Metamorfose: a data transformation framework based on Apache Spark

Evandro Miguel Kuszera[1 2], **Leticia M. Peres**[2], **Marcos Didonet Del Fabro**[2]

[1]Federal Technological University of Paraná (UTFPR) – Dois Vizinhos – PR – Brazil

[2]Federal University of Paraná (UFPR) – C3SL Labs – Curitiba, PR – Brazil

{evandrokuszera@utfpr.edu.br, lmperes,marcos.ddf}@inf.ufpr.br

Abstract. *On the context of large availability of open data, it is important to provide interactive solutions where a data transformation workflow can be easily deployed and developed. This article presents Metamorfose*[1]*, a framework for data transformation based on large-scale data processing engine Apache Spark. Through the presented framework, the user can set up an interactive transformation workflow for loading data, defining mappings between source and target fields, performing transformations, and persisting the data. The user can define transformation functions in Java or Javascript and create chains of transformation where a transformation result can be used as input to the next.*

1. Introduction

The ability to exchange and integrate data is crucial for many types of applications, especially with the diversity of data formats (e.g., structured or semi-structured). With the emergence of open data, new solutions must be developed to easily explore and analyze the large amount of data generated. Data transformation systems (DTS) are important in these scenarios. The function of these systems is to perform transformations where instances of a source schema are translated to instances of a target schema. DTSs are composed of a set of translation tasks that can be defined as a quadruple $\{S, T, I_\text{s}, I_\text{e}\}$, where S is the source schema, T is the target schema, I_s is a valid instance of S and I_e a valid instance of T generated by applying the desired transformations over I_s [Mecca et al. 2012]. A set of mappings M is used to denote the relationship between expected output and provided input.

Schema mapping and ETL (*Extract*, *Transform*, *Load*) tools are widely used to transform and integrate diverse sources of data. Transformations in schema mapping tools are based on declarative specifications (such as SQL, for example), not providing the exact method they are implemented and executed. ETL tools define transformations as data flows over a graph of operators. Operators range from simple mapping between tables to complex data join and splitting operations, where the user can choose a specific implementation to a data transformation [Dessloch et al. 2008].

CLIO [Haas et al. 2005] was one of the pioneering tools to execute schema mappings. Others have recently been proposed as ++Spicy [Marnette et al. 2011], LLunatic [Geerts et al. 2014] and EXLEngine [Atzeni et al. 2017]. Clover ETL [CloverETL 2018], Pentaho Kettle [Pentaho Kettle 2018], OpenRefine [OpenRefine 2018] and Talend Open

[1]Metamorfose video: https://youtu.be/ta9mXuCeIwM

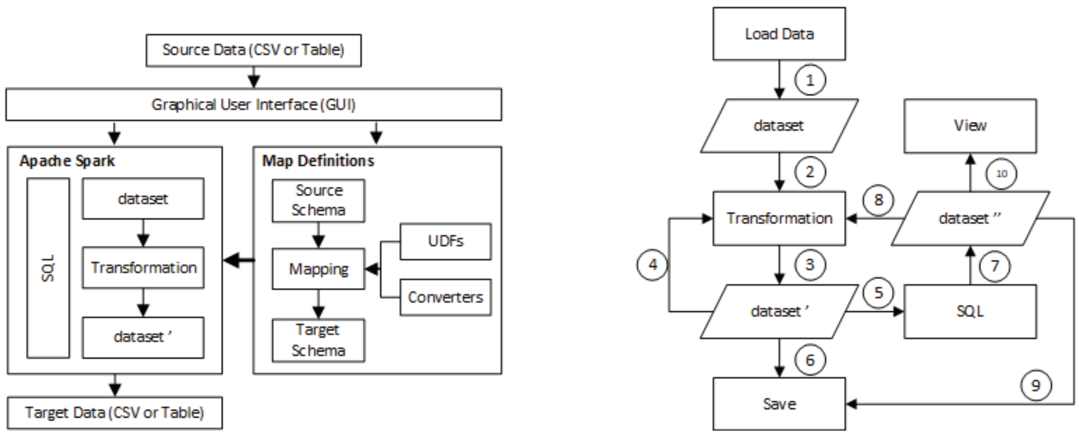
Studio [Talend 2018] are examples of ETL tools. Schema mapping tools are intended for designers who need to express the main components of a data transformation task. While ETL tools are intended for users interested in efficient implementations of a data transformation task. Depending on the context, the use of these tools is not trivial, requiring a complex installation and configuration process to perform simple data transformations. In these scenarios concise tools are an option, not requiring the mastery of a large set of technologies to perform data transformations.

