Not Everybody Speaks RDF: From Knowledge Graph Construction to Knowledge Conversion between Different Data Representations

Mario Scrocca, Alessio Carenini, Marco Grassi, Marco Comerio, and Irene Celino
Cefriel – Politecnico di Milano, Milano, Italy
E-mails: mario.scrocca@cefriel.com, marco.grassi@cefriel.com, marco.grassi@cefriel.com, marco.comerio@cefriel.com, irene.celino@cefriel.com

Abstract. Knowledge representation in RDF guarantees shared semantics and enables interoperability in data exchanges. Various approaches have been proposed for RDF knowledge graph construction, with declarative mapping languages emerging as the most reliable and reproducible solutions. However, not all information systems can understand and process data encoded as RDF. In these scenarios, to guarantee seamless communication there is a need for a further conversion of RDF graphs to one or more target data formats and models. Existing solutions for the declarative lifting of data to RDF are not able to effectively support knowledge conversion towards a generic output. Based on an examination of existing mapping languages and processors for RDF knowledge graph construction, we define a reference workflow supporting a knowledge conversion process between different data representations. The proposed workflow is validated by the `mapping-template` tool, an open-source implementation based on a popular template engine. The template-based mapping language enables the definition of mappings without requiring prior knowledge of RDF and provides flexibility for the target output. The tool is evaluated qualitatively, considering common challenges in the declarative specification of mappings, and quantitatively, considering performance and scalability. This paper extends a previous version of this work by integrating a discussion of the proposed workflow considering the analysed state-of-the-art for knowledge graph construction, introducing the tool's direct support for the execution of RML mapping rules, and describing a more comprehensive qualitative and quantitative evaluation, also considering the results obtained by participating in the Knowledge Graph Construction Challenge 2024.

Keywords: Knowledge Graph Construction, Knowledge Conversion, Declarative Mappings, Mapping Languages

1. Introduction

The challenge of data interoperability can be addressed by representing knowledge according to shared semantics in RDF graphs. In recent years, several approaches have been proposed for lifting, i.e., the generation of RDF graphs from heterogeneous data sources. Declarative mapping languages emerged, in contrast with ad-hoc procedures, as a suitable solution to improve the maintenance and reproducibility of the mapping process. Different requirements led to the definition of multiple declarative mapping languages and the implementation of several mapping processors interpreting and executing them [62]. In this context, the recent and ongoing research activities mainly focus on (i) the extension of declarative mapping languages to support new mapping challenges and requirements for RDF knowledge graph construction [38], and (ii) the improvement of performances for mapping processors against the

identified benchmarks [4, 14]. However, not all information systems are able to process information represented as RDF. In these cases, a knowledge conversion process from the knowledge graph towards a generic output is required [8, 54]. While the available declarative mapping languages for RDF graph generation can not directly support such a process, they propose key contributions to address this problem. Based on the analysis of the literature reported in Section 2, we propose a workflow for a generic knowledge conversion process, enabling not only transformations to/from RDF but also conversions among data formats and data models not bound to Semantic Web technologies. The workflow aims to identify the key characteristics of existing declarative mapping languages and processors, propose an approach to decouple the steps involved, and overcome the limitation of generating an RDF output.

We implemented the `mapping-template` tool[1] as an open-source software component that leverages the Apache Velocity[2] template engine to enact and validate the defined workflow. The adoption of a template-based language enables the definition of mapping rules by users not familiar with RDF and the possibility of targeting a generic output format and schema. Section 3 describes the proposed workflow and how it can be used as a framework to discuss and compare existing contributions for declarative knowledge graph construction. Section 4 presents the tool and how it implements the workflow. In Section 5, we discuss example templates demonstrating how the tool can address the requirements of RDF graph generation and cover additional scenarios. In Section 6, we perform a quantitative evaluation of the tool on an RDF materialisation task, and we compare it with other existing engines. Configurations and results of both evaluations are made available online to promote reproducibility[3]. Finally, in Section 7, we describe the tool's adoption for different use cases, while in Section 8, we draw conclusions and discuss future work.

2. Preliminaries and related work

The W3C Knowledge Graph Construction Community Group[4] involves researchers and practitioners aiming at investigating the problem of RDF graph construction, the different approaches and solutions proposed, and the potential extension of the R2RML W3C recommendation [21] beyond relational databases [38]. This section introduces the main terminology adopted in the paper and the state-of-the-art declarative mapping languages and processors.

2.1. Terminology

The RDF knowledge graph construction process targets the techniques and tools that can process (semi-)structured heterogeneous data sources to generate an RDF representation of the input data. In this paper, we adopt the definitions proposed by Van Assche et al. [62].

Schema mappings define a set of rules according to a *mapping language* to transform a *source schema* in a *target schema*. We identify as schema both the data format (e.g., RDF, JSON) and data model (e.g., ontology, JSON Schema) adopted. A *schema transformation* applies the schema mappings to an input data source represented through the source schema and generates output data according to the target schema. A *data transformation* applies a custom logic (e.g., functions) to process data values (e.g., changes in string capitalization). This paper considers a generic mapping process (also referred to as a *conversion process*) that may require both schema and data transformations. In particular, we do not restrict the data format of the target schema to RDF. As a final remark, it is essential to highlight that we focus on approaches for the *materialisation* of the output, i.e., storing the result of the mapping process.

[1] https://github.com/cefriel/mapping-template
[2] https://velocity.apache.org/
[3] https://github.com/cefriel/mapping-template-eval
[4] https://www.w3.org/community/kg-construct/

2.2. Declarative mapping languages and processors

The process of RDF graph generation from (semi-)structured data encompasses three main approaches [62]:

- *Hard-coded procedures*. The definition of these procedures does not require learning a mapping language; however, they are difficult to maintain since every modification requires a new development for its implementation. Moreover, they are not reusable, and the user should completely handle the optimisation of the mapping process.
- *Format-specific mappings*. The mapping language and processor are optimised for the specific format considered. However, the definition and execution of mappings for data sources in different formats require learning and maintaining multiple solutions. Moreover, it is not possible to integrate data sources with different formats.
- *Declarative mappings*: Propose a single solution for the declarative definition of mappings from different data sources. The mappings are reusable and decoupled from the processor executing them. Indeed, other processors may be used to execute the mappings if they conform to the same adopted mapping language.

The declarative mapping languages can be classified as (i) dedicated languages based on R2RML syntax (R2RML [21], RML [28], D2RML [16], KR2RML [58], R2RML-F [24], xR2RML [50]), (ii) dedicated languages with custom syntax (Helio Mapping Language [17], D-REPR [68]), (iii) repurposed languages based on constraint languages (ShExML [30] extending the ShEx syntax), (iv) repurposed languages based on SPARQL syntax: XSPARQL [1], Facade-X [7], SPARQL-Generate [47]. Each mapping language is implemented by at least one mapping processor able to execute a set of mappings fulfilling its specification.

Different solutions address specific requirements and have their advantages and disadvantages. For this reason, it is crucial to offer comparison workflows for the user to choose and promote initiatives to reconcile the proposed solutions.

The paper from [39] discusses an ontological approach for representing declarative mapping languages generating an RDF output. The paper defines the *Conceptual Mapping* ontology to cover both features offered by state-of-the-art mapping specifications and a set of mapping challenges collected by members of the knowledge graph construction community[5]. The high-level concepts identified by the ontology are considered in this work to support the workflow definition. Moreover, we use the ontological requirements[6] for the evaluation of the mapping capabilities of our tool.

The literature review by Van Assche et al. [62] provides an overview of mapping languages and available mapping processors for RDF graph generation. The review identifies a set of characteristics for both schema transformation and data transformation. It compares approaches for materialisation and virtualisation of the RDF knowledge graph based on declarative mapping languages. RML emerges as the language providing a wider number of compatible mapping processors. We considered the reviewed tools and the extracted characteristics to define a generalised conversion process.

Several efforts in the literature focus on the evolution of mapping languages to cover new requirements. The integration of the Function Ontology (FnO) with RML is proposed in [23] to enable the declaration of data transformations in the mappings. The authors in [61] describe the extension of the RML Logical Source to support Web APIs and streams. Moreover, they introduce the RML Logical Target to define the characteristics of the expected knowledge graph generated. An RML extension to directly support the mapping of in-memory data structures is discussed in [22]. RML Views [5] are proposed to facilitate the mapping of tabular data sources. The RML-star [26] extension for the RML language enables the definition of declarative mappings to generate RDF-star [33] triples, while the RML-CC[7] extension allows generating RDF Collections and Containers. Finally, the RML Fields [25] and RML Logical Views [67] proposals define an approach to handle mapping rules for complex nested data structures. The new RML ontology [38] incorporates several of the discussed extensions and is designed as a modular solution: RML-Core for schema transformations, RML-IO for the logical source and target, RML-CC for collections and containers, RML-FNML for data transformations, RML-star for RDF-star.

[5] https://kg-construct.github.io/workshop/2021/challenges.html
[6] https://github.com/oeg-upm/Conceptual-Mapping/tree/main/requirements
[7] https://github.com/kg-construct/rml-cc

Another set of contributions targets the performance and scalability of the mapping process. The GTFS-Madrid-Bench [13] defines a benchmark to test the scalability of solutions for knowledge graph construction. The KROWN benchmark [60] extends the GTFS-Madrid-Bench, providing an extended set of test cases for RDF graph materialization and has been used for the challenge of the Knowledge Graph Construction workshop [14, 15]. A qualitative and quantitative comparison of different (R2)RML processors is provided by Arenas-Guerrero et al. in [4]. Optimisations for the processing of data transformations defined within the mappings are proposed by FunMap [42] and Dragoman [41]. The usage of support data structures to speed up the mapping execution is proposed by SDM-RDFizer [37]. The optimisation of the execution of join operations defined within the mapping is also investigated considering the removal of self-joins [6] and the replacement with reference conditions [66]. The concurrent processing of mapping rules is investigated by Chimera [55] and Morph-KGC processor [6]. RMLStreamer [52] and RPT/SANSA [59] leverage respectively Apache Flink and Apache Spark to optimise the execution of RML mappings rules. Finally, FlexRML [29] is written using C++ and introduces several solutions (e.g., size estimation algorithm) to minimise memory usage and address knowledge graph construction on resource-constrained devices.

2.3. Beyond RDF knowledge graph construction

Declarative mapping languages for knowledge graph construction assume RDF triples as the expected output of the mapping process. However, a lowering procedure targeting heterogeneous data formats and models is often needed to process the knowledge represented in the RDF graph. The position paper from Bennara et al. [8] discusses how knowledge graphs can foster the interoperability of web services on the Web of Things (WoT) and claims the need for appropriate lowering procedures enabling communication among different devices. In previous work, we described how semantic technologies can enhance data exchanges between different data standards within a multi-stakeholder environment, and we demonstrated it considering a use case from the transportation domain [54]. We claimed the need to lower RDF data to heterogeneous data formats to achieve this goal, and we proposed a solution based on the Apache Velocity language. The presented `mapping-template` tool represents a generalisation of the proposed approach for defining mappings between different data representations. XSPARQL offers a solution for the definition of lifting and lowering mappings to/from RDF but is limited to the XML format [9]. The SPARQL Template Transformation Language (STTL) [19] provides a SPARQL-based solution for the definition of lowering mappings from RDF data to heterogeneous data sources.

