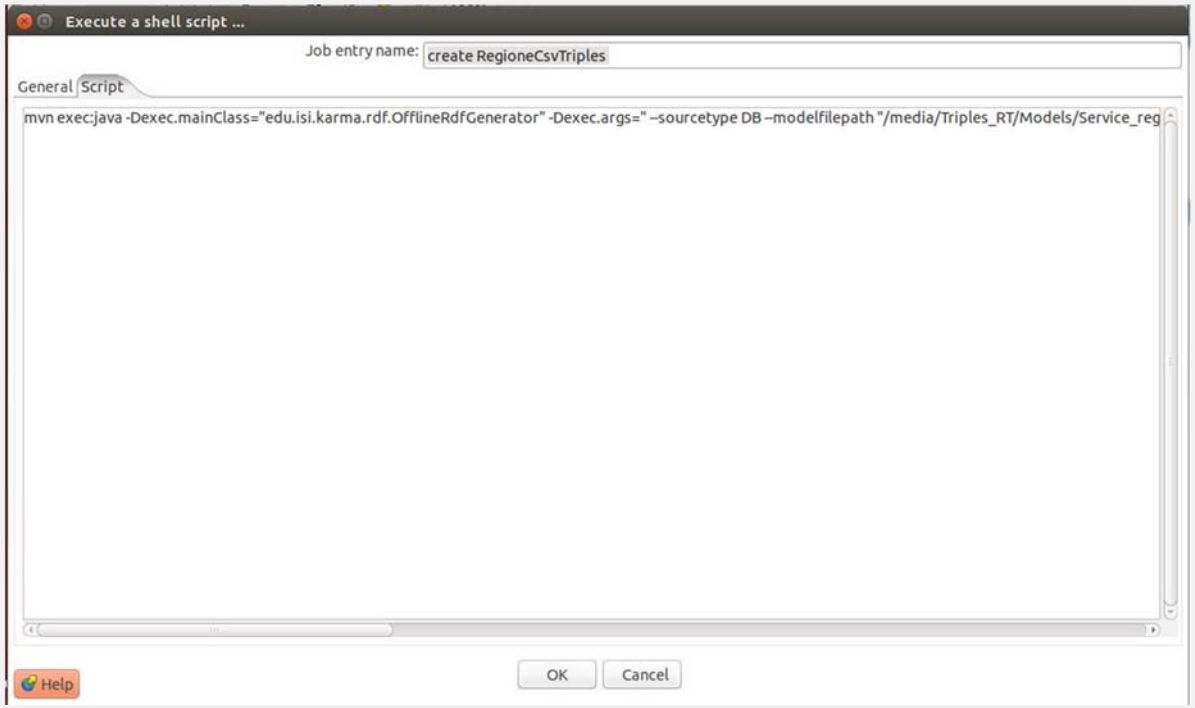


- You can use the **Execute a shell script Step.**
- You can check the option “Insert Script” if you want to execute the script in the Script tab instead of executing the Script file name.

RDF Triples generation phase

Execute a shell script Step

- Insert the specific command to execute the script.




```
mvn exec:java -Dexec.mainClass="edu.isi.karma.rdf.OfflineRdfGenerator" -Dexec.args=" --sourcetype DB –
modelfilepath "/media/Triples_RT/Models/Service_region.ttl" –outputfile
${DestinationDir}/${processName}.n3 --dbtype MySQL --hostname 192.168.0.01 --username x --password x --
portnumber 3306 --dbname Mob --tablename ${processName}" -Dexec.classpathScope=compile
```

RDF Triples generation phase

Execute a shell script Step

```
mvn exec:java -Dexec.mainClass="edu.isi.karma.rdf.OfflineRdfGenerator" -
Dexec.args=" --sourcetype DB –modelfilepath
"/media/Triples_RT/Models/Service_region.ttl" –outputfile
${DestinationDir}/${processName}.n3 --dbtype MySQL --hostname
192.168.0.01 --username x --password x --portnumber 3306 --dbname Mob --
tablename ${processName}" -Dexec.classpathScope=compile
```

- In input you specify the mapping model, the database table (where you get source data) and the connection parameters to database.
- In output you specify the file name (.n3) where the triples RDF will be stored.

