Not Everybody Speaks RDF: Knowledge Conversion between Different Data Representations

Mario Scrocca, Alessio Carenini, Marco Grassi, Marco Comerio and Irene Celino

Cefriel – Politecnico di Milano, Milan, Italy

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.

Keywords

Declarative Mappings, Knowledge Graph Construction, 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 [1]. 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 [2], and (ii) the improvement of performances for mapping processors against the identified benchmarks [3, 4]. However, not all information systems are able to process information represented as RDF. In

KGCW'24: 5th International Workshop on Knowledge Graph Construction, May 27, 2024, Crete, GRE

mario.scrocca@cefriel.com (M. Scrocca); alessio.carenini@cefriel.com (A. Carenini); marco.grassi@cefriel.com (M. Grassi); marco.comerio@cefriel.com (M. Comerio); irene.celino@cefriel.com (I. Celino)

0000-0002-8235-7331 (M. Scrocca); 0000-0003-1948-807X (A. Carenini); 000-0003-3139-3049 (M. Grassi); 0000-0003-3494-9516 (M. Comerio); 0000-0001-9962-7193 (I. Celino)

these cases, a knowledge conversion process from the knowledge graph towards a generic output is required [5, 6]. 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 Section 4 presents the tool. 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. 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[3] 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 [7] beyond relational databases [2]. 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. [1].

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.

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

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.

2.2. Declarative mapping languages and processors

The process of RDF graph generation from (semi-)structured data encompasses three main approaches [1]:

- *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 [7], RML [8], D2RML [9], KR2RML [10], R2RML-F [11], xR2RML [12]), (ii) dedicated languages with custom syntax (Helio Mapping Language [13], D-REPR [14]), (iii) repurposed languages based on constraint languages (ShExML [15] extending the ShEx syntax), (iv) repurposed languages based on SPARQL syntax: XSPARQL [16], Facade-X [17], SPARQL-Generate [18]. 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 [19] 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[4]. The high-level concepts identified by the ontology are considered in this work to support the workflow definition. Moreover, we use the ontological requirements[5] for the evaluation of the mapping capabilities of our tool.

The literature review by Van Assche et al. [1] 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

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

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 [20] to enable the declaration of data transformations in the mappings. The authors in [21] 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 [23] are proposed to facilitate the mapping of tabular data sources. The RML-star [24] extension for the RML language enables the definition of declarative mappings to generate RDF-star [25] triples, while the RML-CC[6] extension allows generating RDF Collections and Containers. Finally, the RML Fields [26] proposal defines an approach to handle mapping rules for complex nested data structures. The new RML ontology [2] 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.

Another set of contributions targets the performance and scalability of the mapping process. The GTFS-Madrid-Bench [27] defines a benchmark to test the scalability of solutions for knowledge graph construction. A qualitative and quantitative comparison of different (R2)RML processors is provided by Arenas-Guerrero et al. in [3]. Optimisations for the processing of data transformations defined within the mappings are proposed by FunMap [28] and Dragoman [29]. The usage of support data structures to speed up the mapping execution is proposed by SDM-RDFizer [30]. Finally, the concurrent processing of mapping rules is investigated by Chimera [31] and Morph-KGC [32].

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. [5] 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 [6]. 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 [33]. The SPARQL Template Transformation Language (STTL) [34]

[6] https://github.com/kg-construct/rml-cc

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 [35, 36] 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 [3] 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.

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[7] 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[8]) usually define their result