In this paper we present Metamorfose, a framework for data transformation based on large-scale data processing engine Apache Spark [Zaharia et al. 2016]. The choice of using Apache Spark was motivated by its support of a wide range of processing workloads and abstractions provided to manipulate datasets. Through Metamorfose the user can load data, define mappings between source and target fields, execute transformations, and persist data. Metamorfose is extensible and independent of the underlying data format. New data sources can be added by creating Spark datasets. In the current version, it can load and persist data as CSV file or relational database table. As a contribution, the framework allows the definition of an interactive transformation workflow that can be integrated with user-defined transformation functions implemented in Java or Javascript. The use of Javascript makes it flexible to add new transformation functions. In the next section the framework will be presented.

2. Metamorfose

The architecture of Metamorfose and mapping definitions will be presented below.

2.1. Architecture of Metamorfose



(a) Architecture of Metamorfose. (b) Execution Flow.

Figure 1. Architecture and Execution Flow of Metamorfose.

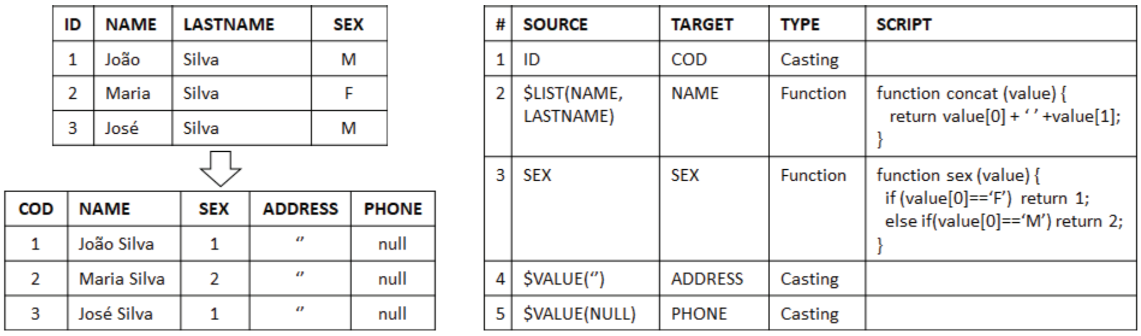
The Figure 1 (a) presents the main components of Metamorfose. Through the graphical user interface it is possible to load data source for transformation, execute data queries through Spark SQL (Apache Spark module) and specify mappings between source and target schema. Each data source is loaded as a Spark dataset. To transform the data, map functions are executed on the source dataset according to the user-defined mappings (Map Definitions module).

The Metamorphosis execution flow over one data source is presented in Figure 1 (b). In (1) the data is loaded as a Spark dataset. Based on the user mappings the dataset is transformed (2) producing (3) dataset' as a result. Dataset' can be used: (4) as input to a new transformation, (5) as input to SQL query or (6) can be persisted to CSV file or relational database table. The SQL query execution on dataset' produces new (7) dataset". Dataset" can be used for new data transformations (8), for persistence (9) or visualization (10). This flow allows to flexible define chains of transformations. Transformations can be applied over the current dataset or can create a new dataset with the resulting data.

2.2. Mapping Definitions

A data schema can be defined as $S = \{s_1, ..., s_n\}$, where s_i represents a data field of the schema S. From a source schema S is possible to derive a target schema T applying transformations on the field values of S. Transformations can be represented as a set of mappings $M = \{(s_1, t_1, f_1), ..., (s_n, t_n, f_n)\}$, where s_i represents one or more source fields, t_i a target field and f_i a transformation function. Transformation chains can be defined using the target schema T as the source schema for next transformation. To persist T data for an existing data source, T must correspond to the data source schema.

Figure 2 (a) shows an example of transformation over a table with person data. ID field is mapped to the COD field in the target schema. NAME and LASTNAME fields are mapped to the NAME field. SEX field data are transformed from 'F' and 'M' to numerical values 1 and 2, respectively. ADDRESS and PHONE fields only exist in the target schema and they will receive an empty string and null value, respectively.



ID	NAME	LASTNAME	SEX
1	João	Silva	M
2	Maria	Silva	F
3	José	Silva	M

COD	NAME	SEX	ADDRESS	PHONE
1	João Silva	1	''	null
2	Maria Silva	2	''	null
3	José Silva	1	''	null

(a) Source and target schemas.