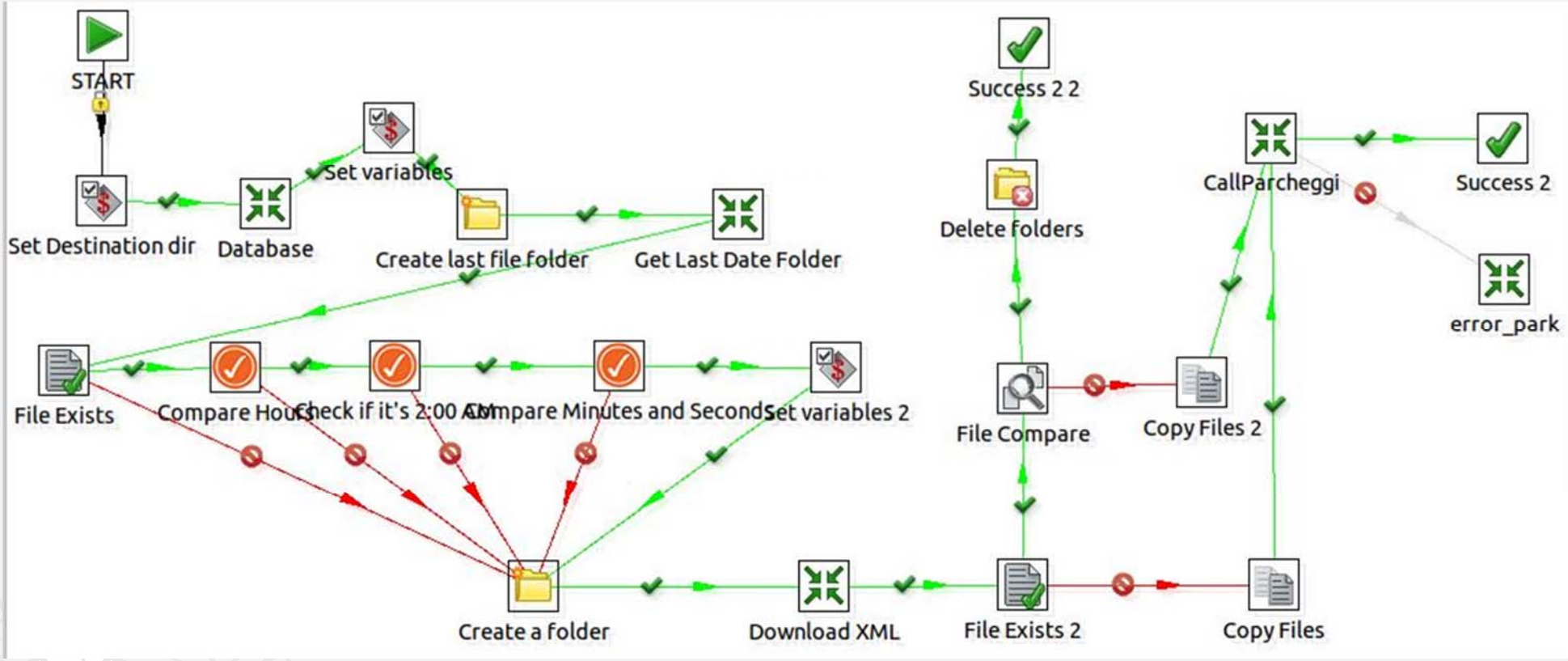
Transformation Parking

- To **process the parking data** for Sii-Mobility
  - **real time data** from Osservatorio Trasporti of Tuscany region (MIIC).

- **2** phases:
  - **INGESTION** phase;
  - **TRIPLES GENERATION** phase.

