UnifiedViews: Towards ETL Tool for Simple yet Powerfull RDF Data Management*

Tomáš Knap, Petr Škoda, Jakub Klímek, and Martin Nečaský

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

Abstract. We present UnifiedViews, an Extract-Transform-Load (ETL) framework that allows users to define, execute, monitor, debug, schedule, and share ETL data processing tasks, which may employ custom plugins (data processing units, DPUs) created by users. UnifiedViews differs from other ETL frameworks by natively supporting RDF data and ontologies. In this paper, we introduce UnifiedViews and discuss future features of the tool towards simplicity of use for non-RDF experts. We are persuaded that UnifiedViews helps RDF/Linked Data publishers and consumers to address the problem of sustainable RDF data processing; we support such statement by introducing a list of projects and other activities where UnifiedViews is successfully exploited.

Keywords: RDF data processing, ETL, Linked Data

1 Introduction and Basic Concepts of UnifiedViews

The advent of Linked Data [1] accelerates the evolution of the Web into an exponentially growing information space where the unprecedented volume of data offers information consumers a level of information integration that has up to now not been possible.

Suppose a consumer building a data mart integrating information from various RDF and non-RDF sources. There are lots of tools used by the RDF/Linked Data community[1], which may support various phases of the data processing; e.g., a consumer may use *any23*[2] for extraction of non-RDF data and its conversion to RDF data, *Virtuoso*[3] database for storing RDF data and executing SPARQL (Update) queries [2,3], *Silk* [6] for RDF data linkage, or *Cr-batch*[4] for RDF data fusion. Nevertheless, the consumer who is preparing a *data processing task* producing the desired data mart typically has to (1) configure every such tool properly (using a different configuration for every tool), (2) implement a script for downloading and unpacking certain source data, (3) write his own script holding the set of SPARQL Update queries refining the data, (4) implement

* This work was partially supported by a grant from the European Union's 7th Framework Programme number 611358 provided for the project COMSODE

[1] http://semanticweb.org/wiki/Tools

[2] https://any23.apache.org/

[3] http://virtuoso.openlinksw.com/

[4] https://github.com/mifeet/cr-batch

M. Nečaský, J. Pokorný, P. Moravec (Eds.): Dateso 2015, pp. 111–120, CEUR-WS.org/Vol-1343.

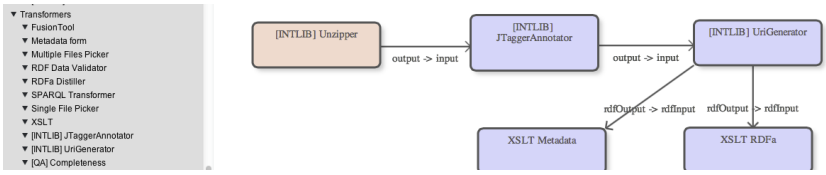

Fig. 1. UnifiedViews framework – Definition of a data processing task

custom transformers which, e.g., enrich processed data with the data in his knowledge base, (5) write his own script executing the tools in the required order, so that every tool has all desired inputs when being launched, (6) prepare a scheduling script, which ensures that the task is executed regularly, and (7) extend his script with notification capabilities, such as sending an email in case of an error during task execution.

Maintenance of such data processing tasks is challenging. Suppose for example that a consumer defines tens of data processing tasks, which should run every week. Further, suppose that certain data processing task does not work as expected. To find the problem, the consumer typically has to browse/query the RDF data outputted by a certain tool; to realise that, he has to manually launch the required tool with the problematic configuration and load the outputted RDF data to the store, such as Virtuoso, supporting browse/query capabilities. Furthermore, when other consumers would like to prepare similar data processing tasks, they cannot share the tools' configurations already prepared by the consumer.

The general problem RDF/Linked Data publishers and consumers are facing is that they have to write most of the logic to define, execute, monitor, schedule, and share the data processing tasks themselves. Furthermore, they do not get any support regarding the debugging of the tasks. To address these problems, we developed UnifiedViews, an Extract-Transform-Load (ETL) framework, where the concept of data processing task is a central concept. Another central concept is the native support for RDF data format and ontologies.

A *data processing task* (or simply task) consists of one or more data processing units. A *data processing unit* (DPU) encapsulates certain business logic needed when processing data (e.g., one DPU may extract data from a SPARQL endpoint or apply a SPARQL query). Every DPU must define its required/optional inputs and produced outputs. UnifiedViews supports an exchange of RDF data between DPUs. Every tool produced by RDF/Linked Data community can be used in UnifiedViews as a DPU, if a simple wrapper is provided[5].