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

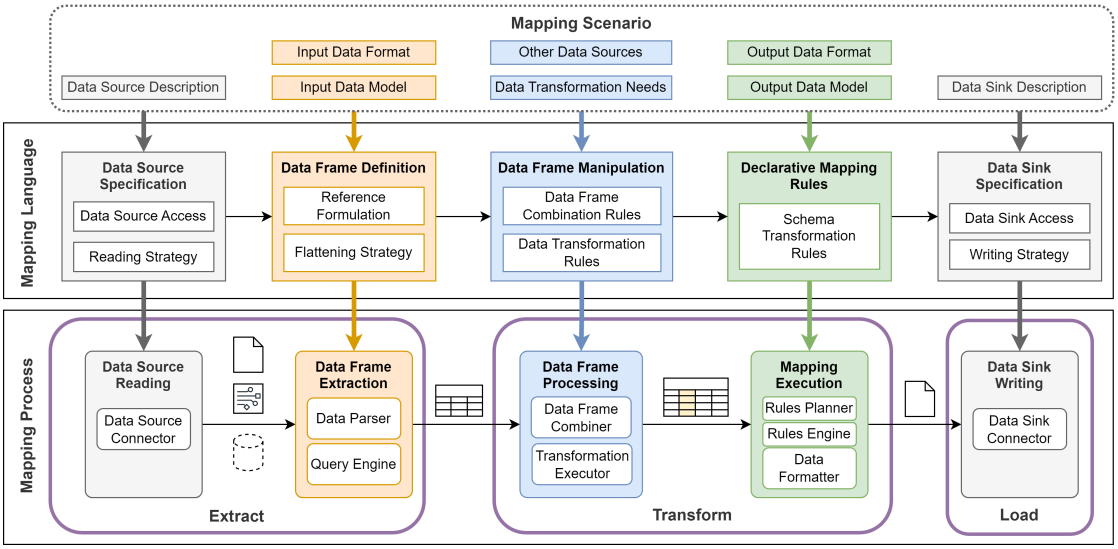

Figure 1: Overview of the proposed workflow for the generalisation of the declarative mapping process.

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 [10] 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 [37]. Similarly, the RML [8] specification implicitly defines a flattening strategy for hierarchical data sources through the `rml:iterator` and `rml:reference` operators. The approach proposed by RML Fields [26] results 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.

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 [29]. Using the FnO ontology [20] 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 [23]).

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 [38] 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 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 [31]. Finally, the result of the mapping process may be split considering different data sinks. The RML Logical Target[7] 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.

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[9] 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[10] (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[11] (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 integrated the tool within the Chimera[12] framework to support the declarative definition of composable semantic transformation pipelines leveraging MTL and the Apache Camel integration framework [39].

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

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 [40], JSON via JsonPath [41], 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.

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[13] identified in [19]. The list of requirements comprises the set of features made available by different declarative mapping languages (`cm-r*`) and the challenges identified by the community (`C*`). In the evaluation, we discuss example mapping templates targeting different mapping scenarios and how they demonstrate the fulfillment of requirements. For each template, the identifiers of the related requirements are mentioned. The complete set of generated templates

[13] https://github.com/oeg-upm/Conceptual-Mapping/tree/main/requirements

can be found in the tool repository[14] together with instructions on how to run them and the generated output. A table summarising the qualitative evaluation is also made available[15].

The coverage of basic requirements is demonstrated through the definition of templates for the examples available in the RML documentation[16]. These templates (`rml-csv`, `rml-xml`, `rml-json`) consider respectively a CSV, XML and JSON 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 2, the snippet (a) shows the mappings of the XML example for our tool, and the snippet (b) the corresponding ones in RML[17]. 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.

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 the user to define using VTL a custom strategy to iterate over the data frames and implement join operations. This approach addresses the mapping challenges related to join operations (`C5` and associated requirements `cm-r23`, `cm-r29`, `cm-r30`) asking the user to explicitly define how to access and process the extracted data for the definition of the output. 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 2, 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 syntax from the one used to generate plain RDF triples. 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

[14] https://github.com/cefriel/mapping-template/tree/main/examples
[15] https://github.com/cefriel/mapping-template-eval/blob/main/conceptual-mapping-reqs-eval.xlsx
[16] https://rml.io/specs/rml/
[17] The prefixes used in the snippets reported in the paper can be resolved by accessing the complete examples online.

