LinkedPipes ETL: Evolved Linked Data Preparation

Jakub Klímek[1,2,3](✉), Petr Škoda[1,2], and Martin Nečaský[1]

[1] Faculty of Mathematics and Physics, Charles University in Prague, Malostranské nám. 25, 118 00 Praha 1, Czech Republic
{klimek,helmich,necasky}@ksi.mff.cuni.cz

[2] University of Economics, Prague, Nám. W. Churchilla 4, 130 67 Praha 3, Czech Republic

[3] Faculty of Information Technology, Czech Technical University in Prague, Thákurova 9, 160 00 Praha 6, Czech Republic

Abstract. As Linked Data gains traction, the proper support for its publication and consumption is more important than ever. Even though there is a multitude of tools for preparation of Linked Data, they are still either quite limited, difficult to use or not compliant with recent W3C Recommendations. In this demonstration paper, we present LinkedPipes ETL, a lightweight, Linked Data preparation tool. It is focused mainly on smooth user experience including mobile devices, ease of integration based on full API coverage and universal usage thanks to its library of components. We build on our experience gained by development and use of UnifiedViews, our previous Linked Data ETL tool, and present four use cases in which our new tool excels in comparison.

Keywords: Linked Data · RDF · ETL · Transformation

1 Introduction

There is already a multitude of tools for Linked Data preparation. The goal of each such tool should be to ease as much as possible the shift from internal or open data in relational databases or Excel, CSV, XML and JSON files to Linked Data. What is often overlooked is the fact, that the target user of such tool is not only a computer scientist willing to use a system with unfriendly interface and sketchy documentation and a steep learning curve. One of the typical users is a government employee in charge of data in legacy systems who just learned about Linked Data and is enthusiastic to publish it. For him, the tool needs to be easy to install, well documented, user friendly, extensible and ideally ready to

This work was supported in part by the Czech Science Foundation (GAČR), grant number 16-09713, in part by the project SVV-2016-260331 and in part by the H2020 project no. 645833 (OpenBudgets.eu).

be integrated into the systems of his government office. This is actually our real-world experience gained by supporting the Czech Social Security Administration in publishing their Linked Data[1].

In this paper, we present LinkedPipes ETL (LP-ETL), our Linked Data preparation tool, which we created based on our experience with development, usage and support of UnifiedViews [1], our previous Linked Data ETL tool, and the feedback we got for it. With LP-ETL we focus on APIs and configurations based on Linked Data principles, so that the tool is usable by and ready to be integrated with other software. We design the user interface based on Google Material Design[2], which includes responsiveness, native support for mobile devices and overall ease of use. We make the typical use cases as easy as possible and we do not bother the user with unnecessary interactions.

The rest of the paper is structured as follows. In Sect. 2 we survey related work. In Sect. 3 we introduce the basic ETL concepts and our tool. In Sect. 4 we present the qualities of LP-ETL on four use cases covering its typical usage. In Sect. 5 we indicate our future plans with the tool and we conclude.

2 Related Work

ETL (Extract Transform Load) is a designation for a data processing task where data is first gathered (extracted), then transformed and at the end, loaded to a local database or file system. LP-ETL is based on our experience gained from *UnifiedViews* (UV), which is also an RDF-enabled ETL tool. UV has various problems which we address in LP-ETL. From the programmer perspective, the biggest flaw is an incomplete API, which only supports execution and monitoring of existing pipelines, which makes it impossible to integrate into a system that would create the pipelines through the API. From the user perspective, the biggest flaw was the frontend application, which was unfriendly and prone to crashes, lags and loss of data caused e.g. by connection interruptions. Another ETL solution is the *LinDa Workbench* [2]. It allows the users to do simple tabular to RDF transformations from CSV/Excel files and relational DBs and run analyses and visualizations on top of the transformed RDF data. However, it does not support further data linking, cleansing, publishing or cataloging and it is not documented how it can be extended with additional plugins. A cloud-based open data publishing platform *DataGraft*[3] is similar to the LinDa Workbench. It also focuses on tabular data to RDF transformation and has a user-friendly interface. At the end of the transformation, the user gets a data page and a SPARQL endpoint for querying his data. However, it also does not support linking, cleansing nor cataloging of the data and it is not documented how additional plugins can be created. In addition, a cloud based solution is often not acceptable for government offices and their data because of trust issues and they prefer a software that they can install or integrate into their own, secured environment.