UnifiedViews allows users to define and adjust data processing tasks, using graphical user interface (an excerpt is depicted in Figure 1). Every consumer may also define their custom DPUs, or share DPUs provided by others together with their configurations. DPUs may be drag&dropped on the canvas where the data processing task is constructed. A data flow between two DPUs is denoted as an edge on the canvas (see Figure 1); a label on the edge clarifies which outputs of a DPU are mapped to which

[5] `https://grips.semantic-web.at/display/UDDOC/Creation+of+Plugins`

inputs of another DPU. UnifiedViews natively supports exchange of RDF data between DPUs; apart from that, files and folders may be exchanged between DPUs.

UnifiedViews takes care of task schedulling, a user may configure UnifiedViews to get notifications about errors in the tasks' executions; user may also get daily summaries about the tasks executed. UnifiedViews ensures that DPUs are executed in the proper order, so that all DPUs have proper required inputs when being launched. UnifiedViews provides users with the debugging capabilities – a user may browse and query (using SPARQL query language) the RDF inputs to and RDF outputs from any DPU. UnifiedViews allows users to share DPUs and tasks as needed.

The code of UnifiedViews is available under a combination of GPLv3 and LGPLv3 license[6] at `https://github.com/UnifiedViews`.

2 Related Work

There are plenty of ETL frameworks for preparing tabular data to be loaded to data warehouses, some of them are also opensource[7] – for example Clover ETL (community edition)[8]. In all these frameworks custom DPUs may be created in some way, but the disadvantage of these non-RDF ETL frameworks is that there is no support for RDF data format and ontologies in the framework itself. As a result, these non-RDF ETL frameworks are, e.g., not prepared to suggest ontological terms in DPU configurations, a feature important when preparing SPARQL queries or mappings of the table columns to RDF predicates. Furthermore, these frameworks do not have a native support for exchanging RDF data between DPUs; also the existing DPUs do not support RDF data format, URIs for identifying things according to Linked Data principles. Therefore, further, we discuss the related work in the area of RDF ETL frameworks.

ODCleanStore (Version 1)[9], was the original Linked data management framework, which was used as an inspiration for ODCleanStore (Version 2)[10], the student's project implemented at Charles University in Prague and defended in March 2014 . UnifiedViews is based on ODCleanStore (Version 2). Linked Data Manager (LDM)[11] is a Java based Linked (Open) Data Management suite to schedule and monitor required ETL tasks for web-based Linked Open Data portals and data integration scenarios. LDM was developed by Semantic Web Company in Austria[12]. They currently decided to replace LDM, used by their clients, with UnifiedViews and further continue to maintain UnifiedViews together with Charles University in Prague, the Czech Linked Data company Semantica.cz s.r.o.[13], and Slovak company EEA s.r.o.[14].

[6] `http://www.gnu.org/licenses/gpl.txt`, `http://www.gnu.org/licenses/lgpl.txt`
[7] `http://sourceforge.net/directory/business-enterprise/enterprise/data-warehousing/etl/`
[8] `http://www.cloveretl.com/products/community-edition`
[9] `http://sourceforge.net/projects/odcleanstore/`
[10] `https://github.com/mff-uk/ODCS/`
[11] `https://github.com/lodms/lodms-core`
[12] `http://www.semantic-web.at`
[13] `http://semantica.cz/en/`
[14] `http://eea.sk/`

DERI Pipes[15] is an engine and graphical environment for general Web Data transformations. DERI Pipes supports creation of custom DPUs; however, an adjustment of the core is needed when a new DPU is added, which is not acceptable; in UnifiedViews, it is possible to reload DPUs as the framework is running. DERI Pipes also does not provide any solution for library version clashes; on the other hand, in UnifiedViews, DPUs are loaded as OSGi bundles, thus, it is possible to use two DPUs requiring two different versions of the same dependency (library) and no clashes arise. In DERI pipes, it is not possible to debug inputs and outputs of DPUs.

Linked Data Integration Framework (LDIF)[5] is an open-source Linked Data integration framework that can be used to transform Web data. The framework consists of a predefined set of DPUs, which may be influenced by their configuration; however, new DPUs cannot be easily added[16]. LDIF provides a user interface to monitor results of executed tasks.; however, when compared with UnifiedViews, LDIF does not provide any graphical user interface for defining and scheduling tasks, managing DPUs, browsing and querying inputs from and output to the DPUs, and managing users and their roles in the framework. LDIF also does not provide any possibility to share pipelines/DPUs among users. On the other hand, LDIF provides possibility to run tasks using Hadoop[17].