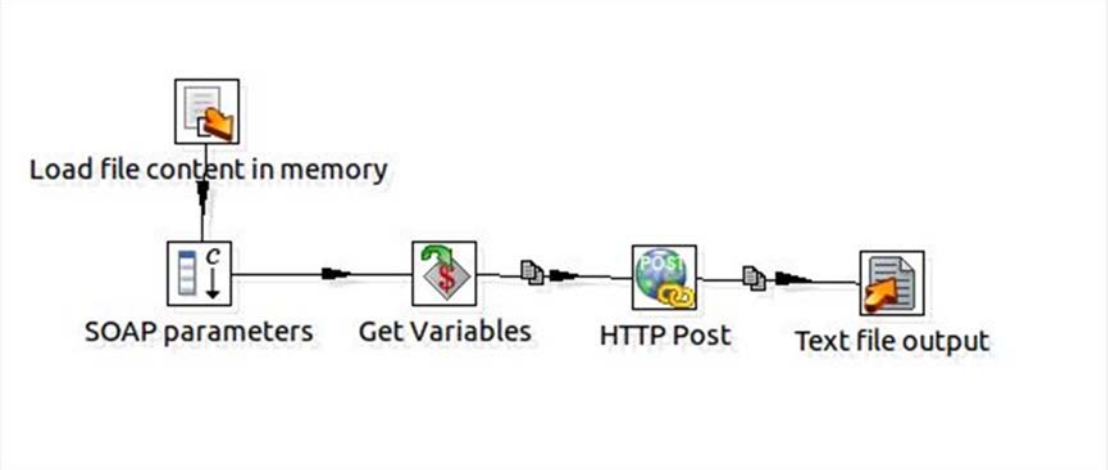
Ingestion phase

To importing and storage data (in a database) for later use.



Ingestion phase

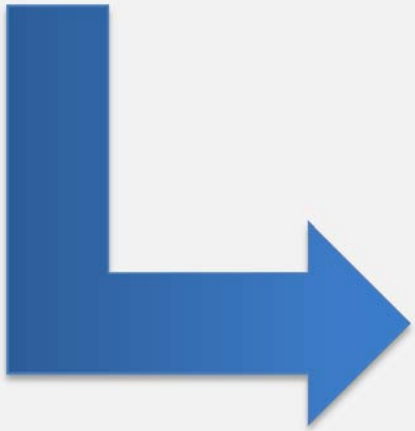
1. Taken an XML file (request.xml) that will be used to invoke the web service (forms the HTTP Post body)



2. Creation of static fields that are passed to HTTP post and HTTP headers such as SOAP action, content-type, username and password.

Ingestion phase

3. Adding the parameter catalog to identify a sensors group.
4. Invocation of web service with **HTTP Post step.**
5. Storing data on Hbase (CallParcheggi transformation).