[1] https://data.cssz.cz.
[2] https://design.google.com.
[3] https://datagraft.net/.

3 LinkedPipes ETL (LP-ETL)

LinkedPipes ETL[4] runs using Java and Node.js and it can be compiled from source using Git and Maven. It contains a backend, which runs the data transformations and exposes the APIs, and a frontend, which is a web application and provides a pipeline editor and an execution monitor to the user. A pipeline is a repeatable data transformation process consisting of configurable components, each responsible for an atomic data transformation task, such as a SPARQL query of CSV to RDF transformation.

4 Demonstrated Use Cases

The individual use cases of LinkedPipes ETL will be demonstrated on the live demo instance[5]. The aim is to show how LP-ETL helps with the creation and debugging of typical ETL pipelines with Linked Data output. To demonstrate the complexity of typical ETL pipelines let us discuss 182 datasets published by the COMSODE project team[6]. For each of the datasets, a UV pipeline was developed. In total, 89 different components were used in these pipelines. These are responsible for gathering input data, its transformation to RDF including alignment with various ontologies and linkage to third-party datasets and for cleansing of datasets. Each pipeline contained an average number of 23 component instances. The execution time of the pipelines ranged from several minutes in the case of transformation of small XLS files with codelists to RDF representation in SKOS, to several days in the case of transformation of the Czech Registry of Addresses to its Linked Data representation including alignment with various ontologies and linkage to external data sources. This shows that real-world ETL pipelines are complex, cover many processing steps and may run for a very long time. Therefore, the support for smarter workflow for creating ETL pipelines and their advanced debugging demonstrated by our use-cases is imperative.

4.1 Smarter Workflow for Creating ETL Pipelines

In this use case, we will demonstrate the process of creating a simple ETL pipeline (see Fig. 1), from downloading the data sources, transforming them using SPARQL and other techniques and loading them to a triplestore. This will be demonstrated on the CSV open data of the Czech Trade Inspection Authority[7]. The development of the pipelines in LinkedPipes ETL is governed by an evolved workflow. In each step where a user adds another component to the pipeline, the components are recommended according to various criteria. The first criterion is *data unit compatibility*, which is checked and only components that consume what the previous component produces will be offered. Another criterion is the

4 http://etl.linkedpipes.com.
5 http://demo.etl.linkedpipes.com.
6 http://data.comsode.eu/.
7 http://www.coi.cz/userdata/files/dokumenty-ke-stazeni/open-data/kontroly.csv.

probability of the appearance of the component after the last component in the pipeline, to which it will be connected. This probability is based on existing pipelines in the instance. The last criterion is *full text search and tag based search*. Each component has a name and a set of tags assigned to it. The user can start typing and the offered components get filtered both according to their names and the tags. The usefulness of tags can be demonstrated on the Decompress component that unpacks ZIP and BZIP2 files. When a user searches for a component that could unzip his input files, he may type *unzip* or *zip*, which will not suggest the Decompress component simply based on its name. This is were tags help, the Decompress component can have tags for each of the supported formats (i.e. *zip*, *bzip2*) and for the actions (*unzip*, *unbzip*, *decompress*, *unpack*) which increases the chances to find the required component even without the knowledge of its name.

4.2 Advanced Debugging Support

In a typical case of development of an ETL pipeline, errors happen. They can be caused for example by external services failing or misconfiguration of components. In these situations, having a proper support for debugging is crucial. Imagine an ETL pipeline that runs for a week, or a complex pipeline consisting of a hundred components, both of which happen in the real world. At any time, the pipeline may fail while already having done some substantial work. In UnifiedViews, when this happened, there were only two options. The user could either rerun the pipeline or search the server for the RDF data.

In LP-ETL we developed a new *debug from* functionality, which allows the user to resume a failed pipeline from an arbitrary point of its previous execution after it is fixed. This saves some effort, especially for long running pipelines, since the user can fix the component that was misconfigured and continue execution from that point. In addition, LinkedPipes ETL implements the *debug to* functionality, which allows the user to run a part of a pipeline from its start to a certain point. A typical use case is when following the *progressive enrichment* Linked Data pattern[8]. First, a complete pipeline with simple transformation of input data is developed, including data gathering, loading and cataloging. Later, the data transformation is enhanced e.g. by improving a SPARQL query in one component, leaving the rest of the pipeline intact. This improved debugging functionality will be demonstrated on the same pipeline as the use case in Subsect. 4.1. There is a pipeline which failed on the *RDF to file* component due to misconfiguration. We fix it and restart the pipeline from this component. Next, the pipeline fails again (see Fig. 1) on the other *RDF to file*, again due to misconfiguration. The dark green bordered components are done in the first execution, the light green bordered components done in the current one and the red component is the one which failed.