3 Impact of the UnifiedViews Framework

The goal of the *OpenData.cz initiative*[18] is to extract, transform and publish Czech open data in the form of Linked Data, so that the initiative contributes to the Czech Linked (Open) Data cloud. For this effort, UnifiedViews framework is successfully used since September 2013; so far we published tens of datasets, hundreds of milions of triples; Figure 2 depicts an excerpt of the datasets (blue circles) published with UnifiedViews and the integration of these datasets (links are depicted by blue arrows, pointing from the linking dataset to the linked dataset).

Project INTLIB[19] aims at extracting (1) references between legislation documents, such as decisions and acts, (2) entities (e.g., a citizen, a president) defined by these documents and (3) the rights and obligations of these extracted entities. UnifiedViews is used in INTLIB to extract data from selected sources of legislation documents, convert it to RDF data, and provide it as Linked Data.

COMSODE FP7 project[20] has the goal to create a publication platform for publishing (linked) open data. UnifiedViews is used there as the core tool for converting hundreds of original datasets to RDF/Linked Data.

UnifiedViews framework is being integrated to the stack of tools produced by the *LOD2 project*[21]. As a result, anybody using tools from LOD2 stack, such as Virtuoso

[15] http://pipes.deri.org/
[16] http://ldif.wbsg.de/
[17] http://hadoop.apache.org/
[18] http://opendata.cz
[19] http://www.isvav.cz/projectDetail.do?rowId=TA02010182
[20] http://www.comsode.eu/
[21] http://lod2.eu/

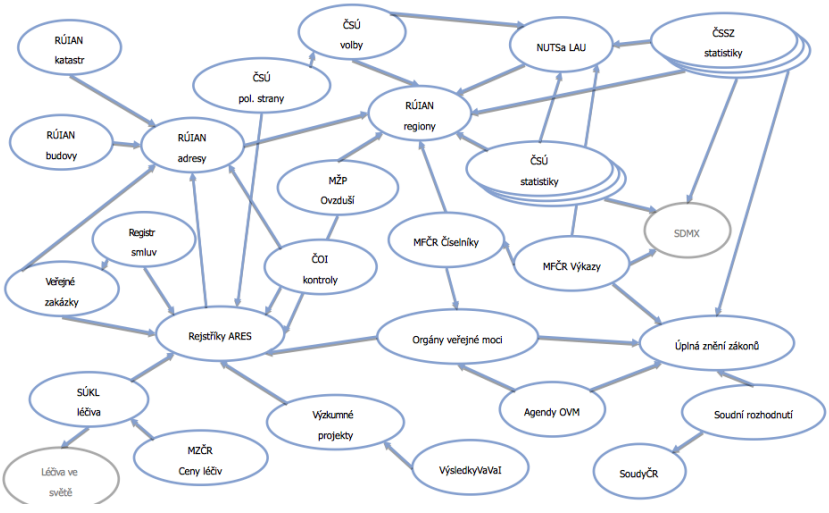

Fig. 2. Datasets published by `opendata.cz` initiative – an excerpt

and Silk, has also the possibility to use UnifiedViews. UnifiedVIews will be also used in the recently starting EU H2020 project called YourDataStories[22].

UnifiedViews framework is intended to be used for commercial purposes by companies Semantica.cz s.r.o., Czech Republic, EEA s.r.o., Slovak Republic, Semantic Web Company, Austria, TenForce, Belgium, to help their customers to prepare and process RDF data.

4 Ongoing and Future Work Towards Simplicity of Use

In this section, we introduce the ongoing and future work on UnifiedViews towards simplicity of use of the tool for non-RDF experts. Each section below describes the planned feature. In the sections below, we are talking about a *task designer* – a person who creates new or adjusts existing data processing tasks. All sections contain motivation, goals to be achieved, and at least outline how the goals will be realised.

4.1 Automated Schema Alignment and Object Linkage

Motivation. Suppose that a data processing task is created to daily extract tabular data provided by Czech Hydrometeorological Institute about the air pollution in various cities of the Czech Republic and publish them as Linked Data. Publishing data as Linked Data involves (1) alignment of the schema used for the published data with well-known schemas (RDF vocabularies) used in the Linked Data community, e.g., in Linking Open Data cloud[23] and, (2) linkage of the published objects with the objects

[22] `https://www.insight-centre.org/content/your-data-stories`
[23] `http://lod-cloud.net/`

already available in the Linked Data space, so that common objects in the datasets have the same identifiers across the datasets. As a result, Linked Data applications work on top of integrated datasets (thanks to (2)), which use common schema elements (thanks to (1)).