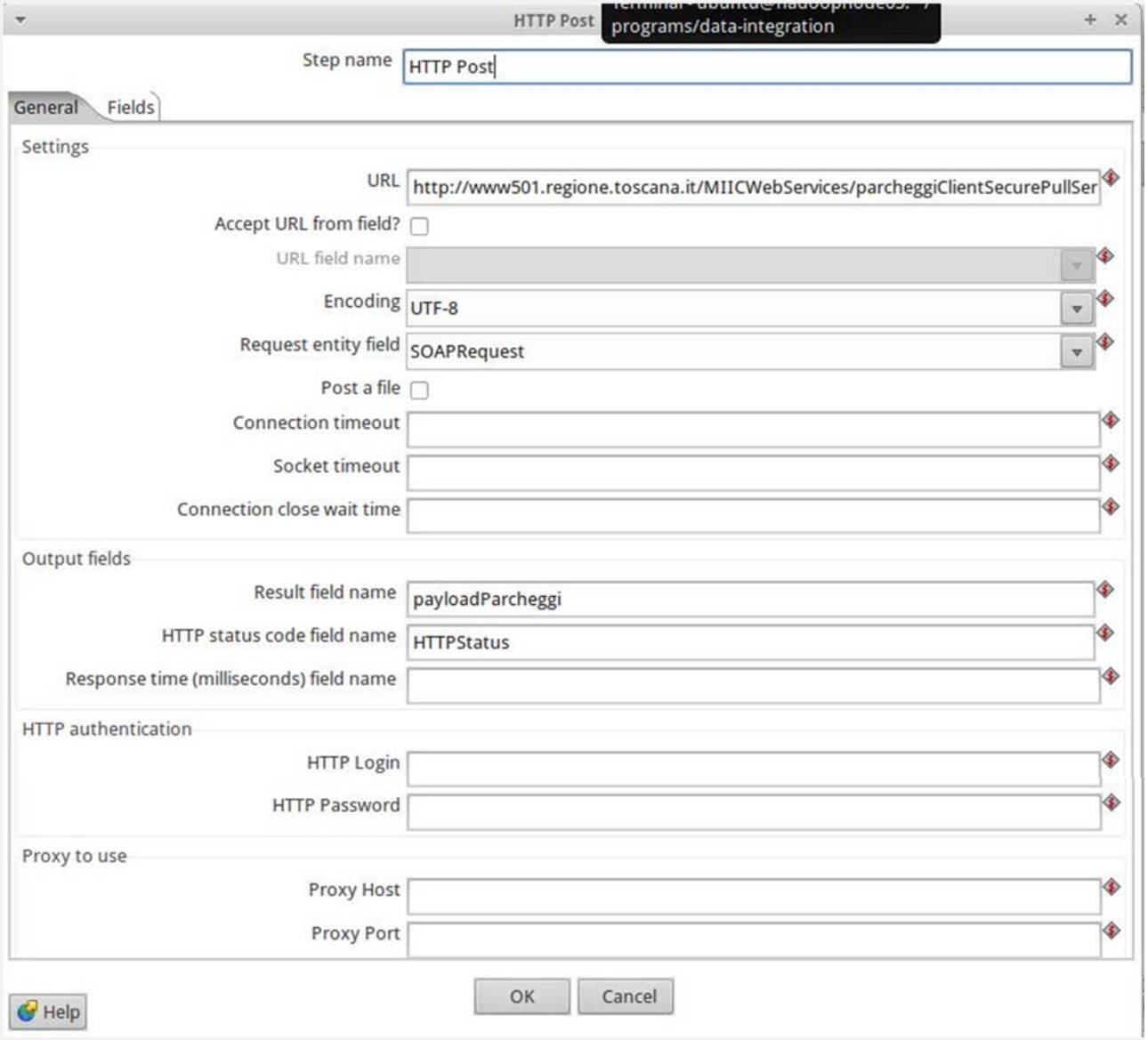


#	Alias	Key	Column family	Column name
1	FinalKey	Y		
2	actualDate	N	Family1	actualDate
3	carParkIdentity	N	Family1	carParkIdentity
4	carParkOccupancy	N	Family1	carParkOccupancy
5	carParkStatus	N	Family1	carParkStatus
6	catalog	N	Family1	catalog
7	exitRate	N	Family1	exitRate
8	fillRate	N	Family1	fillRate
9	numberOfVacantParkingSpaces	N	Family1	numberOfVacantParkingSpaces
10	occupiedSpaces	N	Family1	occupiedSpaces
11	process	N	Family1	process
12	situationRecordCreationTime	N	Family1	situationRecordCreationTime
13	situationRecordObservationTime	N	Family1	situationRecordObservationTime
14	supplierIdentification	N	Family1	supplierIdentification
15	timestamp	N	Family1	timestamp
16	totalCapacity	N	Family1	totalCapacity
17	validityStatus	N	Family1	validityStatus