[8] http://patterns.dataincubator.org/book/progressive-enrichment.html.

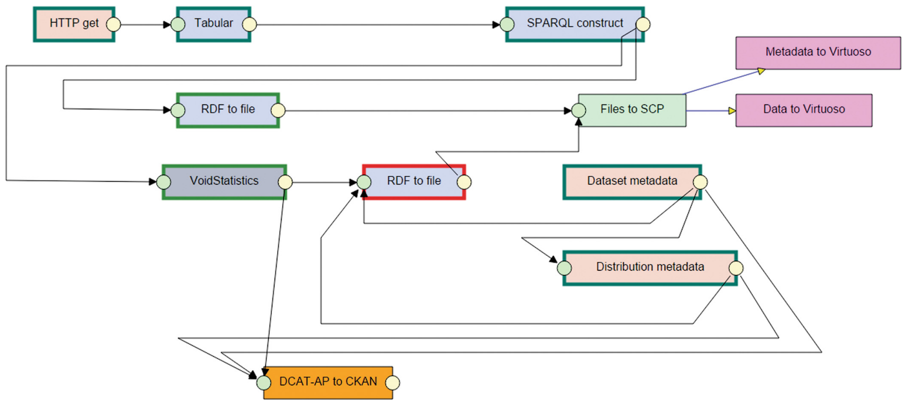

Fig. 1. Pipeline in LinkedPipes ETL fails again, fresh and old executions distinguished (Color figure online)

4.3 Full API Coverage and RDF Configuration

LinkedPipes ETL can be integrated in larger platforms and extended by alternate user interfaces. It was designed with this in mind from the beginning. All of the components have REST API interfaces and use RDF for configuration of components and pipelines. The same interface is used by LinkedPipes ETL itself and this means that it can be also used by any other software. This will be demonstrated by showing how a pipeline can be uploaded, executed and its state monitored from the command line using *curl*.

4.4 Sharing Pipelines and Fragments

LinkedPipes ETL eases sharing of pipelines and pipeline fragments. Since pipeline definitions are simple RDF files in JSON-LD serialization, they can be shared on the web or on GitHub for reuse. We will demonstrate this on the pipeline from Subsects. 4.1 to 4.2 where we will download it to demonstrate the debugging capabilities and the fully working version from the LinkedPipes ETL web documentation.

5 Future Work and Conclusions

We are currently developing *wizards* for typical ETL tasks, e.g. conversion of a CSV source to Linked Data similar to DataGraft. This task in its basic form requires only the location of the source CSV file and the ID of the target dataset, that can be used for the target folder structure, file names and IRIs. The pipeline consisting of the Tabular component, which implements the CSV on the Web W3C Recommendation[9], RDF and files conversion, SPARQL queries and loaders can then be generated based on this input and the settings of an LP-ETL

[9] https://www.w3.org/TR/csv2rdf/.

instance. An *import tool for UnifiedViews pipelines* is also being developed. In this paper we presented LinkedPipes ETL, a lightweight ETL tool for Linked Data preparation. We demonstrated its added value on four use cases, comparing it mainly to UnifiedViews, our previous ETL tool. LinkedPipes ETL is currently in use by the OpenBudgets.eu project[10].

References

1. Knap, T., Škoda, P., Klímek, J., Nečaký, M.: UnifiedViews: towards ETL tool for simple yet powerfull RDF data management. In: Proceedings of the Dateso 2015 Annual International Workshop on DAtabases, TExts, Specifications and Objects, Nepřívěc u Sobotky, Jičín, Czech Republic, 14 April 2015, pp. 111–120 (2015)
2. Thellmann, K., Orlandi, F., Auer, S.: LinDA - visualising and exploring linked data. In: Proceedings of the Posters and Demos Track of 10th International Conference on Semantic Systems - SEMANTiCS 2014, Leipzig, 9 (2014)

[10] http://openbudgets.eu.