If the task designer is a Linked Data expert, he is able to manually integrate the data, e.g., link the RDF representation of the cities introduced in the source tabular data to the generally accepted representation of the cities – identifiers used by the LAU codes dataset[24]; based on that, Linked Data applications may not only show pollution in the particular city, but also, e.g., level of carbon emissions in that city, demographic statistics for the population in the city, number of child inhabitants in that city, etc.

Linked data experts may also ensure that published data is using well-know RDF vocabularies used in Linked Open Data community to publish certain types of data; for example, the task designer may ensure that instead of automatically generated predicate `ex:firstName` holding first names of persons, the predicate `foaf:givenName` is used. Such adjustments of the RDF data related to alignment of the schemas or object linkage are not trivial and cannot be easily done by non-experts.

Goal to be achieved. UnifiedViews should simplify Linked Data publishing for non-RDF experts by:

1. Automatically discovering that certain columns in the processed tabular represent certain types of data (e.g., cities of the Czech Republic) and automatically mapping values in this column to URIs taken from the preferred dataset for the given type of data (e.g., from the dataset with LAU codes). As a result, all datasets use the same identifiers for the same types of data, which realises the data integration and avoids costly and adhoc application integration.
2. Automatically suggest the mappings of the used RDF vocabulary terms (e.g., predicated) to well-known vocabulary terms (e.g., predicates), which increases the understandability of the data and reuse of the data by various applications.

To realise 1), first, it is necessary to identify that certain columns contain certain types of values; such identification is always probabilistic and typically based on the comparison of the name of the column with the list of names of the RDF classes and/or based on matching sample data from the considered column against known codelists, such as list of Czech cities; experiments are needed to decide the particular algorithm for identification of types among input data. Second step to realise 1) is to apply predefined Silk [6] rules for the given identified type of data within the column of the input tabular data. To realise 2), various schema matching techniques has to be experimented [4].

4.2 Hiding Sparql Queries

Motivation. Linked data expert is able to query RDF data using SPARQL query language [2,3], so for an expert it is enough to have one generic DPU, which is able to execute arbitrary SPARQL query on top of processed RDF data. Nevertheless, using

[24] `http://opendata.cz/linked-data`

SPARQL query language for rather typical and simple operations with the data, such as renaming predicates or replacing predicate's value based on a regular expression, may be considered as too heavy-weight and difficult for RDF beginners and as tedious for experienced users.

Goal to be achieved. UnifiedViews should simplify work with SPARQL query language by providing a set of DPUs for executing rather typical and simple operation on top of RDF data; such DPUs may be configured via a configuration dialog, SPARQL query behind is completely hidden from the task designer.

To realise the goal, list of typical operation should be written down and DPUs should be prepared. Discussion with the users – task designers – is crucial to focus on the most typically used operations on top of RDF data.

4.3 Autocompleting Terms from Well-known Vocabularies

Motivation. In many cases, e.g., when aligning vocabulary terms as described in Section 4.1 or when configuring DPU hiding complexity of SPARQL query language as described in Section 4.2, task designer has to define certain vocabulary terms. Since the number of well-known vocabularies is quite high, task designer may easily use wrong vocabulary term or misspell the term.

Goal to be achieved. As the task designer is configuring DPUs, UnifiedViews should suggest and autocomplete vocabulary terms from well-known Linked Data vocabularies. Task designer should be not only provided with the suggested term, but also with the description of the term, its formal definition, its recommended usage etc.

To realise this goal, data processing tasks should be prepared to populate RDF database regularly with the well-known Linked Data vocabularies – such knowledge base is then used for suggesting and autocompleting the vocabulary terms. Selected components of the DPUs' configurations, e.g., text fields, should by design support suggesting of terms from well known vocabularies – so any DPU developer may use such vocabulary autocomplete aware text field when defining configuration dialog for his DPU.

4.4 Sustainable RDF Data Processing

Motivation. As the task designer updates the task, the interconnections among and configurations of the DPUs comprising that task are adjusted. As a result, task designer may introduce errors in the definition of the task yielding in erroneous or no data produced by the task.

Goal to be achieved. UnifiedViews should address the problem of sustainable RDF data processing by allowing task designer to define for each DPU a set of SPARQL queries, which tests that the output data produced by the given DPU satisfies certain conditions. Such set of SPARQL queries for testing data outputted by the DPU plays

similar role as standard JUnit tests – to test that any change to the DPU configuration did not change the produced data of that DPU in an unexpected way. Task designer should be supported with the autocomplete feature (described in 4.2) as he is specifying the SPARQL unit tests.

To realise this goal, every DPU detail should be extended with the possibility to define set of SPARQL ASK queries to realise unit testing.