The advantages of applying a single approach for the definition of lifting and lowering mappings emerge from the literature mentioned. The possibility of reusing declarative lifting mappings (e.g., RML mappings) for both directions is also discussed but with limited results [2, 34] due to the difficulties of inverting uniquely and unambiguously the assertions defined for the lifting process.

3. A workflow for declarative knowledge conversion

Starting from the analysis of available languages and tools for declarative RDF knowledge graph construction, we designed a workflow to generalise the declarative conversion process between different data representations. We consider a *mapping scenario* where data from a data source, represented according to an input data format and data model, should be converted to an output data format and output data model and stored in a data sink. The mapping scenario may involve integrating additional data sources to generate the output and data transformations to be applied during the process. The workflow defines the building blocks for a generic *declarative mapping language* and the corresponding block for a *mapping process* executing the mappings.

Figure 1 describes the complete workflow proposed through a diagram that identifies and decouples the different steps. The mapping process can be described as an Extract-Transform-Load (ETL) process [4] defined through a declarative mapping language. The workflow is designed to synthesise the state-of-the-art solutions for RDF knowledge graph construction and overcome the limitation of generating only RDF outputs.

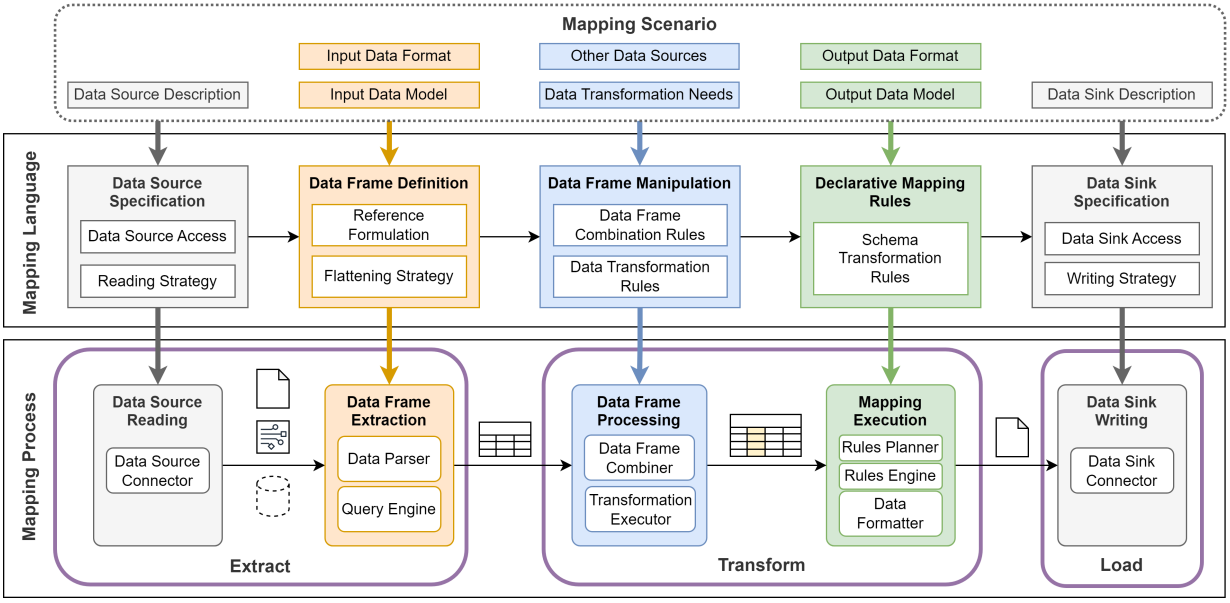

Fig. 1. Overview of the proposed workflow for the generalisation of the declarative mapping process.

3.1. Extract

The *Data Source Specification* defines how to access and retrieve the data to be processed during the mapping process. Different configurations may be needed according to the data source, for example, whether it is local or remote, a dataset or a data service (stream or connection to a database). The *Data Source Access* indicates the location (e.g., URL) to access the data source, the protocol to access the resource, and the security mechanisms restricting the access. The *Reading Strategy* indicates the type of interaction expected, e.g., push versus pull mechanism, synchronous versus asynchronous, batch versus stream. The RML Logical Source[8] can be used to define a *Data Source Specification* declaratively. The implementation of the *Data Source Reading* functionality requires the selection of *Data Source Connector*(s) supporting the selected data source(s) and the expected interaction in reading data from them.

The parsing and extraction process from heterogeneous data sources can be generalised considering the concept of *data frame*, i.e. a two-dimensional data structure made of rows and columns. The selection of a data frame as the input data structure to apply the mapping rules is inherited by declarative languages based on the R2RML syntax. Indeed, tabular data sources already fit a data frame, and query languages (like SPARQL[9]) usually define their result set in a tabular format. To enable the definition of declarative rules over hierarchical data formats (e.g., JSON and XML), several declarative mapping languages relied on the definition of an intermediate representation based on a tabular data structure. For example, the authors in [58] considered the Nested Relational Model (NRM) an intermediate abstraction. Differently from NRM, we define the identification of a complete flattening strategy as a requirement for a generic conversion process, e.g., not allowing nested tables or objects. Indeed, NRM can also be normalised in the general case [48]. Similarly, the RML [28] specification implicitly defines a flattening strategy for hierarchical data sources through the `rml:iterator` and `rml:reference` operators. The approaches proposed by RML Fields [25] result in a more explicit identification of a tabular structure for complex nested data sources in RML. In this paper, we claim that the explicit declarative definition of the data frame(s) as the intermediate abstraction between the *Extract* and *Transform* steps of the mapping process can facilitate the definition and optimisation of schema and data transformations.

[8] https://w3id.org/rml/io/spec
[9] https://www.w3.org/TR/rdf-sparql-query/

The *Data Frame Definition* defines a proper *Reference Formulation* (e.g., SQL, XQuery, etc.) to express a *Flattening Strategy* that extracts one or more data frames from the input data source according to its data format and model. This workflow mainly focuses on mapping (semi-) structured data sources; however, assuming a specific procedure to define a data frame from unstructured data sources (e.g., text in a PDF), the overall workflow may also be applied to these data sources.

The implementation of the *Data Frame Extraction* functionality requires the selection of two components: a *Data Parser* responsible for parsing data received from the data source according to their specific format (e.g., CSV/XML/JSON), and a *Query Engine* capable of extracting the data frame from the parsed data and according to the *Data Frame Definition*. In this context, we identify as *Query Engine* a generic component that can interpret the *Flattening Strategy* defined to extract the data frame from the parsed data. Examples of a *Query Engine* are a SQL query engine in the case of a relational database, a SPARQL query engine in the case of RDF data, or a more generic library extracting a data frame from a JSON object.

3.2. Transform

The specification of *Data Frame Manipulation* considers both the need for combining the extracted data frame with *Other Data Sources* and the *Data Transformation Needs*.

The *Data Frame Combination Rules* specify how multiple data frames extracted from different (or the same) data source(s) should be combined to define the combined frame that mapping rules will target. Other combination rules can be adopted according to relational algebra operations (e.g., the union of data frames, cartesian product, or join operation).

The *Data Transformation Rules* specify how to manipulate a set of data in the data frame, e.g., all the values for a column in the data frame. Generally, a data transformation rule is an arbitrary function processing a portion of the data frame. In some cases, data transformation rules may be restricted to functional computations, i.e., without a state or side effects. However, specific scenarios may require more generic computations (e.g., transformation of the data frame should keep track of values already processed). It is always possible to decouple data transformations from the *Declarative Mapping Rules* definition [41]. Using the FnO ontology [23] and RML-FNML, data transformation rules can be declaratively described in RML and associated with a specific implementation the processor can execute.

The implementation of the *Data Frame Manipulation* functionality requires the selection of two components: a *Data Frame Combiner* capable of executing the combination of one or more data frames according to the rules specified, and a *Transformation Executor* capable of applying the data transformation logic required to the data frame. Data frame combination rules can be avoided if the combination is applied during the data frame extraction (e.g., performing the extraction with a join query over multiple input data sources [5]).

The specification of *Declarative Mapping Rules* is based on *Schema Transformation Rules* that define how to process the data frame(s) to obtain the target data format and model. We believe that the work done by the community in defining fully declarative mapping languages based on the R2RML approach has the drawback of focusing on the output of RDF triples via *TriplesMap*. This approach requires the introduction of several extensions of the syntax to enable specific types of outputs (cf. Section 2). Moreover, it can be verbose and counter-intuitive for the final user, e.g., if constant RDF triples should be materialised or if multiple triples should be generated for a single input. Solutions like YARRRML [35] facilitate the definition of the mappings, however, they still follow a *TriplesMap*-based approach. The languages based on SPARQL benefit from the flexibility provided by the CONSTRUCT clause to facilitate the user's definition of the expected target. However, they are also bound to the generation of an RDF-based output. Finally, it should be noted that several mapping languages for RDF generation are based on Semantic Web specifications (e.g., RDF, SPARQL, ShEx). A syntax for the specification of declarative mapping rules towards a generic output can support additional conversion requirements.

Implementing the *Mapping Execution* functionality requires the identification of a *Rule Engine* component that can access data from the extracted and manipulated data frames and produce the output according to the specified declarative mappings. The *Mapping Execution* can rely on additional components. A *Mapping Rule Planner* determines and optimises the order in which mapping rules should be executed. A *Data Formatter* validates the

Table 1
Coverage of the proposed workflow by mapping language specifications.

Declarative mapping language specification	**Data Source Specification**	**Data Frame Definition**	**Data Frame Manipulation**	**Declarative Mapping Rules**	**Data Sink Specification**
R2RML [21]	RDB	X		RDF	
R2RML-F [24]			X		
RML [28]	(Relational) Database File(s)	X	X	RDF	
Source/Target RML [61]	Web APIs and Streams				X
RML-CC[7]				RDF Collections/Containers	
RML+FnO [23]			X		
RML-star [26]				RDF-star	
RML Fields [25]		X			
RML Logical Views [67]		X	X		
RML Reference Condition [66]		X	X		
RML Views [4]	X	X			
RML In-memory [22]	X				
Conceptual Mapping ontology [39]	X	X	X	RDF	

produced output according to a specific data format and can obtain different representations of the same output (e.g., pretty-printing, different RDF serialisations).

3.3. Load

The *Data Sink Specification* defines how to connect (*Data Sink Access*) and send (*Writing Strategy*) the data obtained as a result of the mapping process. As for the *Data Source Access*, different configurations may be specified. Furthermore, the definition of an incremental writing strategy may be required to determine how the output data should be partitioned for writing [55]. Finally, the result of the mapping process may be split considering different data sinks. The RML Logical Target[8] supports a declarative *Data Sink Specification* for an output RDF graph.

The implementation of the *Data Sink Writing* functionality requires the selection of *Data Sink Connector*(s), supporting the target data source and the expected interaction in writing data.

3.4. Related Work Discussion

This section discusses how the workflow proposed can be used to support the comparison of existing languages and tools for knowledge graph construction introduced in Section 2. Considering mapping languages, Table 1 provides an overview of which aspects of the workflow are covered by different works in the literature on mapping languages based on the R2RML syntax. The declarative specification of heterogeneous data sources (*Data source specification*) is supported to enable several use cases through different extensions. The adoption of techniques for the declarative definition of the flattening strategy (*Data frame definition*) is investigated to support use cases with nested and multiple data sources. The support for the specification of data transformation functions and the combination of data extracted from different data sources (e.g., *join* conditions) is analysed in the literature (*Data frame definition*). All the proposed works limit the definition of the output (*Declarative mapping rules*) as RDF(-star) triples. The work presented in [61] investigates the declarative specification of the target data sink (*Data sink specification*).