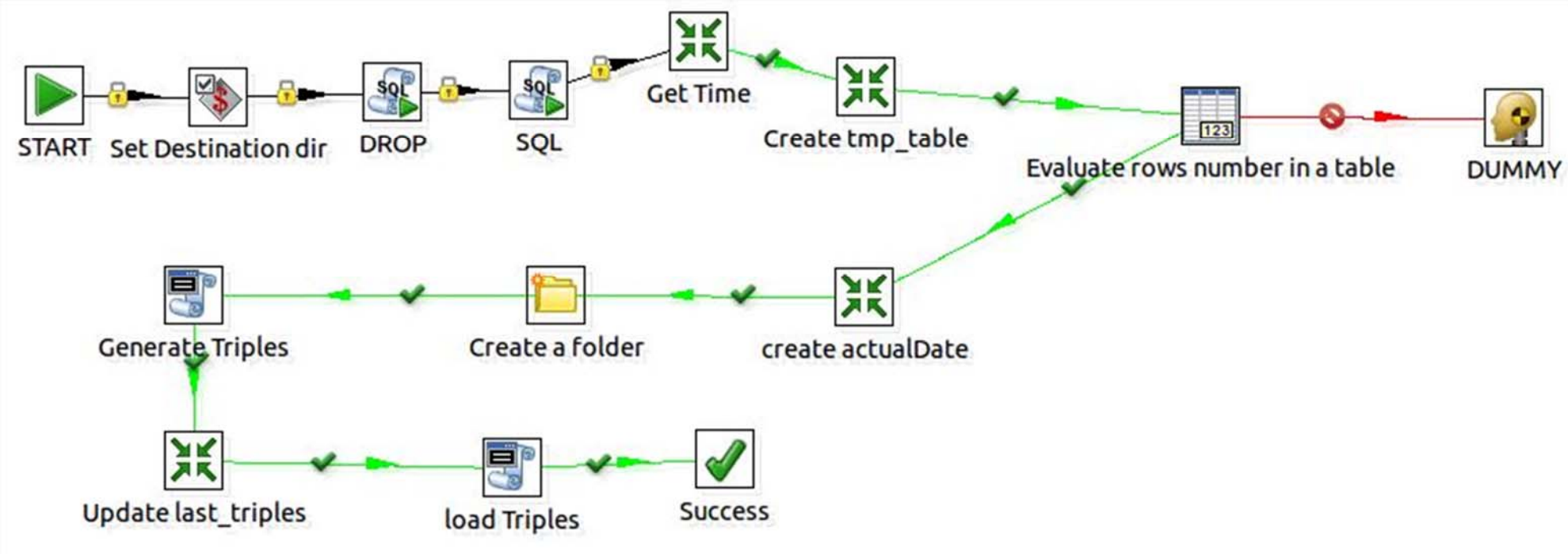
Ingestion phase

HTTP Post

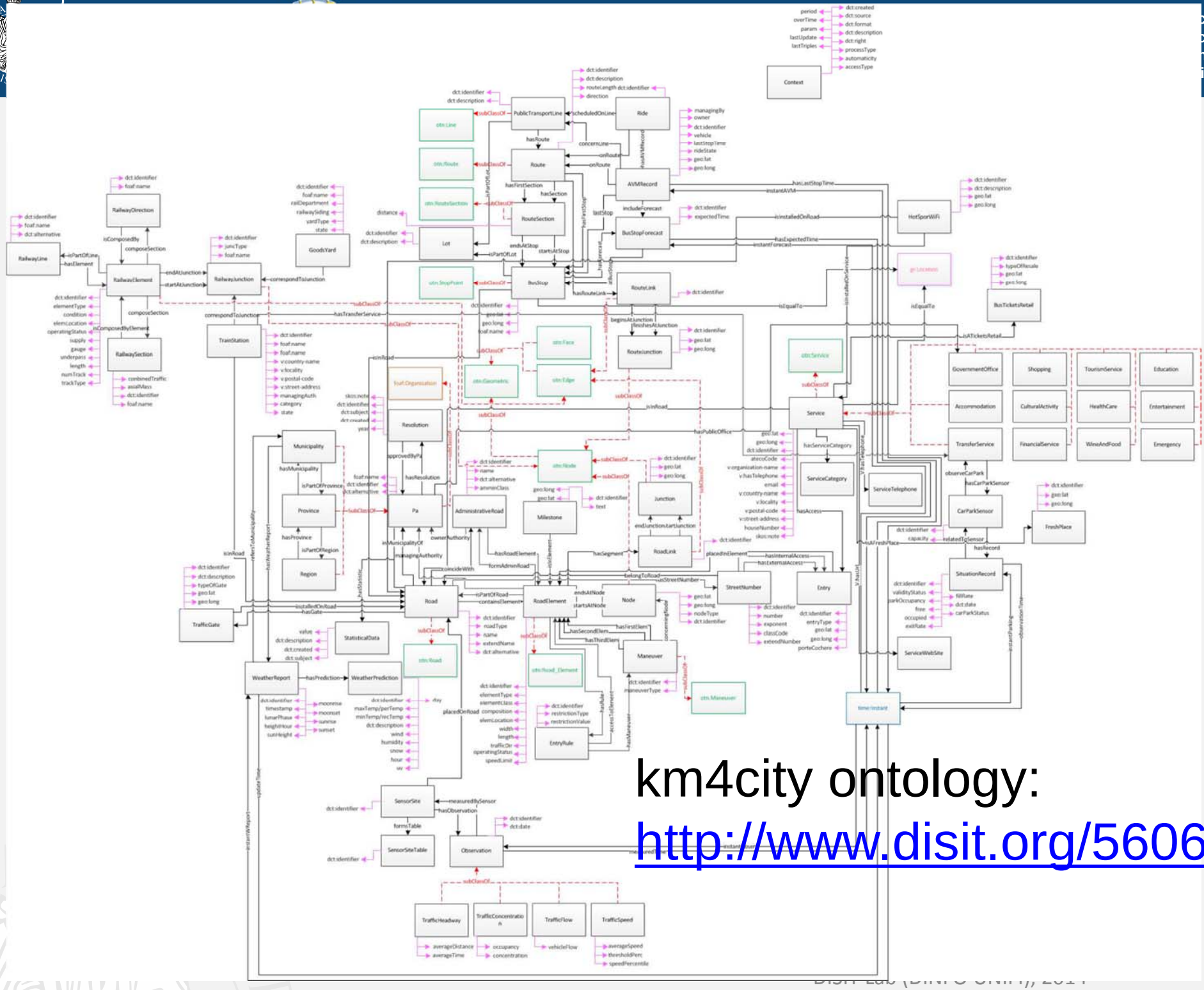
- This step performs the invocation of the web service using a SOAP (Simple Object Access Protocol) protocol.
- You can specify the service endpoint (URL).