4.5 Wizards for Simple Definition of Data Processing Tasks

Motivation. Defining data processing tasks typically requires detailed knowledge of the DPUs that are available in the deployed UnifiedViews instance; task designer has to know which DPUs are suitable for the task at his hand, how it should be configured and interconnected with other DPUs.

Goal to be achieved. It should be possible to define simple tasks without the knowledge of the DPUs, its configurations. UnifiedViews will contain so-called wizards, which provides task designers step by step guides for defining new data processing tasks – at least for typical types of data processing tasks, e.g, extracting tabular data and publishing it as Linked Data, or extracting data from relational databases and publishing it as Linked Data.

To realise the goal, list of typical types of data processing tasks should be written down and wizards should be prepared for such tasks. Discussion with the users – task designers – is crucial to focus on the most typical types of tasks. The idea of incorporating wizards to existing UnifiedViews frontend is as follows: when a task designer creates new pipeline, he may either manually define the task or start the wizard which will guide him through the process of task preparation; task designer may then manually finetune the definition of the task.

4.6 Assessing Quality of Produced Data, Recommendation of Cleansing DPUs

Motivation. As the goal of a task designer is to produce high quality data, the task designer should be informed about any problems in the data, e.g., w.r.t. syntactic/semantic accuracy of the produced Linked Data or completeness of the published dataset. Furthermore, if such problems may be corrected, they should be corrected.

Goal to be achieved. UnifiedViews should provide a set of DPUs assessing the quality of the produced data and set of DPUs being able to cleanse the problems in the data. UnifiedViews should also automatically recommend cleansing DPUs for data processing tasks based on the problems revealed in the data.

To realise the goal, list of quality assessment and cleansing DPUs should be implemented, being inspired by the list of data quality dimensions and metrics relevant for Linked Data [7]. The recommendation of cleansing DPUs should be based on the types of quality assessment DPUs which reported problems.

4.7 Evolution of DPUs

Motivation. DPUs may evolve as the time goes, different tasks use different versions of the same DPU. When the version of the DPU is updated, configuration used in the tasks must be also updated without the needed to reconfigure the DPU by the task designer.

Goal to be achieved. UnifiedViews must be able to cope with the changing versions of the DPUs; each new version of the DPU may bring changes to the DPU's configuration. UnifiedViews must be able to automatically convert outdated configuration, so that it may be used in the latest version of the DPU.

To realise the goal, interface of the DPUs should be extended, so that DPU developers may provide a migration method converting previous configuration to the current DPU configuration. As a result, if outdated version of the configuration is encountered and should be updated to the correct version, a sequence of these migration methods may be automatically executed by UnifiedViews (if these methods are properly provided by the DPU developer).

5 Conclusions

We presented UnifiedViews, an ETL framework with a native support for processing RDF data. The framework allows to define, execute, monitor, debug, schedule, and share data processing tasks. UnifiedViews also allows users to create custom plugins - data processing units.

We discussed future intended features of the tool w.r.t simplicity of use of the tool for non-RDF experts or those not familiar with all Linked Data vocabularies, datasets etc. For each intended feature we discussed its motivation, goals and its realisation.

We are persuaded that UnifiedViews is a matured tool, which addresses the major problem of RDF/Linked Data consumers – the problem of sustainable RDF data processing; we support such statement by introducing a list of projects where UnifiedViews is successfully used and mention two commercial exploitations of the tool.

References

1. C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1 – 22, 2009.
2. S. H. Garlik, A. Seaborne, and E. Prud'hommeaux. SPARQL 1.1 Query Language. W3C Recommendation, 2013. `http://www.w3.org/TR/2013/REC-sparql11-query-20130321/`, Retrieved 20/03/2014.
3. P. Gearon, A. Passant, and A. Polleres. SPARQL 1.1 Update. Technical report, W3C, 2013. Published online on March 21st, 2013 at `http://www.w3.org/TR/2013/REC-sparql11-update-20130321/`, Retrieved 20/03/2014.
4. E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, Dec. 2001.
5. A. Schultz, A. Matteini, R. Isele, C. Bizer, and C. Becker. LDIF : Linked Data Integration Framework. In *Proceedings of the Second International Workshop on Consuming Linked Data (COLD)*, Bonn, Germany, 2011. CEUR-WS.org.

6. J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Silk - A Link Discovery Framework for the Web of Data. In *Proceedings of the WWW2009 Workshop on Linked Data on the Web (LDOW)*, Madrid, Spain, 2009. CEUR-WS.org.
7. A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, and S. Auer. Quality assessment for linked data: A survey. *Semantic Web Journal*, 2015.