Considering mapping processors, the available engines[10] cover all the steps required to support the functionalities associated with the target mapping language. However, the processors provide integrated solutions for the execution

[10]https://github.com/kg-construct/awesome-kgc-tools

of the mappings that are usually difficult to compare without analysing their source code. The proposed decoupling in building blocks enables a comparison of optimisations and enhancements discussed in the literature and implemented by different processors. As discussed in previous work [55], the selection of libraries for implementing the *Data Frame Extraction* functionality may heavily impact the usage of computational resources and the execution time. The implementation of this functionality should avoid multiple parsing procedures on the same data source (e.g., if it contributes to the definition of multiple data frames) and exploit caching mechanisms to avoid the execution of the same query multiple times. Also, the adoption of incremental writing strategies in the *Data Sink Writing* phase may affect the overall performance of the mapping process. The authors in [41] discuss how the decoupling of the *Data Frame Processing* phase from the *Mapping Execution* can benefit the performances of the mapping processors. Iglesias et al. in [37] demonstrate the benefits of creating support data structures to optimise the execution of join operations and minimise the access to data frames in the *Data Frame Processing*. Arenas et al. in [6] discuss how a smart *Mapping Rule Planner* can be exploited to identify dependencies among different mapping rules and exploit concurrency in their processing. Similarly, another work by Iglesias et al. [36] proposes a strategy to efficiently plan and schedule the mapping rules by leveraging greedy algorithms.

We believe that the proposed workflow can also support the development of more decoupled mapping processors, providing flexibility for the user in introducing optimizations based on the specific requirements of the mapping scenario being addressed. In this direction, the prototype implemented for the RML Logical View specification [67] investigates the performance benefits of applying the proposed specification to existing RML processors. Indeed, the explicit definition of *views* in RML results in an approach similar to the one discussed for our workflow considering data frames. RML Logical Views enable a decoupling between the definition of tabular views and the mapping rules, and this results in the possibility of optimising data access for nested data sources during the execution or ahead of the execution by processing the input data sources accordingly. As an example, the optimization of the mapping execution can be improved if a minimal set of data is extracted as data frames from the data sources taking into account the mapping rules to be processed. Moreover, multiple join conditions may be optimised if applied directly to the data frame, potentially also leveraging indexes and/or support data structures for their execution.

4. The `mapping-template` tool

The `mapping-template`[1] tool provides a solution for implementing generic data and schema transformations and is designed according to the workflow discussed in Section 3. The `mapping-template` is released open-source under an Apache License 2.0. It can be downloaded from GitHub[11] for standalone usage and is also available on Maven Central as a library.

The `mapping-template` tool is based on the Apache Velocity Engine[2], a template engine to dynamically generate a generic output according to a predefined structure. The Velocity Template Language[12] (VTL) allows for the definition of a template that is composed of (i) *static* elements that are added to the output as constant strings, (ii) *dynamic* variables that are bound at runtime to specific values, (iii) *directives* that can be used to define a specific logic (e.g., *if/else*). A typical use case for templates is rendering web pages according to the data dynamically retrieved by a user. The `mapping-template` tool extends the VTL syntax for the definition of a Mapping Template Language[13] (MTL) to specify mapping rules between different data representations. The definition of mapping rules as templates trades some aspects of a fully declarative approach, but provides flexibility in the generated output and facilitates the definition of mapping rules by users unfamiliar with RDF as discussed in Section 5.

The *Data Source Reading* and *Data Sink Writing* functionalities are partially supported via the MTL to enable the execution via CLI. However, we opted for decoupling these steps to avoid the need to import several external libraries into the tool. We took this decision assuming that the `mapping-template` tool may easily be integrated within existing ETL tools, providing several production-ready data connectors out-of-the-box. In this direction, we

[11] https://github.com/cefriel/mapping-template/releases
[12] https://velocity.apache.org/engine/2.0/vtl-reference.html
[13] https://github.com/cefriel/mapping-template/wiki/Mapping-Template-Language-(MTL)

integrated the tool within the Chimera[14] framework to support the declarative definition of composable semantic transformation pipelines leveraging MTL and the Apache Camel integration framework [31].

The *Data Frame Extraction* process is standardized by a `Reader` interface enabling the extraction of a data frame from a generic input data source. The selection of a specific `Reader` implementation depends on the *Reference Formulation* of the data. The *Flattening Strategy* should be provided in the template as a parameter of the `getDataFrame` method exposed by the *Reader*. The tool currently implements the *Data Frame Extraction* for CSV, XML via XQuery [53], JSON via JsonPath [32], RDF via SPARQL and relational databases via SQL. Specific `Readers` are implemented for each supported *Reference Formulation*, requiring different configurations to extract the data frame. For example, the XQuery `Reader` can process queries over XML inputs to extract a data frame. The user should specify in the definition of the template the proper query to extract the required data frames. Multiple data frames can be defined in a single template from different data sources exploiting different `Readers`. Additional input data formats or different *Reference Formulation* for the formats already supported can be integrated by providing a dedicated implementation of the `Reader` interface.

Once a data frame has been obtained, the *Data Frame Manipulation* and the *Declarative Mapping Rules* can be defined by leveraging the Velocity Template Language. The usage of custom Java functions in the template for data transformation is possible if a suitable implementation is provided in the tool configuration. Functions can be applied to the data frame or directly during the processing of the declarative mapping rules. A set of commonly used functions is made available by default. The template language provides direct access to the data frames and gives the user complete control over the definition of their processing. On the one hand, the user can access the different data frames using the VTL directives (e.g., `foreach`). On the other hand, the template-based approach allows for an unconstrained textual output, not limited to the production of RDF. The access to data frames can be optimised by the user defining the mapping template considering the specific mapping scenario. For example, multiple accesses to a data frame can be optimised by merging different rules accessing the same data frame to generate different outputs. Moreover, the tool makes available a set of functions to define and exploit support data structures for the optimisation of join operations between different data frames. Finally, the `mapping-template` tool provides formatting and validation capabilities for specific output formats, namely for XML, JSON and different RDF serialisations. The tool can be easily extended by implementing the *Formatter* interface to process additional data formats generated as an output of the template.

The `mapping-template` tool also supports the execution of RML mappings[15] encoded using the new RML ontology [38]. We introduced this feature to guarantee compatibility with a fully declarative approach for knowledge graph construction and to enable a better evaluation of the proposed workflow and tool with respect to existing RML mapping processors. The support for RML mappings is added by applying the `mapping-template` tool in two steps, as shown in Figure 2.

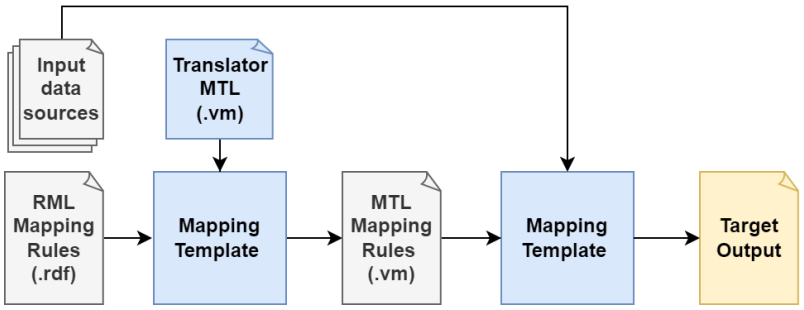

Fig. 2. *Implemented approach for supporting the execution of RML mapping rules.*

[14]https://github.com/cefriel/chimera

[15]The feature is currently implemented in a dedicated branch and will be made available on Maven Central in the next major release of the tool https://github.com/cefriel/mapping-template/tree/feat-rml-compiler/rml

In the first step, the RML mapping rules are processed via a specific template[16] defined to process the input as RDF and generate a corresponding template using MTL. A specific query, adapted from the SDM-RDFizer implementation[17], is used to extract an appropriate data frame accessing the RML mapping rules. The data frame is then processed to generate the appropriate directives using MTL. In the second step, the generated MTL template is processed by applying the defined rules to the input data sources to generate the target output. By default, the RDF triples are serialised as *nquads*, but an appropriate *Formatter* may be specified to obtain the output according to a different RDF serialization. The execution of RML via the `mapping-template` tool can be performed transparently in a single invocation, by utilizing a dedicated option and providing the RDF serialization of the mapping rules, or in two decoupled steps. For this second option, it is sufficient to invoke the `mapping-template` tool twice since the execution of the RML-to-MTL template can be performed as it would be done for any other MTL template having as input an RDF dataset. This case allows checking and potentially optimising the MTL template generated from RML before executing it.

5. Qualitative Evaluation

This section discusses a qualitative evaluation of the `mapping-template` tool considering the requirements for declarative mapping languages for RDF knowledge graph construction identified in [39] and the RML test cases[18].

The list of requirements[19] comprises the set of features made available by different declarative mapping languages and the challenges identified by the community. Each requirement is associated with an identifier of type `cm-r*`[20]. In the evaluation, we discuss example mapping templates targeting different mapping scenarios and how they demonstrate the fulfilment of requirements. For each template, the identifiers of the related requirements are mentioned. The complete set of generated templates can be found in the tool repository[21] together with instructions on how to run them and the generated output. Table 2 summarises the qualitative evaluation, also referencing the corresponding portion of the workflow.

The coverage of basic requirements is demonstrated through the definition of templates for the examples available in the RML documentation[22]. These templates (`rml-csv`, `rml-xml`, `rml-json`, `r2rml`) consider respectively a CSV, XML, JSON and SQL data source as input (`cm-r2`), define the data to be processed for each data source (`cm-r10`), specify how the extracted data should be dynamically converted to RDF (`cm-r8`, `cm-r11`, `cm-r12`). In Figure 3, the snippet (a) shows the mappings of the XML example for our tool, and the snippet (b) the corresponding ones in RML[23]. A mapping template defines the extraction of a data frame using the correct *Reader* associated with the considered data format and *Reference Formulation*. The XMLReader in the example adopts XQuery to define a *Flattening Strategy*. The extracted data frame can then be iterated using the template language VTL to generate the same output RDF triples of the corresponding RML mappings.

The template language does not constrain the generated output, thus facilitating the definition of rules for producing valid RDF triples also considering datatypes, language tags, blank nodes and named graphs (from `cm-r16` to `cm-r21`). The example (`csv-multiple-values`) addresses the mapping challenge to dynamically generate language tags (`C1` and `cm-r12`). The same approach can be applied to generate datatype tags dynamically. Furthermore, this example shows the definition of a custom data transformation function directly applied to the data frame. The column `title` in the input data contains both a string and the associated language tag. A function is used to split the information into two different columns.