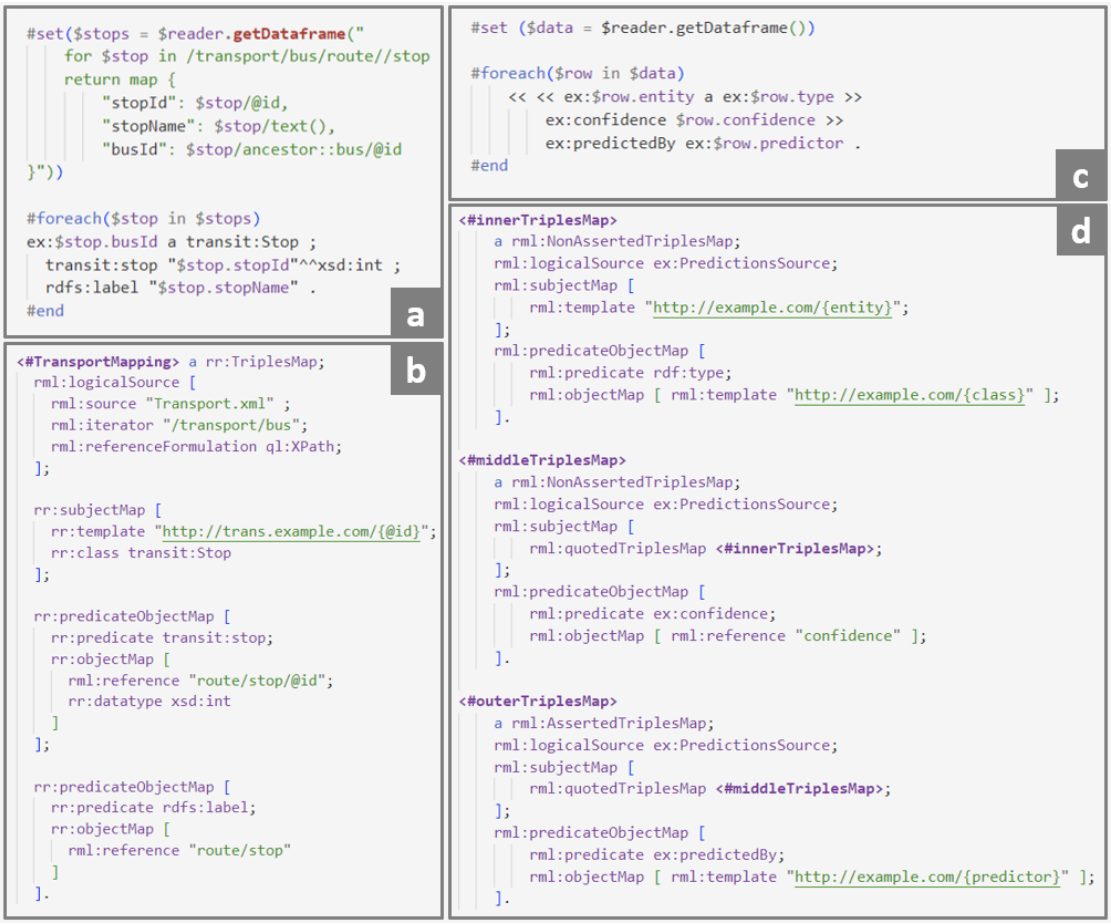

Figure 2: Example mapping templates: (a) conversion from XML to RDF, (c) conversion from CSV to RDF-star. Snippet (b) contains RML mappings equivalent to (a), and snippet (d) contains RML-star mappings equivalent to (c).

specify heterogeneous conversion pipelines as shown in the Chimera tutorial[18]. The definition of iterators in case of scenarios associated with complex nested data (`C2`, `cm-r14`, `cm-r27`) is delegated to the specific *Reader* but does not support the generation of nested data frames.

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[18].

The evaluation demonstrates how `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.