RDF Triples generation phase



The triples are generated with the **km4city** ontology (**http://www.disit.org/5606**) and then loaded on OWLIM RDF store.


km4city ontology:
http://www.disit.org/5606

Process Scheduler

- For **Real Time data** (car parks, road sensors, etc.) the ingestion and triple generation processes should be performed periodically (no for **static data**).

- Use of a scheduler to manage periodic execution of ingestion and triple generation processes.
  - This tool throws the processes with predefined interval determined in phase of configuration.

Process Scheduler Work

	Service Data (static data)	Road / rail graph (static data)	MIIC (real time data)	LAMMA (real time data)	Total
DataSet	29	117	170	285	601
Processes	29	11	170	285	495

MIIC: parking data + traffic sensors data.

- 170 Processes scheduled every 1800 s, 48 times per day.
- → 8160 process execution per day

LAMMA: Weather forecasts.

- 285 Processes scheduled every 21600 s, 4 times per day.
- → 1140 process execution per day

Service Data, Road / rail graph: Processes started manually.

RDF Triples generated

Macro Class	Static Triples	Real Time Triples loaded
Administration	2.431	0
Local Public Transport	644.405	0
Metadata	416	0
Point of Interest	471.657	0
Sensors (Traffic and parking)	0	11.111.078
Street-guide	68.985.026	0
Temporal	0	1.715.105
Total	70.103.935	12.826.183

Triples monthly
21.691.882

Quality Improvement, QI

Class	%QI	Total rows	Class	%QI	Total rows
Accoglienza	34,627	13256	Georeferenziati	38,754	2016
Agenzie delle Entrate	27,124	306	Materne	41,479	539
Arte e Cultura	37,716	3212	Medie	42,611	116
Visite Guidate	38,471	114	Mobilita' Aerea	41,872	29
Commercio	42,105	323	Mobilita' Auto	38,338	196
Banche	41,427	1768	Prefetture	39,103	449
Corrieri	42,857	51	Sanità	42,350	1127
Elementari	42,004	335	Farmacie	42,676	2131
Emergenze	42,110	688	Università	42,857	43
Enogastronomia	42,078	5980	Sport	52,256	1184
Formazione	42,857	70	Superiori	42,467	183
Accoglienza	34,627	13256	Tempo Libero	25,659	564

Service data from Tuscany region.

%QI = improved service data percentage after QI phase.

fine

fine