[16] RML-to-MTL translatorhttps://github.com/cefriel/mapping-template/blob/feat-rml-compiler/src/main/resources/rml/rml-compiler.vm
[17] https://github.com/SDM-TIB/SDM-RDFizer
[18] https://w3id.org/rml/portal
[19] https://github.com/oeg-upm/Conceptual-Mapping/tree/main/requirements
[20] The identifiers `cm-r26` and `cm-r28` are missing due to a numbering error in the original list of requirements [39].
[21] https://github.com/cefriel/mapping-template/tree/main/examples
[22] https://rml.io/specs/rml/
[23] The prefixes used in the snippets reported in the paper can be resolved by accessing the complete examples online.

a

```
#set($stops = $reader.getDataframe("
    for $stop in /transport/bus/route//stop
    return map {
        "stopId": $stop/@id,
        "stopName": $stop/text(),
        "busId": $stop/ancestor::bus/@id
}"))

#foreach($stop in $stops)
ex:$stop.busId a transit:Stop ;
  transit:stop "$stop.stopId"^^xsd:int ;
  rdfs:label "$stop.stopName" .
#end
```

b

```
<#TransportMapping> a rr:TriplesMap;
  rml:logicalSource [
    rml:source "Transport.xml" ;
    rml:iterator "/transport/bus";
    rml:referenceFormulation ql:XPath;
  ];

  rr:subjectMap [
    rr:template "http://trans.example.com/{@id}";
    rr:class transit:Stop
  ];

  rr:predicateObjectMap [
    rr:predicate transit:stop;
    rr:objectMap [
      rml:reference "route/stop/@id";
      rr:datatype xsd:int
    ]
  ];

  rr:predicateObjectMap [
    rr:predicate rdfs:label;
    rr:objectMap [
      rml:reference "route/stop"
    ]
  ].
```

c

```
#set ($data = $reader.getDataframe())

#foreach($row in $data)
    << << ex:$row.entity a ex:$row.type >>
        ex:confidence $row.confidence >>
        ex:predictedBy ex:$row.predictor .
#end
```

d

```
<#innerTriplesMap>
    a rml:NonAssertedTriplesMap;
    rml:logicalSource ex:PredictionsSource;
    rml:subjectMap [
        rml:template "http://example.com/{entity}";
    ];
    rml:predicateObjectMap [
        rml:predicate rdf:type;
        rml:objectMap [ rml:template "http://example.com/{class}" ];
    ].

<#middleTriplesMap>
    a rml:NonAssertedTriplesMap;
    rml:logicalSource ex:PredictionsSource;
    rml:subjectMap [
        rml:quotedTriplesMap <#innerTriplesMap>;
    ];
    rml:predicateObjectMap [
        rml:predicate ex:confidence;
        rml:objectMap [ rml:reference "confidence" ];
    ].

<#outerTriplesMap>
    a rml:AssertedTriplesMap;
    rml:logicalSource ex:PredictionsSource;
    rml:subjectMap [
        rml:quotedTriplesMap <#middleTriplesMap>;
    ];
    rml:predicateObjectMap [
        rml:predicate ex:predictedBy;
        rml:objectMap [ rml:template "http://example.com/{predictor}" ];
    ].
```

Fig. 3. Example mapping templates: (a) conversion from XML to RDF, (c) conversion from CSV to RDF-star. Snippet (b) contains RML mappings equivalent to (a), and snippet (d) contains RML-star mappings equivalent to (c).

A more complex example (`yarrrml-tutorial`) from the YARRRML tutorial shows how to apply a function for data transformation (`cm-r15`), join data frames from multiple data sources (`cm-r10`, `cm-r13`), and specify a named graph for the triples (`cm-r12`). All the functions made available in the tool's configuration can be invoked through a mapping template and applied as nested functions (`cm-r25`). Using an `if` directive in VTL, a function can also be used to conditionally generate a specific output (`cm-r22`).

The `mapping-template` tool allows via MTL to specify *inner* and *left* join operations between data frames. However, given a specific set of mapping rules, the user could also specify via MTL custom strategies to iterate more efficiently over the data frames. For example, a smaller data frame can be extracted to perform the join, considering only the data required. This approach addresses the mapping challenges related to join operations (`C5` and associated requirements `cm-r23`, `cm-r29`, `cm-r30`). Functions can be used in the template to conditionally determine the processing of the join operation (`cm-r24`).

The example `rml-star` considers a mapping scenario provided in the RML-star documentation with nested quoted triples generated from a CSV file. In Figure 3, the snippet (c) shows the mappings for our tool, and the snippet (d) shows the corresponding ones in RML-star. The example demonstrates how the template approach simplifies the definition of mapping rules towards a custom output without requiring the user to adopt a different

Identifier	Conceptual Mapping Requirement	Workflow Reference	Qualitative Evaluation
cm-r1	A mapping can describe data sources retrieved synchronously, asynchronously and as streams	Data Source Reading	Partially covered by MTL. Supported via Chimera pipeline.
cm-r2	A data source can describe data in different formats specifying its Media Type	Reference Formulation	rml-csv, rml-xml, rml-json, r2rml
cm-r3	A data source may have a specified data access service	Data Source Reading	Partially covered by MTL. Supported via Chimera pipeline.
cm-r4	A data access service can specify security terms	Data Source Reading	Not covered by MTL. Supported via Chimera pipeline.
cm-r5	A data access service can specify up to one retrieval protocol	Data Source Reading	Not covered by MTL. Supported via Chimera pipeline.
cm-r6	A data source can specify the encoding	Data Source Reading	Not covered by MTL. Supported via Chimera pipeline.
cm-r7	A mapping can describe data sources and their access	Data Source Reading	Partially covered by MTL. Supported via Chimera pipeline.
cm-r8	A mapping specifies statement maps to declare the transformation of frames into RDF triples	Schema Transformation Rules	rml-csv, rml-xml, rml-json, r2rml
cm-r9	Multiple data sources can be described with combined frames and nested frames	Data Frame Combination Rules	yarrrml-tutorial
cm-r10	A source frame describes exactly one data source	Data Frame Definition	rml-csv, rml-xml, rml-json, r2rml
cm-r11	A statement map can describe constant and dynamic subjects, predicates, objects	Schema Transformation Rules	rml-csv, rml-xml, rml-json, r2rml
cm-r12	A statement map can describe constant and dynamically up to 1 named graph, datatype and language tag	Schema Transformation Rules	rml-csv, rml-xml, rml-json, r2rml, yarrrml-tutorial
cm-r13	A resource map can reference one or more data fields from one or more data sources	Data Frame Combination Rules	yarrrml-tutorial
cm-r14	A resource map may contain data with different levels of iteration of a source	Flattening Strategy	Partially covered via MTL. A custom flattening strategy may be required.
cm-r15	A resource map may contain functions	Data Transformation Rules	yarrrml-tutorial
cm-r16	A subject can only be expressed as a blank node or an IRI	Schema Transformation Rules	rml-csv, rml-xml, rml-json, r2rml
cm-r17	A predicate can only be expressed as an IRI	Schema Transformation Rules	rml-csv, rml-xml, rml-json, r2rml
cm-r18	An object can only be expressed as a blank node, an IRI, a literal, a container or a list	Schema Transformation Rules	rml-csv, rml-xml, rml-json, r2rml, rml-star
cm-r19	A named graph can only be expressed as an IRI or blank node	Schema Transformation Rules	rml-csv, rml-xml, rml-json, r2rml
cm-r20	A datatype tag can only be expressed as an IRI or a List	Schema Transformation Rules	rml-csv, rml-xml, rml-json, r2rml
cm-r21	A language tag can only be expressed as a Literal or a List	Schema Transformation Rules	rml-csv, rml-xml, rml-json, r2rml
cm-r22	A statement map can be subject to a condition	Data Transformation Rules	csv-multiple-values
cm-r23	A statement map can be linked to another statement map with none, one or several conditions	Data Frame Combination Rules	yarrrml-tutorial
cm-r24	The condition to link statements may be any boolean condition	Data Frame Combination Rules	yarrrml-tutorial
cm-r25	A function may have nested functions	Data Transformation Rules	yarrrml-tutorial
cm-r27	A frame can have nested frames to access multi-value references	Flattening Strategy	Partially covered via MTL. A custom flattening strategy may be required.
cm-r29	A statement map can have objects of type literal using data from different sources using combined frames	Schema Transformation Rules	yarrrml-tutorial
cm-r30	A statement map can have datatype and language tags using data from different sources using combined frames	Schema Transformation Rules	yarrrml-tutorial

Table 2

Qualitative evaluation of the `mapping-template` tool.

syntax from the one used to generate plain RDF triples. Indeed, to generate RDF-star, a user knowing MTL should only be able to write RDF-star, while a user knowing RML should learn RML-star. The flexibility of the generated output also simplifies the generation of RDF collections and containers (`C4` and `cm-r18`).

The Mapping Template Language does not directly support a declarative *Data Source/Sink Specification* (`cm-r1`, from `cm-r2` to `cm-r7`). However, the integration of the `mapping-template` within Chimera enables the usage of the Apache Camel DSL and the available components to specify heterogeneous conversion pipelines as shown in the Chimera tutorial[24]. The definition of iterators in case of scenarios associated with complex nested data (`C2`, `cm-r14`, `cm-r27`) is delegated to the specific *Reader* or to the definition of flattening strategies via custom functions.

Finally, the flexibility of the approach based on templates enables the definition of mapping rules towards a generic output without requiring an extension of the syntax. The example `csv-to-json` demonstrates the definition of mapping rules for non-RDF output generating JSON data from an input CSV data source. Similarly, it is possible to generate a custom output considering different formats as input. An example template performing a lowering operation from RDF to CSV is available in the Chimera tutorial[24].

The evaluation summarised in Table 2 demonstrates how the `mapping-template` tool can (i) cover the core requirements identified for RDF knowledge graph construction (20 requirements fully covered, and 8 indirectly or partially) and (ii) generalise the mapping process towards non-RDF outputs.

To complement the qualitative evaluation, we also tested the compliance of the `mapping-template` tool with respect to the new RML test cases as part of track 1 of the Knowledge Graph Construction Challenge 2024 [63]. In particular, we focused on the evaluation of compliance considering the *RML Core* module, comprising all the core features of RML and targeting five different types of data sources: CSV, XML, JSON, MySQL and Postgres. For each test, the tool made available by the challenge organisers was used to automatically validate the obtained result against the expected one. All the MTL templates and outputs generated for each test case are made available online together with the configuration used to run the challenge tool[25]. As a result of the challenge, the tool was able to pass 219 over 238 and we identified and solved the following errors: (i) the SPARQL query accessing the RML mapping rules was not able to detect all inconsistencies in the provided RML, therefore, we introduced the validation via the RML SHACL shapes [38] of the input RML to automatically discard invalid mapping files; (ii) the need for adding a base IRI could not be evaluated only based on the mapping rules but its addition should consider the value extracted from the input data sources while processing the mapping rules; (iii) following R2RML, the RML spec mandates the automatic processing of certain values considering their datatype in the SQL database, therefore, it is needed to retrieve the type of columns when processing the mapping rules and to apply a custom function to the value according to the identified type (e.g., conversion from binary to Hex); (iv) the different conventions in MySQL and Postgres (e.g., regarding case sensitivity in table names) may cause issues in accessing the data sources for certain test cases, therefore, a specific logic is required to handle the two databases. The current version of the tool is able to pass all the 238 test cases defined by the *RML Core* module specification.

6. Quantitative Evaluation

This section presents a quantitative evaluation of the tool, demonstrating that the generic mapping approach proposed does not affect performances and can be comparable with state-of-the-art mapping processors considering an RDF graph construction task. Moreover, we discuss the advantages and drawbacks of adopting a template engine to execute the mapping rules as done for the `mapping-template` tool. In the presented diagrams the average metric is reported considering a logarithmic ($\log_{10}$) scale.