[18]https://github.com/cefriel/chimera-tutorial

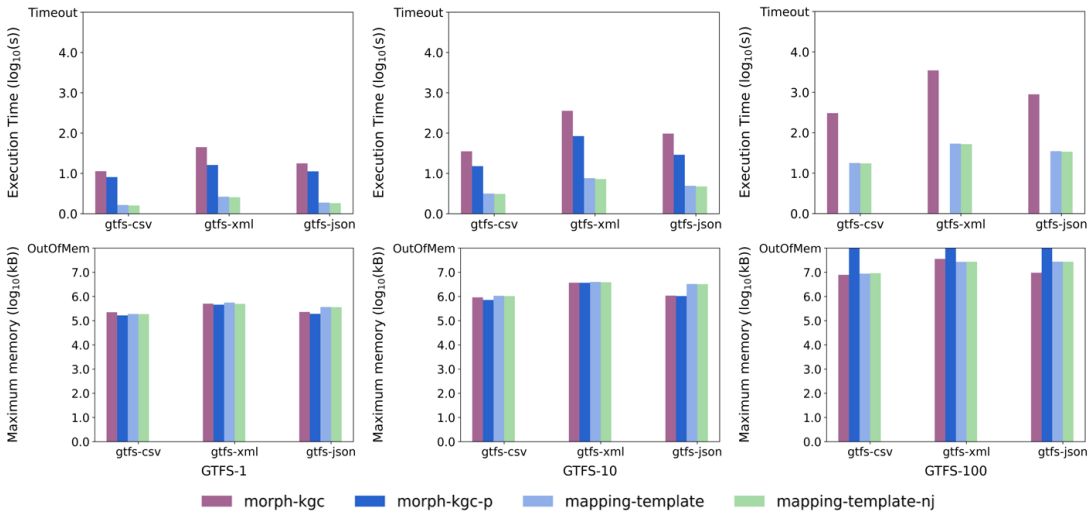

Figure 3: Results of the quantitative evaluation comparing metrics using the GTFS Madrid Benchmark.

6. Quantitative Evaluation

This section presents a quantitative evaluation[19] 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. To test the performance and scalability of the `mapping-template` tool, we considered the GTFS-Madrid-Bench [27]. 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 with the `morph-kgc` v2.3.1[20] processors. We adopted a set of RML mappings simplifying the join operation for the GTFS shapes file [3]. A set of templates implementing the same mapping rules were 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 selected `morph-kgc` for the evaluation considering its state-of-the-art performance and scalability results [32]. The `morph-kgc` processor was executed with parallel (`morph-kgc-p`) and sequential processing (`morph-kgc`). 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 3 reports the metrics registered for each configuration. Each test was executed three times, and the average metric is reported considering a logarithmic ($\log_{10}$) scale. The results

[19]Testing configuration and raw results at https://github.com/cefriel/mapping-template-eval
[20]https://github.com/oeg-upm/morph-kgc

show that the mapping-template tool completes the task with a lower execution time while registering similar memory consumption. 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 [42, 31], the difference in the metrics for the mapping-template tool was limited in the performed evaluation. The obtained results can be motivated considering 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, possibly reducing the *join* operations to be performed. 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 may be penalised by the fact that its implementation inherently guarantees the removal of duplicated triples before serializing the output.

To compare with additional RML processors and to investigate how the different parameters in the mappings (e.g., duplicates) affect the overall performance [43] of the mapping-template tool, we are currently working on a compiler from RML to MTL[21].

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 [39]. 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[22], 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 [6]. In the Horizon 2020 SPRINT project[23], the tool was used for the dynamic definition of converters for dataset conversion and service mediation [31] to support data exchanges between transportation operators. In the Horizon 2020 TANGENT project[24], the tool was used to implement the project solution for data harmonisation and fusion [44, 39]. 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

[21] https://github.com/cefriel/mapping-template/tree/feat-rml-compiler/rml
[22] https://snap-project.eu
[23] http://sprint-transport.eu/
[24] https://tangent-h2020.eu/

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 [45], 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.

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 [24]). 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[12] 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. As future work, we plan to perform a user study to compare the Mapping Template Language with other declarative mapping languages for RDF generation. Moreover, we are working to provide a solution to enable the execution of RML mappings via our tool to perform a more detailed performance and scalability evaluation and to spread its adoption among users preferring a fully declarative approach for mapping rules.

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

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

[3] J. Arenas-Guerrero, M. Scrocca, A. Iglesias-Molina, J. Toledo, L. Pozo-Gilo, D. Doña, Ó. Corcho, D. Chaves-Fraga, Knowledge graph construction with R2RML and RML: an ETL system-based overview, in: D. Chaves-Fraga, A. Dimou, P. Heyvaert, F. Priyatna, J. F. Sequeda (Eds.), Proceedings of the 2nd International Workshop on Knowledge Graph Construction co-located with 18th Extended Semantic Web Conference (ESWC 2021), Online, June 6, 2021, volume 2873 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2873/paper11.pdf.

