**Schema extraction and data injection**

This plugin allows you to import existing relational databases in an empty polystore. This process is divided into two phases:

1. Schema and Data Extraction: the plugin will connect to your relational database(s) in order to extract a TML schema preserving the original relational constraints (identifiers, indexes, relations between tables, …). Moreover, the plugin will also generate SQL scripts necessary to migrate the existing data to the future polystore.
2. Data Injection: once the TML schema extracted, the user can manually create the polystore. Once the latter created, the user can execute the second phase: the plugin will take as input the SQL scripts generated by the previous phase in order to inject the data to the new polystore.

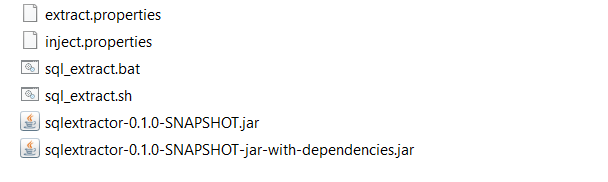
Below, we describe the procedure for installing and executing the plugin.

1. **Plugin installation**

For building the plugin with maven:

*mvn clean package*

The jar file and the required configuration files are generated:



1. **Schema and Data Extraction**

To connect to your relational databases, the plugin needs you to specify the required url and credentials. This information must be contained in the ‘*extract.properties*’ configuration file. In this file, you can specify the connection information of one or several relational databases. This is an example of structure for this configuration file:



Several parameters can be specified :

* URL : the user must specify the JDBC url necessary to connect to the relational database. (*jdbc:****dbmsName****://****ip****:****port****/****databaseName***)

**Important**: if you define a MariaDB database connection, specify MySQL in the url instead (*jdbc:mysql://…).*

* DRIVER : the user must specify the JDBC driver necessary to connect to the relational database
* USER : the user must specify a login having the read rights
* PASSWORD : the user must specify a password
* SCHEMA : the user must specify the database schema name to connect
* DOCUMENT\_SPLIT : this param is optional and allows the user to specify the database column(s) that will be migrated in a document database (currently MongoDB). Example: at line 6, the user indicates that column ‘history’ of table ‘Employee’ must be migrated in a document database (in the polystore) as well as column ‘nbOfEmployees’ of table ‘Department’.

NB: you can use a suffix for the above parameters when you have several databases to connect. Example: URL2 will be the url of the second database, URL3 will be the url of the third database, etc.

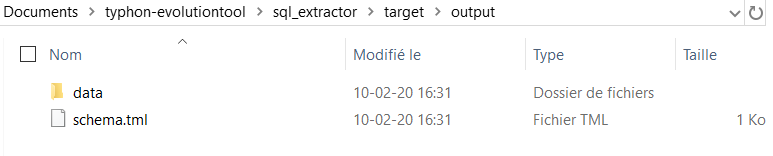
* MAX\_SQL\_QUERIES\_PER\_FILE: while the plugin will connect the relational database(s) to generate the SQL migration scripts, the user can specify the maximal number of insert queries per SQL scripts. This param can be useful in presence of a huge amount of records to migrate.
* MAX\_JSON\_RECORDS\_PER\_FILE: the ‘DOCUMENT\_SPLIT’ param allows the user to specify the columns to migrate in a document database; in this case, the plugin will generate json files containing the json records to migrate in the document database. Like the SQL migration scripts, the user can specify the maximal number of json records per file.

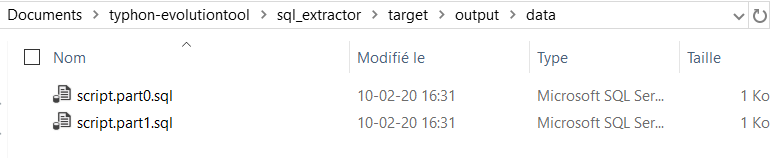
Once you have edited the configuration file, you can launch the extraction process according to your operating system:

* Windows: *sql\_extract.bat -extract extract.properties output*
* Linux: *bash sql\_extract.sh -extract extract.properties output*

Where *extract.properties* is the configuration file and *output* the output directory where the TML schema and migration scripts will be saved. Warning: this output directory must be existing.

Once the extraction is terminated, you should obtain (1) the TML schema and (2) the migration scripts.

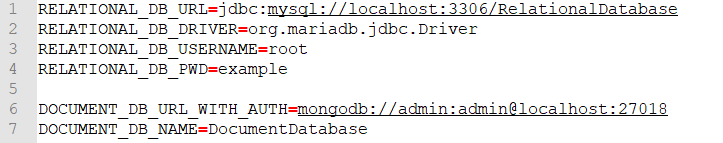




1. **Data Injection**

Before injecting data into your polystore, you must manually deploy your polystore by using the generated TML schema.

Once your polystore is deployed, you can inject the data. However, you must firstly specify the url and credentials required to connect the relational database and the document database of your polystore. This information must be defined in the ‘*inject.properties*’ file. This is an example of structure for this file:



Several parameters are required:

* RELATIONAL\_DB\_URL: the JDBC url required to connect to the relational database of your polystore
* RELATIONAL\_DB\_DRIVER: the JDBC driver required to connect to the relational database of your polystore (at the moment, only mariadb is implemented)
* RELATIONAL\_DB\_USERNAME: a login having the read rights.
* RELATIONAL\_DB\_PWD: the password
* DOCUMENT\_DB\_URL\_WITH\_AUTH: the url necessary to connect the document database of your polystore (at the moment, only MongoDB is implemented). Warning: this url must contain the username and password as well. In this example: *mongodb://username:password@ip:port*

Once you have edited the configuration file, you can launch the data injection process according to your operating system:

* Windows: *sql\_extract.bat -inject inject.properties output/data*
* Linux: *sql\_extract.sh -inject inject.properties output/data*

Where *inject.properties* is the configuration file described above and *output/data*, the directory containing the SQL/JSON migration scripts generated during the previous extraction phase.

Once the injection is terminated, your polystore will be operational and ready to use.