As a preliminary test for the performance and scalability of the `mapping-template` tool, we considered the GTFS-Madrid-Bench [13] following the approach by Arenas et al. in [4]. The benchmark provides a set of (R2)RML mappings and a generator to create input data sources in different formats and sizes. We considered three data formats (CSV, XML and JSON) and three scaling factors (1,10,100) comparing the `mapping-template` tool

[24] https://github.com/cefriel/chimera-tutorial
[25] https://github.com/cefriel/mapping-template-eval/tree/main/kgc-challenge-2024/track1

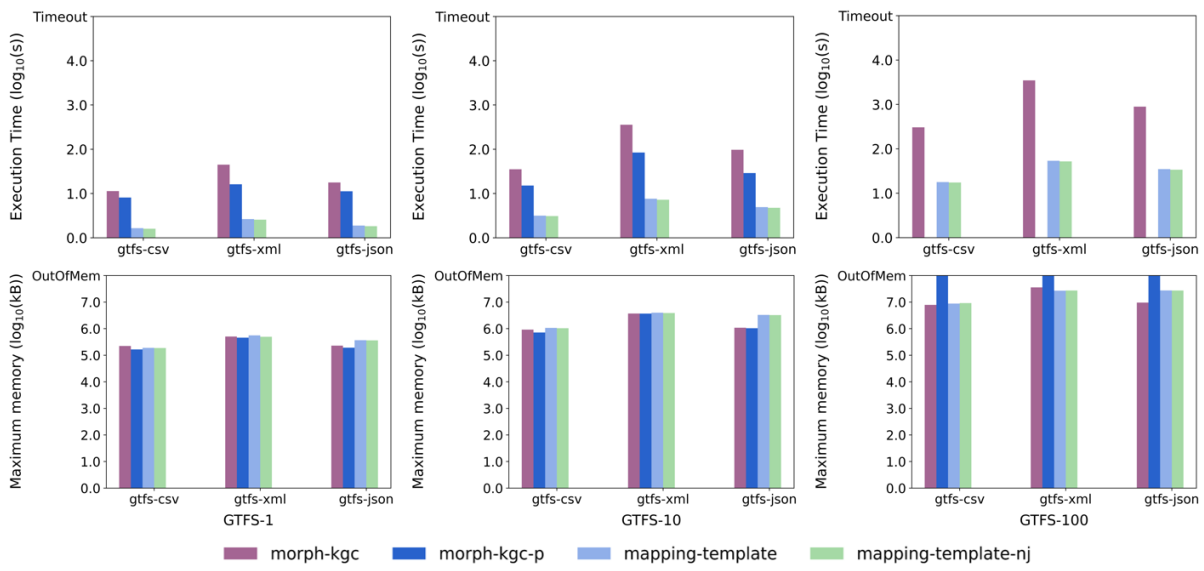

Fig. 4. Results of the quantitative evaluation comparing metrics using the GTFS Madrid Benchmark.

with the `morph-kgc` v2.3.1[26] processors. We selected `morph-kgc` for the evaluation considering its state-of-the-art performance and scalability results [6]. The `morph-kgc` processor was executed with parallel (`morph-kgc-p`) and sequential processing (`morph-kgc`). We adopted a set of RML mappings simplifying the join operation for the GTFS shapes file [4]. A set of templates implementing the same mapping rules was generated for the `mapping-template` tool. In this first set of templates, we defined a *join* operation between two data frames as specified by the join condition in RML. An additional set of templates, compared in the evaluation as `mapping-template-nj`, is defined to test the performances of the template approach using optimised mappings without join operations.

We consider the execution time (with a timeout of 24 hours) and the maximum memory used (each processor is run within a Docker container with a memory limit set at 64GB) as metrics for the evaluation. The experiments were executed on a virtual machine with 12 Intel(R) Xeon(R) E-2136 CPU @ 3.30GHz, 128 GB RAM and SSD.

Figure 4 reports the metrics registered for each configuration, each test was executed three times. The results show that the `mapping-template` tool completes the task with a lower execution time while registering similar memory consumption for all three data formats. The `morph-kgc` with parallel processing ran out of memory for the input data with scale 100. Interestingly, while the introduction of *join* conditions tends to affect the performance of processors based on RML [55, 66], the difference in the metrics for the `mapping-template` tool was limited in the performed evaluation. The results obtained can be motivated by the fact that the tool benefits from the efficient and optimised execution of templates provided by the Velocity Engine. However, it is also important to notice that the MTL allows the user to optimise the mapping rules according to the specific mapping scenario. For example, the number of data frames extracted from the input data sources can be minimised. Finally, it should be noted that, even if the inputs and mappings used for the evaluation do not generate duplicated triples, the `morph-kgc` execution time could be penalised by the fact that its implementation inherently guarantees the removal of duplicated triples before serializing the output.

To better investigate and compare the performances of the `mapping-template` tool with other mapping processors, we participated in track 2 for the Knowledge Graph Construction Challenge 2024 [63]. This track focused on the comparison of performances by requiring each tool to transform to RDF a set of input data sources according to a specific set of RML mapping rules. The first part of the challenge was based on the GTFS-Madrid-Bench to test the behaviour of the tools with different scales of the same data sources (1, 10, 100, 1000) and considering

[26] https://github.com/oeg-upm/morph-kgc

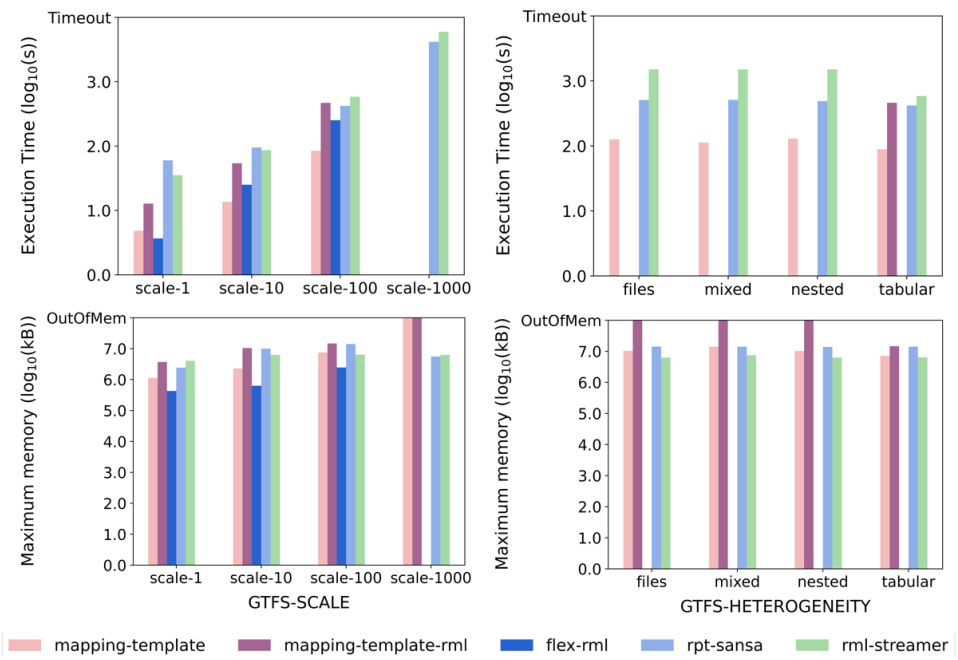

Fig. 5. Results using the GTFS Madrid Benchmark for the KGCW Challenge 2024.

heterogeneous mixes of data source types (tabular, files, nested, mixed). The second part of the challenge focused on the different parameters that may affect the mapping process [12] and defined different test cases by varying the number of data records, data properties, duplicate values, empty values, predicates and objects, join operations[27]. The organisers made available a tool [60] for the reproducible execution of the challenge, the collection of the metrics and the obtained results. Differently from the first edition of the challenge, each participant was provided with the same virtual-machine with the following characteristics: 4 vCPU, 16 Gb RAM, 130 Gb HD, Ubuntu OS. The complete specification of the test cases of the challenge and the set of raw results obtained by each tool is made available on Zenodo by the organisers [63].

We participated in the challenge prior to the development of direct support for RML mappings in the `mapping-template` tool. For this reason, we manually created a corresponding MTL template for a limited set of test cases to execute the same knowledge graph construction task[28]. Besides the penalisation due to the limited set of test cases executed, the `mapping-template` tool reached third place in the overall ranking. In particular, the `mapping-template` tool obtained the best results in terms of execution time and CPU usage while performing not as well in terms of memory consumption. To obtain a full comparison of performances, we complemented the results obtained for the challenge by running the same test cases with the new version of the `mapping-template` and providing the RML mapping files directly as input. The configuration for the evaluation and the raw results are made available online[3]. In the following paragraphs, we report and discuss the results of the challenge and one of the additional tests executed (indicated as `mapping-template-rml`). Each test was executed 5 times, and we report here the results of the other three mapping engines executing both parts of track 1: FlexRML [29], RPT/Sansa [59] and RML-Streamer with the RML-view-to-CSV [65].

Figure 5 provides the result for the first part associated with the GTFS-Madrid-Bench. It can be noted that the `mapping-template-rml` performance is not comparable with the one obtained by the `mapping-template` directly executing MTL mappings. This is due to the need, in the general case of translating RML to MTL, to

[27] For join operations also different type of join were tested but we do not report here the results since we did not obtain notable differences.

[28] We did not cover the test cases varying the number of join and mappings due to the presence of many different mapping files to be adapted. As a result, Figure 8 does not report data for the `mapping-template`.

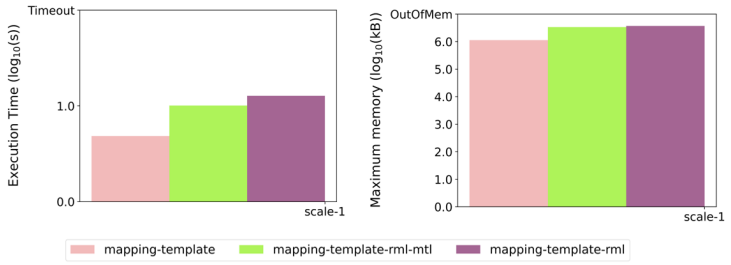

Fig. 6. Results comparing `mapping-template` mapping rules for GTFS Scale 1 for the KGCW Challenge 2024.