#	SOURCE	TARGET	TYPE	SCRIPT
1	ID	COD	Casting	
2	$LIST(NAME, LASTNAME)	NAME	Function	function concat (value) { return value[0] + ' ' +value[1]; }
3	SEX	SEX	Function	function sex (value) { if (value[0]=='F') return 1; else if(value[0]=='M') return 2; }
4	$VALUE('')	ADDRESS	Casting	
5	$VALUE(NULL)	PHONE	Casting	

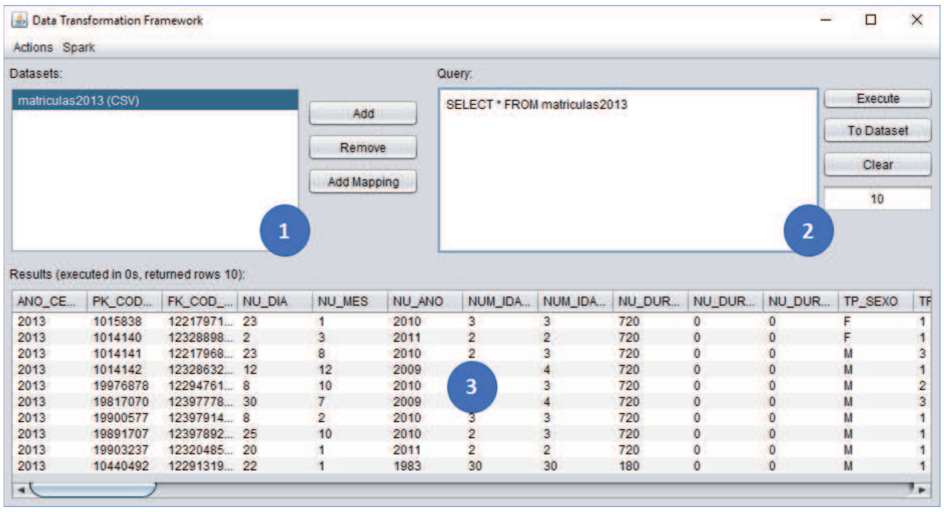
(b) Transformations settings.

Figure 2. Mapping and transformation example.

Mappings are declarative statements that establish correspondence between source and target fields, which can be integrated with data transformation functions. Figure 2 (b) shows the mappings used to transform the data. $VALUE() and $LIST() are reserved words and act as Metamorfose operators. Line 1 displays a one-to-one mapping between source and target, with *Casting* data type conversion. In line 2 the word $LIST denotes a many-to-one mapping, where a list of source fields is sent to user-defined function (UDF) *concat*. In this simple example a concatenation operation is performed. However, complex functions can be defined in Javascript. In line 3 is an example of one-to-one mapping using UDF *sex*. In lines 4 and 5 the word $VALUE is used to insert constant values into the target schema. This set of mappings can be persisted as a JSON file for future use.

3. Demonstration

The Metamorfose tool provides a graphical user interface to ease the definition of mappings and data transformation execution. In the demonstration we will present the main flow of tool: data loading, mapping definitions and transforming execution.



SELECT * FROM matriculas2013

Results (executed in 0s, returned rows 10):

ANO_CE...	PK_COD...	FK_COD_...	NU_DIA	NU_MES	NU_ANO	NUM_IDA...	NUM_IDA...	NU_DUR...	NU_DUR...	NU_DUR...	TP_SEXO	TF
2013	1015838	12217971...	23	1	2010	3	3	720	0	0	F	1
2013	1014140	12328898...	2	3	2011	2	2	720	0	0	F	1
2013	1014141	12217968...	23	8	2010	2	3	720	0	0	M	3
2013	1014142	12328632...	12	12	2009	[illegible]	4	720	0	0	M	1
2013	19976878	12294761...	8	10	2010	[illegible]	3	720	0	0	M	2
2013	19817070	12397778...	30	7	2009	[illegible]	4	720	0	0	M	3
2013	19900577	12397914...	8	2	2010	3	3	720	0	0	M	1
2013	19891707	12397892...	25	10	2010	2	3	720	0	0	M	1
2013	19903237	12320485...	20	1	2011	2	2	720	0	0	M	1
2013	10440492	12291319...	22	1	1983	30	30	180	0	0	M	1