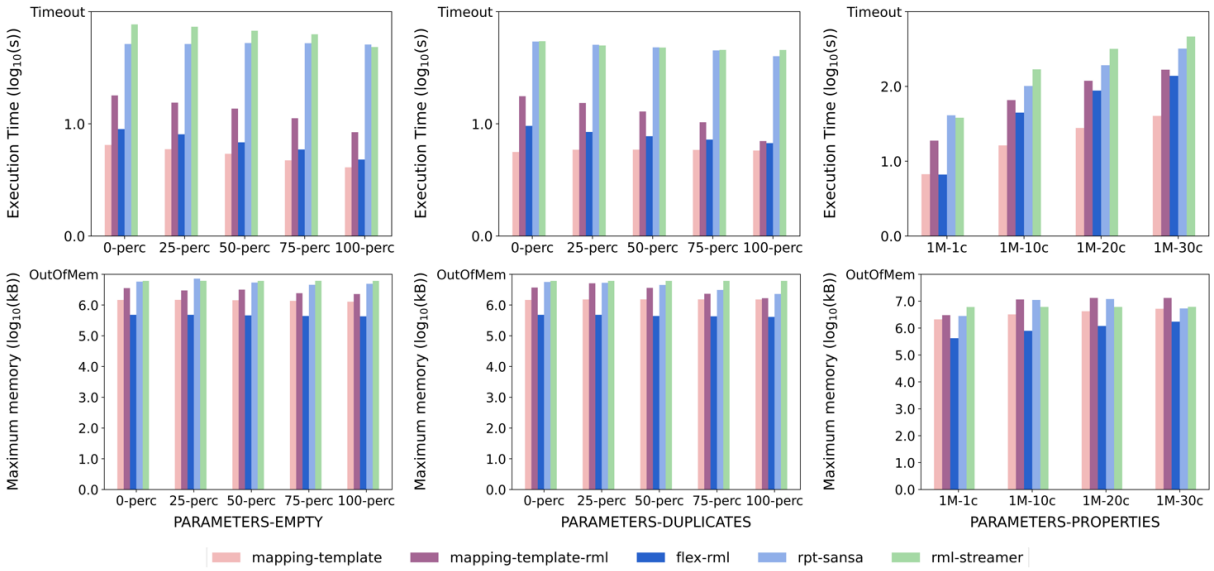

Fig. 7. Results for Track 2 - Knowledge Graph Construction Parameters of the KGCW Challenge 2024 (Empty, Duplicates, Properties).

introduce additional checks in the resulting mapping files to ensure the correctness of the generated output during its execution. Indeed, the MTL template can be simplified considering a specific mapping scenario, as we did when we manually defined the templates, but not in the general case. In the case of the heterogeneity tests, the `mapping-template-rml` was not able to run three tests due to the adoption of a JSON file for the *Shapes* file, causing out-of-memory errors due to the difficulties in optimising multiple accesses with JSONPath to the input file. Figure 6, report the GTFS-Madrid-Bench results for scale-1 for the `mapping-template` by adding also the metrics obtained for the case in which an MTL template generated from RML is directly executed with the tool, i.e., removing the translation overhead. This diagram shows, even with small-input files, that the difference of performances can not be attributed to such overhead.

Figure 7 and 8 report the results for all the other test cases associated with different parameters affecting the KG construction. The same trends commented on above can be observed for the test cases considering the different parameters. Overall, it emerges how the `mapping-template` is able to offer very good execution times while not managing to optimise memory consumption. Moreover, the tool works better when the size of the input is limited. The results obtained by executing directly the MTL templates demonstrate the advantages that can be achieved by optimising the mapping template rules for a specific mapping scenario.

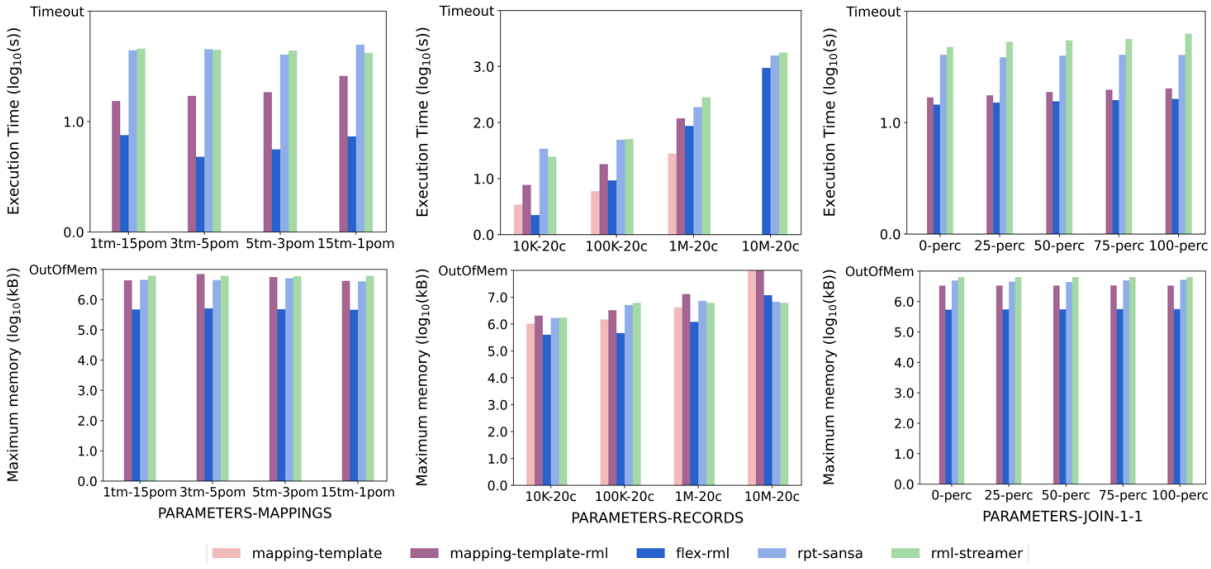

Fig. 8. Results for Track 2 - Part 1 Knowledge Graph Construction Parameters of the KGCW Challenge 2024 (Mapping, Records, Join 1-1).

7. Adoption cases of the `mapping-template` tool

The `mapping-template` tool has already been adopted to support different use cases. The integration within the Chimera framework enables its adoption for production-ready scenarios considering knowledge conversion among heterogeneous information systems. In particular, we used the tool to implement an any-to-one mapping approach for interoperability leveraging a reference ontology as a global conceptual model [31]. Such an approach can adopt (R2)RML for the lifting towards the reference ontology and the `mapping-template` tool for the lowering from RDF to the target representation. However, in our experience with the tool, it became clear that adopting a single approach for both data transformations can facilitate the definition of the mappings by external stakeholders. A preliminary feedback from Chimera users suggested a preference for the approach based on templates, because of developers' familiarity with similar technologies. In the EIT Digital SNAP project[29], the tool supported the lowering of transportation data from an RDF representation according to a reference ontology to the standards mandated by the European Commission in XML format [54]. In the Horizon 2020 SPRINT project[30], the tool was used for the dynamic definition of converters for dataset conversion and service mediation [55] to support data exchanges between transportation operators. In the Horizon 2020 TANGENT project[31], the tool was used to implement the project solution for data harmonisation and fusion [31, 56]. The data are retrieved from heterogeneous data services from different stakeholders, lifted to RDF according to a common suite of ontologies, and then converted to a predefined set of JSON schemas feeding applications for the dynamic management of multimodal traffic. Within TANGENT, we also tested the tool for the definition and execution of a set of templates facilitating the serialisation of a portion of the reference conceptual model from CSV files to OWL ontologies. In the Horizon Europe SmartEdge project [3], we are further developing the tool to increase its maturity level, support the mediation of data exchanges between different IoT nodes and improve performance and scalability for execution on resource-constrained devices.

[29] https://snap-project.eu
[30] http://sprint-transport.eu/
[31] https://tangent-h2020.eu/

8. Conclusions and Future Work

In this paper, we presented a workflow and the related `mapping-template`[1] tool for knowledge conversion between different data representations. In particular, we extend the focus from approaches that address only the lifting problem (to RDF) to solutions that address both the lifting and lowering problems (both to/from RDF), as well as generic conversions between different data formats and models to support integration requirements among heterogeneous information systems. Our workflow is soundly based on the literature about declarative RDF graph construction and brings together different contributions to support the definition of a generalised declarative mapping process. The tool implements the proposed workflow by adopting a template-based mapping language to overcome some of the limitations of the state-of-the-art approaches such as the difficulties in extending and maintaining a fully declarative specification to define the desired output (e.g., the effort for developers in adapting existing mapping processors and for the users in learning the new syntax for RML-star [26]). We also presented a preliminary qualitative and quantitative evaluation. We showed how the proposed approach can cover the requirements for RDF graph generation and we also analysed the performance of our implementation on an RDF graph construction task.

The need for the `mapping-template` tool is motivated by our experience and difficulties in applying existing solutions to practical use cases in the mobility and industrial markets, in which we validated our approach. The presented tool is maintained as a company asset by Cefriel to support its value proposition on KG construction and data interoperability both in customer projects and research projects. We publicly released the `mapping-template` with an open-source license on GitHub, where we also provide a guide to create template-based mappings and examples considering different mapping scenarios. Furthermore, we integrated the tool within the Chimera[14] framework to support enterprise integration practices and we provide an end-to-end tutorial implementing a data conversion pipeline. To ease its adoption by the community and to ensure long-term sustainability, we also released the software tool on Maven Central.

Our presented resource is of interest for the knowledge graph construction community, but also for the developers' community in general, to address data heterogeneity problems. For an average developer, without a deep understanding of RDF, our template-based approach appears to be less verbose and simpler than RML-based solutions. In future work, we plan to perform a user study to compare the Mapping Template Language with other declarative mapping languages for RDF generation. For example, we would like to investigate the preferred approach for users with different technical expertise (e.g., no developer background). Moreover, we would like to investigate a better formalisation of MTL to enable the definition of alternative implementations, e.g., in Python using the Jinja template engine[32].

Acknowledgments

The presented research was partially supported by the SMARTEDGE project, funded under the Horizon Europe RIA Research and Innovation Programme (Grant Agreement 101092908).

References

[1] W. Akhtar, J. Kopecký, T. Krennwallner and A. Polleres, XSPARQL: Traveling between the XML and RDF Worlds - and Avoiding the XSLT Pilgrimage, in: *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*, S. Bechhofer, M. Hauswirth, J. Hoffmann and M. Koubarakis, eds, Lecture Notes in Computer Science, Vol. 5021, Springer, 2008, pp. 432–447. doi:10.1007/978-3-540-68234-9_33.

[2] C. Allocca and A. Gougousis, A Preliminary Investigation of Reversing RML: From an RDF dataset to its Column-Based data source, *Biodiversity data journal* **3** (2015), e5464. doi:10.3897/BDJ.3.e5464.

[3] D. Anicic et al., SmartEdge project Deliverable D3.1 – Design of Tools for Continuous Semantic Integration, 2023. https://www.smart-edge.eu/deliverables/.

[32] https://github.com/pallets/jinja

[4] J. Arenas-Guerrero, M. Scrocca, A. Iglesias-Molina, J. Toledo, L. Pozo-Gilo, D. Doña, Ó. Corcho and D. Chaves-Fraga, Knowledge Graph Construction with R2RML and RML: An ETL System-based Overview, in: *Proceedings of the 2nd International Workshop on Knowledge Graph Construction co-located with 18th Extended Semantic Web Conference (ESWC 2021), Online, June 6, 2021*, CEUR Workshop Proceedings, Vol. 2873, CEUR-WS.org, 2021. http://ceur-ws.org/Vol-2873/paper11.pdf.

[5] J. Arenas-Guerrero, A. Alobaid, M. Navas-Loro, M.S. Pérez and O. Corcho, Boosting Knowledge Graph Generation from Tabular Data with RML Views, in: *The Semantic Web: 20th International Conference, ESWC 2023, Hersonissos, Crete, Greece, May 28–June 1, 2023, Proceedings*, Springer, 2023, pp. 484–501. doi:10.1007/978-3-031-33455-9_29.

[6] J. Arenas-Guerrero, D. Chaves-Fraga, J. Toledo, M.S. Pérez and O. Corcho, Morph-KGC: Scalable knowledge graph materialization with mapping partitions, *Semantic Web* (2022). doi:10.3233/SW-223135.

[7] L. Asprino, E. Daga, A. Gangemi and P. Mulholland, Knowledge Graph Construction with a Façade: A Unified Method to Access Heterogeneous Data Sources on the Web, *ACM Trans. Internet Technol.* (2022). doi:10.1145/3555312.

[8] M. Bennara, A. Zimmermann, M. Lefrançois and N. Messalti, Interoperability of Semantically-Enabled Web Services on the WoT: Challenges and Prospects, in: *Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services*, 2020, pp. 149–153. doi:10.1145/3428757.3429199.

[9] S. Bischof, S. Decker, T. Krennwallner, N. Lopes and A. Polleres, Mapping between RDF and XML with XSPARQL, *J. Data Semant.* **1**(3) (2012), 147–185. doi:10.1007/S13740-012-0008-7. https://doi.org/10.1007/s13740-012-0008-7.

[10] A. Carenini et al., ST4RT Semantic Transformations for Rail Transportation, in: *7th Transport Research Arena (TRA 2018)*, Zenodo, 2018. doi:10.5281/zenodo.1440984.

[11] J. Chakraborty, A. Padki and S.K. Bansal, Semantic ETL - State-of-the-Art and Open Research Challenges, in: *11th IEEE International Conference on Semantic Computing, ICSC 2017, San Diego, CA, USA, January 30 - February 1, 2017*, IEEE Computer Society, 2017, pp. 413–418. doi:10.1109/ICSC.2017.94.

[12] D. Chaves-Fraga, K.M. Endris, E. Iglesias, Ó. Corcho and M. Vidal, What are the Parameters that Affect the Construction of a Knowledge Graph?, in: *On the Move to Meaningful Internet Systems*, Springer, 2019, pp. 695–713. doi:10.1007/978-3-030-33246-4_43.

[13] D. Chaves-Fraga, F. Priyatna, A. Cimmino, J. Toledo, E. Ruckhaus and O. Corcho, GTFS-Madrid-Bench: A benchmark for virtual knowledge graph access in the transport domain, *Journal of Web Semantics* **65** (2020), 100596. doi:10.1016/j.websem.2020.100596.

[14] D. Chaves-Fraga, A. Dimou, A. Iglesias-Molina, U. Serles and D.V. Assche, Preface for the 4th Edition of the International Knowledge Graph Construction Workshop, in: *Proceedings of the 4th International Workshop on Knowledge Graph Construction co-located with 20th Extended Semantic Web Conference*, CEUR Workshop Proceedings, Vol. 3471, CEUR, Hersonissos, Greece, 2023, ISSN: 1613-0073. https://ceur-ws.org/Vol-3471/preface.pdf.

[15] D. Chaves-Fraga, A. Dimou, A. Iglesias-Molina, U. Serles and D.V. Assche, Preface for the 5th Edition of the International Knowledge Graph Construction Workshop, in: *Proceedings of the 5th International Workshop on Knowledge Graph Construction*, CEUR Workshop Proceedings, Vol. 3718, CEUR, Hersonissos, Greece, 2024, ISSN: 1613-0073. https://ceur-ws.org/Vol-3718/preface.pdf.

[16] A. Chortaras and G. Stamou, D2RML: Integrating Heterogeneous Data and Web Services into Custom RDF Graphs, in: *Workshop on Linked Data on the Web co-located with The Web Conference 2018, LDOW@WWW 2018, Lyon, France April 23rd, 2018*, T. Berners-Lee, S. Capadisli, S. Dietze, A. Hogan, K. Janowicz and J. Lehmann, eds, CEUR Workshop Proceedings, Vol. 2073, CEUR-WS.org, 2018. http://ceur-ws.org/Vol-2073/article-07.pdf.

[17] A. Cimmino and R. García-Castro, Helio: A framework for implementing the life cycle of knowledge graphs, *Semantic Web* **15**(1) (2024), 223–249, Publisher: IOS Press. doi:10.3233/SW-233224. https://content.iospress.com/articles/semantic-web/sw233224.

[18] M. Comerio, A. Fiano, M. Grassi and M. Scrocca, Mobility data harmonisation: the TANGENT solution, in: *Proceedings of the 10th Transport Research Arena*, Lecture Notes in Mobility, Springer, 2024, (to be published).

[19] O. Corby and C. Faron-Zucker, A Transformation Language for RDF Based on SPARQL, in: *Web Information Systems and Technologies*, Lecture Notes in Business Information Processing, Springer International Publishing, Cham, 2016, pp. 318–340. ISBN 978-3-319-30996-5. doi:10.1007/978-3-319-30996-5_16.

[20] R. Cyganiak, Tarql (sparql for tables): Turn csv into rdf using sparql syntax, Technical Report, 2015, http://tarql.github.io.

[21] S. Das, S. Sundara and R. Cyganiak, R2RML: RDB to RDF Mapping Language, W3C Recommendation, W3C, 2012, http://www.w3.org/TR/2012/REC-r2rml-20120927/.

[22] I. Dasoulas, D. Chaves-Fraga, D. Garijo and A. Dimou, Declarative RDF Construction from In-Memory Data Structures with RML, in: *Proceedings of the 4th International Workshop on Knowledge Graph Construction co-located with 20th Extended Semantic Web Conference*, D. Chaves-Fraga, A. Dimou, A. Iglesias-Molina, U. Serles and D.V. Assche, eds, CEUR Workshop Proceedings, Vol. 3471, CEUR, Hersonissos, Greece, 2023, ISSN: 1613-0073. https://ceur-ws.org/Vol-3471/#paper7.

[23] B. De Meester, T. Seymoens, A. Dimou and R. Verborgh, Implementation-independent function reuse, *Future Generation Computer Systems* **110** (2020), 946–959. doi:https://doi.org/10.1016/j.future.2019.10.006. https://www.sciencedirect.com/science/article/pii/S0167739X19303723.

[24] C. Debruyne and D. O'Sullivan, R2RML-F: Towards Sharing and Executing Domain Logic in R2RML Mappings, in: *Proceedings of the Workshop on Linked Data on the Web, LDOW 2016, co-located with 25th International World Wide Web Conference (WWW 2016)*, S. Auer, T. Berners-Lee, C. Bizer and T. Heath, eds, CEUR Workshop Proceedings, Vol. 1593, CEUR-WS.org, 2016. http://ceur-ws.org/Vol-1593/article-13.pdf.

[25] T. Delva, D.V. Assche, P. Heyvaert, B.D. Meester and A. Dimou, Integrating Nested Data into Knowledge Graphs with RML Fields, in: *Proceedings of the 2nd International Workshop on Knowledge Graph Construction co-located with 18th Extended Semantic Web Conference (ESWC 2021), Online, June 6, 2021*, CEUR Workshop Proceedings, Vol. 2873, CEUR-WS.org, 2021. http://ceur-ws.org/Vol-2873/paper9.pdf.

[26] T. Delva, J. Arenas-Guerrero, A. Iglesias-Molina, O. Corcho, D. Chaves-Fraga and A. Dimou, RML-star: A Declarative Mapping Language for RDF-star Generation, in: *Proceedings of the ISWC 2021 Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice*, O. Seneviratne, C. Pesquita, J. Sequeda and L. Etcheverry, eds, CEUR Workshop Proceedings, Vol. 2980, CEUR, Virtual Conference, October, 2021, ISSN: 1613-0073. https://ceur-ws.org/Vol-2980/#paper374.

[27] T. Delva, J. Arenas-Guerrero, A. Iglesias-Molina, Ó. Corcho, D. Chaves-Fraga and A. Dimou, RML-star: A Declarative Mapping Language for RDF-star Generation, in: *Proceedings of the ISWC 2021 Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 20th International Semantic Web Conference (ISWC 2021), Virtual Conference, October 24-28, 2021*, CEUR Workshop Proceedings, Vol. 2980, CEUR-WS.org, 2021. http://ceur-ws.org/Vol-2980/paper374.pdf.

[28] A. Dimou, M.V. Sande, P. Colpaert, R. Verborgh, E. Mannens and R.V. de Walle, RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data, in: *Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014)*, Vol. 1184, CEUR-WS.org, 2014. http://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf.

[29] M. Freund, S. Schmid, R. Dorsch and A. Harth, FlexRML: A Flexible and Memory Efficient Knowledge Graph Materializer, in: *The Semantic Web*, Springer Nature Switzerland, Cham, 2024, pp. 40–56. ISBN 978-3-031-60635-9. doi:10.1007/978-3-031-60635-9_3.

[30] H. García-González, D. Fernández-Álvarez and J.E.L. Gayo, ShExML: An Heterogeneous Data Mapping Language based on ShEx, in: *Proceedings of the EKAW 2018 Posters and Demonstrations Session co-located with 21st International Conference on Knowledge Engineering and Knowledge Management (EKAW 2018), Nancy, France, November 12-16, 2018*, P. Cimiano and O. Corby, eds, CEUR Workshop Proceedings, Vol. 2262, CEUR-WS.org, 2018, pp. 9–12. http://ceur-ws.org/Vol-2262/ekaw-poster-08.pdf.

[31] M. Grassi, M. Scrocca, A. Carenini, M. Comerio and I. Celino, Composable Semantic Data Transformation Pipelines with Chimera, in: *Proceedings of the 4th International Workshop on Knowledge Graph Construction co-located with 20th Extended Semantic Web Conference*, D. Chaves-Fraga, A. Dimou, A. Iglesias-Molina, U. Serles and D.V. Assche, eds, CEUR Workshop Proceedings, Vol. 3471, CEUR, Hersonissos, Greece, 2023, ISSN: 1613-0073. https://ceur-ws.org/Vol-3471/paper9.pdf.

[32] S. Gössner, G. Normington and C. Bormann, JSONPath: Query Expressions for JSON, *Request for Comments*, RFC Editor, 2024. doi:10.17487/RFC9535. https://www.rfc-editor.org/info/rfc9535.

[33] O. Hartig et al., RDF-star and SPARQL-star, RDF-DEV Community Group, 2021. https://w3c.github.io/rdf-star/cg-spec.

[34] M. Hert, G. Reif and H.C. Gall, A Comparison of RDB-to-RDF Mapping Languages, in: *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 2532–. ISBN 9781450306218. doi:10.1145/2063518.2063522.

[35] P. Heyvaert, B. De Meester, A. Dimou and R. Verborgh, Declarative Rules for Linked Data Generation at Your Fingertips!, in: *European Semantic Web Conference*, Springer, 2018, pp. 213–217.

[36] E. Iglesias, S. Jozashoori and M.-E. Vidal, Scaling up knowledge graph creation to large and heterogeneous data sources, *Journal of Web Semantics* **75** (2023), 100755. doi:10.1016/j.websem.2022.100755. https://www.sciencedirect.com/science/article/pii/S1570826822000397.

[37] E. Iglesias, S. Jozashoori, D. Chaves-Fraga, D. Collarana and M. Vidal, SDM-RDFizer: An RML Interpreter for the Efficient Creation of RDF Knowledge Graphs, in: *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, M. d'Aquin, S. Dietze, C. Hauff, E. Curry and P. Cudré-Mauroux, eds, ACM, 2020, pp. 3039–3046. doi:10.1145/3340531.3412881.