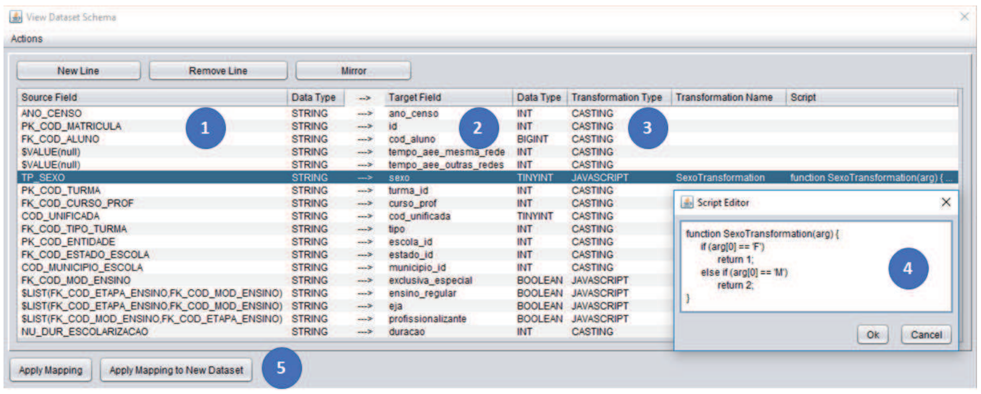
Figure 3. Metamorfose main screen.

3.1. Data Loading and Exploration

The Figure 3 presents the Metamorfose main screen. In (1) the user can load or remove datasets, view the list of available datasets, or add mappings to selected dataset. *Add Mapping* button shows the mapping screen (details in the next section). The user can load datasets from CSV files or relational database tables (Postgres, for example). In (2) the user can submit SQL queries (using Spark SQL) on the loaded datasets. In (3) the user can view and explore the data produced by the SQL query. SQL query results can be added as a new dataset in the list of available datasets pressing *To Dataset* button in (2). In the example of Figure 3 a CSV file (*matriculas2013*) with four million records was loaded and a SQL query was submitted to visualize the data in the result table.

3.2. Mappings

The user must select a dataset from the list and click the *Add Mapping* button to define mappings. The mapping screen appears and the selected dataset scheme (source) is loaded automatically. The Figure 4 presents a mapping previously defined and loaded from a JSON file. In (1) the fields and data types of the source dataset are displayed. In (2) the user must specify the fields and data types of the target dataset. In (3) the user must define the type of the transformation (*Casting* or *Javascript*). For transformations involving Javascript UDFs the user must enter the code in the edit screen in (4). After mapping definitions between source and target fields, two options are available (5): the *Apply Mapping* button applies the mappings over the source dataset and modify your data and schema, or the *Apply Mapping to New Dataset* button creates a new dataset from source dataset. The latter option allows it to define transformation chains. The new datasets are added in the list of available datasets for further use. In Figure 4 there are mappings for field renaming, insertion of constant values in the target dataset and *Javascript* UDFs where one or more source fields are transformed to one target field according with UDF logic.



Source Field	Data Type	-->	Target Field	Data Type	Transformation Type	Transformation Name	Script
ANO_CENSO	STRING	--->	ano_censo	INT	CASTING		
PK_COD_MATRICULA	STRING	--->	id	INT	CASTING		
FK_COD_ALUNO	STRING	--->	cod_aluno	BIGINT	CASTING		
$VALUE(null)	STRING	--->	tempo_aee_mesma_rede	INT	CASTING		
$VALUE(null)	STRING	--->	tempo_aee_outras_redes	INT	CASTING		
TP_SEXO	STRING	--->	sexo	TINYINT	JAVASCRIPT	SexoTransformation	function SexoTransformation(arg) { ...
PK_COD_TURMA	STRING	--->	turma_id	INT	CASTING		
FK_COD_CURSO_PROF	STRING	--->	curso_prof	INT	CASTING		
COD_UNIFICADA	STRING	--->	cod_unificada	TINYINT	CASTING		
FK_COD_TIPO_TURMA	STRING	--->	tipo	INT	CASTING		
PK_COD_ENTIDADE	STRING	--->	escola_id	INT	CASTING		
FK_COD_ESTADO_ESCOLA	STRING	--->	estado_id	INT	CASTING		
COD_MUNICIPIO_ESCOLA	STRING	--->	municipio_id	INT	CASTING		
FK_COD_MOD_ENSINO	STRING	--->	exclusiva_especial	BOOLEAN	JAVASCRIPT		
$LIST(FK_COD_ETAPA_ENSINO,FK_COD_MOD_ENSINO)	STRING	--->	ensino_regular	BOOLEAN	JAVASCRIPT		
$LIST(FK_COD_ETAPA_ENSINO,FK_COD_MOD_ENSINO)	STRING	--->	eja	BOOLEAN	JAVASCRIPT		
$LIST(FK_COD_MOD_ENSINO,FK_COD_ETAPA_ENSINO)	STRING	--->	profissionalizante	BOOLEAN	JAVASCRIPT		
NU_DUR_ESCOLARIZACAO	STRING	--->	duracao	INT	CASTING		

```
function SexoTransformation(arg) {
    if (arg[0] == 'F')
        return 1;
    else if (arg[0] == 'M')
        return 2;
}
```

Figure 4. Metamorfose mapping definition screen.

3.3. Data Transformation

After defining the mappings the user can visualize or persist the contents of the dataset through Metamorfose main screen (Figure 3). The persistence options are CSV and JDBC and are available through the *Actions* menu. At this point the data transformations are executed by Apache Spark.

4. Evaluation

Yearly, the National Institute of Educational Studies and Research Anísio Teixeira (INEP)[2] provides open data about educational census in Brazil. These data including enrollments, teachers, schools and classes in CSV format that exceed 69 million records per year. This data source can be used to analyze the situation of education in Brazil. However, the format of data is not rigid, variations are found in every release, such as adding new fields, changing and removing existing fields, for example. Metamorfose was used to map and transform CSV data to a predefined PostgreSQL database.

We have performed experiments using Brazilian enrollment data from the year 2013. Due to the number of records, the data were made available in five CSV files representing the South, Southeast, Midwest, North and Northeast regions of the country (around 55 million records). It were defined 82 mapping fields between CSV file and target relational schema. One-to-one, many-to-one mappings and Javascript UDFs were used to transform data.

Table 1. Number of records and execution time of experiment performed.

Brazil Region	Records	Time
Midwest	4.038.979	10 min
South	7.276.108	18 min
Northeast	16.729.543	41 min
Southeast + North	27.379.690	65 min

Metamorfose was run on a machine with Intel Core i7 2.5GHz processor, 16GB RAM, Windows 10 Home and Postgres 10. The Table 1 shows the number of records by region of Brazil and execution time to transform and load the data to relational database.

[2]INEP: http://www.inep.gov.br/

This experiment aimed to validate the tool on a data set of moderate size. It is possible to observe that the execution time has linear growth in respect of the number of records. Future experiments will be carried out considering more records and processing nodes.

5. Conclusions

In this paper we have presented the Metamorfose tool, a framework for data transformation built on large-scale data processing engine called Apache Spark. Through a graphical user interface is possible to define an interactive data transformation workflow where the user can loading data, defining mappings, performing transformations, exploring and persisting the results. A mapping definition can be integrated with data transformation functions implemented in Java or Javascript, providing a flexible way to define complex transformations. It is possible to execute SQL queries to filter, aggregate and join the datasets before or after every data transformation. The results can be persisted as CSV file or as relational database table. Metamorfose was validated using real open data about education in Brazil. As future work, new features will be added including transformation functions, supporting for other data sources (JSON and XML) and support for other database models.

References

Atzeni, P., Bellomarini, L., Bugiotti, F., and De Leonardis, M. (2017). Executable Schema Mappings for Statistical Data Processing. *Distributed and Parallel Databases*.

CloverETL (2018). http://www.cloveretl.com/. Accessed on 6 January 2018.

Dessloch, S., Hernandez, M. A., Wisnesky, R., Radwan, A., and Zhou, J. (2008). Orchid: Integrating schema mapping and etl. In *2008 IEEE 24th ICDE*, pages 1307–1316.

Geerts, F., Mecca, G., Papotti, P., and Santoro, D. (2014). That's All Folks! Llunatic Goes Open Source. *PVLDB*, 7(13).

Haas, L. M., Hernández, M. A., Ho, H., Popa, L., and Roth, M. (2005). Clio grows up: From research prototype to industrial tool. In *Proceedings of the 2005 ACM SIGMOD*, SIGMOD '05, pages 805–810, New York, NY, USA. ACM.

Marnette, B., Mecca, G., Papotti, P., Raunich, S., and Santoro, D. (2011). ++spicy: an opensource tool for second-generation schema mapping and data exchange. *PVLDB*, 4(12).

Mecca, G., Papotti, P., Raunich, S., and Santoro, D. (2012). What is the iq of your data transformation system? In *Proc. of the 21st ACM CIKM*, CIKM 2012, pages 872–881.

OpenRefine (2018). http://openrefine.org/. Accessed on 4 January 2018.

Pentaho Kettle (2018). https://www.hitachivantara.com/. Accessed on 4 April 2018.

Talend (2018). https://www.talend.com/. Accessed on 5 Febuary 2018.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., and Stoica, I. (2016). Apache spark: A unified engine for big data processing. *Commun. ACM*, 59(11):56–65.