[38] A. Iglesias-Molina, D. Van Assche, J. Arenas-Guerrero, B. De Meester, C. Debruyne, S. Jozashoori, P. Maria, F. Michel, D. Chaves-Fraga and A. Dimou, The RML Ontology: A Community-Driven Modular Redesign After a Decade of Experience in Mapping Heterogeneous Data to RDF, in: *The Semantic Web ISWC 2023*, T.R. Payne, V. Presutti, G. Qi, M. Poveda-Villalón, G. Stoilos, L. Hollink, Z. Kaoudi, G. Cheng and J. Li, eds, Lecture Notes in Computer Science, Springer Nature Switzerland, Cham, 2023, pp. 152–175. ISBN 978-3-031-47243-5. doi:10.1007/978-3-031-47243-5_9.

[39] A. Iglesias-Molina, A. Cimmino, E. Ruckhaus, D. Chaves-Fraga, R. García-Castro and O. Corcho, An ontological approach for representing declarative mapping languages, *Semantic Web* **15**(1) (2024), 191–221. doi:10.3233/SW-223224.

[40] F. Jouault, F. Allilaire, J. Bézivin and I. Kurtev, ATL: A model transformation tool, *Sci. Comput. Program.* **72**(1–2) (2008), 31–39. doi:10.1016/j.scico.2007.08.002.

[41] S. Jozashoori, E. Iglesias and M. Vidal, Dragoman: Efficiently Evaluating Declarative Mapping Languages over Frameworks for Knowledge Graph Creation (2022), (pre-print). doi:10.48550/arXiv.2210.15645.

[42] S. Jozashoori, D. Chaves-Fraga, E. Iglesias, M. Vidal and Ó. Corcho, FunMap: Efficient Execution of Functional Mappings for Knowledge Graph Creation, in: *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part I*, J.Z. Pan, V.A.M. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne and L. Kagal, eds, Lecture Notes in Computer Science, Vol. 12506, Springer, 2020, pp. 276–293. doi:10.1007/978-3-030-62419-4_16.

[43] J. Klímek, P. Škoda and M. Nečaskỳ, LinkedPipes ETL: Evolved linked data preparation, in: *European Semantic Web Conference*, Springer, 2016, pp. 95–100.

[44] T. Knap, M. Kukhar, B. Macháč, P. Škoda, J. Tomeš and J. Vojt, UnifiedViews: An ETL framework for sustainable RDF data processing, in: *European Semantic Web Conference*, Springer, 2014, pp. 379–383.

[45] D. Lanti, G. Xiao and D. Calvanese, VIG: Data scaling for OBDA benchmarks, *Semantic Web* **10**(2) (2019), 413–433.

[46] M. Lefrançois, A. Zimmermann and N. Bakerally, Flexible RDF generation from RDF and heterogeneous data sources with SPARQL-Generate, in: *Proceedings of the 20th International Conference on Knowledge Engineering and Knowledge Management (EKAW'16)*, Bologna, Italy, 2016.

[47] M. Lefrançois, A. Zimmermann and N. Bakerally, Flexible RDF Generation from RDF and Heterogeneous Data Sources with SPARQL-Generate, in: *Knowledge Engineering and Knowledge Management - EKAW 2016 Satellite Events, EKM and Drift-an-LOD, Bologna, Italy, November 19-23, 2016, Revised Selected Papers*, P. Ciancarini, F. Poggi, M. Horridge, J. Zhao, T. Groza, M.C. Suárez-Figueroa, M. d'Aquin and V. Presutti, eds, Lecture Notes in Computer Science, Vol. 10180, Springer, 2016, pp. 131–135. doi:10.1007/978-3-319-58694-6_16.

[48] A. Makinouchi, A Consideration on Normal Form of Not-necessarily-normalized Relation in the Relational Data Model., in: *VLDB*, Vol. 1977, Citeseer, 1977, pp. 447–453.

[49] B.D. Meester, W. Maroy, A. Dimou, R. Verborgh and E. Mannens, RML and FnO: Shaping DBpedia Declaratively, in: *The Semantic Web: ESWC 2017 Satellite Events*, Vol. 10577, Springer, 2017, pp. 172–177. doi:10.1007/978-3-319-70407-4_32.

[50] F. Michel, L. Djimenou, C. Faron-Zucker and J. Montagnat, Translation of Relational and Non-relational Databases into RDF with xR2RML, in: *WEBIST 2015 - Proceedings of the 11th International Conference on Web Information Systems and Technologies, Lisbon, Portugal, 20-22 May, 2015*, SciTePress, 2015, pp. 443–454. doi:10.5220/0005448304430454.

[51] S.M. Oo, T. Verbeken and B.D. Meester, RMLWeaver-JS: An algebraic mapping engine in the KGCW Challenge 2024, in: *Proceedings of the 5th International Workshop on Knowledge Graph Construction*, CEUR Workshop Proceedings, Vol. 3718, CEUR, Hersonissos, Greece, 2024, ISSN: 1613-0073. https://ceur-ws.org/Vol-3718/paper8.pdf.

[52] S.M. Oo, G. Haesendonck, B. De Meester and A. Dimou, RMLStreamer-SISO: An RDF Stream Generator fromăStreaming Heterogeneous Data, in: *The Semantic Web ISWC 2022*, Springer International Publishing, Cham, 2022, pp. 697–713. ISBN 978-3-031-19433-7. doi:10.1007/978-3-031-19433-7_40.

[53] J. Robie, M. Dyck and J. Spiege, XQuery 3.1: An XML Query Language, W3C Recommendation, W3C, 2017, https://www.w3.org/TR/xquery-31/.

[54] M. Scrocca, M. Comerio, A. Carenini and I. Celino, Turning Transport Data to Comply with EU Standards While Enabling a Multimodal Transport Knowledge Graph, in: *Proceedings of the 19th International Semantic Web Conference*, Vol. 12507, Springer, 2020, pp. 411–429. doi:10.1007/978-3-030-62466-8_26.

[55] M. Scrocca, A. Carenini, M. Comerio and I. Celino, Semantic Conversion of Transport Data Adopting Declarative Mappings: An Evaluation of Performance and Scalability, in: *Proceedings of the 3rd International Workshop Semantics And The Web For Transport*, D. Chaves-Fraga, P. Colpaert, M. Sadeghi, M. Scrocca and M. Comerio, eds, CEUR Workshop Proceedings, Vol. 2939, CEUR, Online, September, 2021, ISSN: 1613-0073. https://ceur-ws.org/Vol-2939/#paper2.

[56] M. Scrocca, M. Grassi, M. Comerio, V.A. Carriero, T.D. Dias, A.V.D. Silva and I. Celino, Intelligent Urban Traffic Management via Semantic Interoperability across Multiple Heterogeneous Mobility Data Sources, in: *The Semantic Web ISWC 2024*, 2024, (to appear). https://arxiv.org/abs/2407.10539.

[57] M.G. Skjæveland, D.P. Lupp, L.H. Karlsen and J.W. Klüwer, OTTR: Formal Templates for Pattern-Based Ontology Engineering, in: *Advances in Pattern-Based Ontology Engineering*, IOS Press, 2021, pp. 349–377. doi:10.3233/SSW210025.

[58] J. Slepicka, C. Yin, P. Szekely and C. Knoblock, KR2RML: An Alternative Interpretation of R2RML for Heterogenous Sources, in: *Proceedings of the 6th International Workshop on Consuming Linked Data*, O. Hartig, J. Sequeda and A. Hogan, eds, CEUR Workshop Proceedings, Vol. 1426, CEUR, Bethlehem, Pennsylvania, 2015, ISSN: 1613-0073. https://ceur-ws.org/Vol-1426/#paper-08.

[59] C. Stadler, L. Bühmann, L.-P. Meyer and M. Martin, Scaling RML and SPARQL-based Knowledge Graph Construction with Apache Spark, in: *Proceedings of the 4th International Workshop on Knowledge Graph Construction co-located with 20th Extended Semantic Web Conference*, CEUR Workshop Proceedings, Vol. 3471, CEUR, Hersonissos, Greece, 2023, ISSN: 1613-0073. https://ceur-ws.org/Vol-3471/paper8.pdf.

[60] D. Van Assche, D. Chaves-Fraga and A. Dimou, KROWN: A Benchmark for RDF Graph Materialization, 2024. doi:10.5281/zenodo.1097932. https://w3id.org/kg-construct/KROWN.

[61] D. Van Assche, G. Haesendonck, G. De Mulder, T. Delva, P. Heyvaert, B. De Meester and A. Dimou, Leveraging Web of Things W3C Recommendations for Knowledge Graphs Generation, in: *Web Engineering*, M. Brambilla, R. Chbeir, F. Frasincar and I. Manolescu, eds, Springer International Publishing, Cham, 2021, pp. 337–352. ISBN 978-3-030-74296-6.

[62] D. Van Assche, T. Delva, G. Haesendonck, P. Heyvaert, B. De Meester and A. Dimou, Declarative RDF graph generation from heterogeneous (semi-)structured data: A systematic literature review, *Web Semant.* **75**(C) (2023). doi:10.1016/j.websem.2022.100753.

[63] D. Van Assche, D. Chaves-Fraga, A. Dimou, U. Serles and A. Iglesias, KGCW 2024 Challenge @ ESWC 2024, Zenodo, 2024. doi:10.5281/zenodo.11577087.

[64] G. Vetere and M. Lenzerini, Models for semantic interoperability in service-oriented architectures, *IBM Systems Journal* **44**(4) (2005), 887–903. doi:10.1147/sj.444.0887.

[65] E.d. Vleeschauwer and B.D. Meester, RMLStreamer supported by RML-view-to-CSV in the performance track of the KGCW Challenge 2024, in: *Proceedings of the 5th International Workshop on Knowledge Graph Construction*, D. Chaves-Fraga, A. Dimou, A. Iglesias-Molina, U. Serles and D.V. Assche, eds, CEUR Workshop Proceedings, Vol. 3718, CEUR, Hersonissos, Greece, 2024, ISSN: 1613-0073. https://ceur-ws.org/Vol-3718/paper7.pdf.

[66] E.d. Vleeschauwer, S.M. Oo, B.D. Meester and P. Colpaert, Reference Conditions: Relating Mapping Rules Without Joining, in: *Proceedings of the 4th International Workshop on Knowledge Graph Construction co-located with 20th Extended Semantic Web Conference*, D. Chaves-Fraga, A. Dimou, A. Iglesias-Molina, U. Serles and D.V. Assche, eds, CEUR Workshop Proceedings, Vol. 3471, CEUR, Hersonissos, Greece, 2023, ISSN: 1613-0073. https://ceur-ws.org/Vol-3471/paper10.pdf.

[67] E.d. Vleeschauwer, P. Maria, B.D. Meester and P. Colpaert, RML-view-to-CSV: A Proof-of-Concept Implementation for RML Logical Views, in: *Proceedings of the 5th International Workshop on Knowledge Graph Construction*, CEUR Workshop Proceedings, Vol. 3718, CEUR, Hersonissos, Greece, 2024, ISSN: 1613-0073. https://ceur-ws.org/Vol-3718/paper2.pdf.

[68] B. Vu, J. Pujara and C.A. Knoblock, D-REPR: A Language for Describing and Mapping Diversely-Structured Data Sources to RDF, in: *Proceedings of the 10th International Conference on Knowledge Capture*, K-CAP '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 189196–. ISBN 9781450370080. doi:10.1145/3360901.3364449.