[4] D. Chaves-Fraga, A. Dimou, A. Iglesias-Molina, U. Serles, 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, volume 3471 of *CEUR Workshop Proceedings*, CEUR, Hersonissos, Greece, 2023. URL: https://ceur-ws.org/Vol-3471/#preface, iSSN: 1613-0073.

[5] M. Bennara, A. Zimmermann, M. Lefrançois, 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.

[6] M. Scrocca, M. Comerio, A. Carenini, 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, volume 12507, Springer, 2020, pp. 411–429. doi:10.1007/978-3-030-62466-8_26.

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

[8] A. Dimou, M. V. Sande, P. Colpaert, R. Verborgh, E. Mannens, 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), volume 1184, CEUR-WS.org, 2014. URL: http://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf.

[9] A. Chortaras, G. Stamou, D2RML: integrating heterogeneous data and web services into custom RDF graphs, in: T. Berners-Lee, S. Capadisli, S. Dietze, A. Hogan, K. Janowicz, J. Lehmann (Eds.), Workshop on Linked Data on the Web co-located with The Web Conference 2018, LDOW@WWW 2018, Lyon, France April 23rd, 2018, volume 2073 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018. URL: http://ceur-ws.org/Vol-2073/article-07.pdf.

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

[11] C. Debruyne, D. O'Sullivan, R2RML-F: towards sharing and executing domain logic in R2RML mappings, in: S. Auer, T. Berners-Lee, C. Bizer, T. Heath (Eds.), Proceedings of the Workshop on Linked Data on the Web, LDOW 2016, co-located with 25th International World Wide Web Conference (WWW 2016), volume 1593 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2016. URL: http://ceur-ws.org/Vol-1593/article-13.pdf.

[12] F. Michel, L. Djimenou, C. Faron-Zucker, 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. URL: https://doi.org/10.5220/0005448304430454. doi:10.5220/0005448304430454.

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

[14] B. Vu, J. Pujara, 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, p. 189–196. URL: https://doi.org/10.1145/3360901.3364449. doi:10.1145/3360901.3364449.

[15] H. García-González, D. Fernández-Álvarez, J. E. L. Gayo, Shexml: An heterogeneous data mapping language based on shex, in: P. Cimiano, O. Corby (Eds.), 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, volume 2262 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018, pp. 9–12. URL: http://ceur-ws.org/Vol-2262/ekaw-poster-08.pdf.

[16] W. Akhtar, J. Kopecký, T. Krennwallner, A. Polleres, XSPARQL: traveling between the XML and RDF worlds - and avoiding the XSLT pilgrimage, in: S. Bechhofer, M. Hauswirth, J. Hoffmann, M. Koubarakis (Eds.), The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings, volume 5021 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 432–447. URL: https://doi.org/10.1007/978-3-540-68234-9_33. doi:10.1007/978-3-540-68234-9_33.

[17] L. Asprino, E. Daga, A. Gangemi, P. Mulholland, Knowledge graph construction with a

façade: A unified method to access heterogeneous data sources on the web, ACM Trans. Internet Technol. (2022). URL: https://doi.org/10.1145/3555312. doi:10.1145/3555312.

[18] M. Lefrançois, A. Zimmermann, N. Bakerally, Flexible RDF generation from RDF and heterogeneous data sources with sparql-generate, in: P. Ciancarini, F. Poggi, M. Horridge, J. Zhao, T. Groza, M. C. Suárez-Figueroa, M. d'Aquin, V. Presutti (Eds.), Knowledge Engineering and Knowledge Management - EKAW 2016 Satellite Events, EKM and Drift-an-LOD, Bologna, Italy, November 19-23, 2016, Revised Selected Papers, volume 10180 of *Lecture Notes in Computer Science*, Springer, 2016, pp. 131–135. URL: https://doi.org/10.1007/978-3-319-58694-6_16. doi:10.1007/978-3-319-58694-6_16.

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

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

[21] D. Van Assche, G. Haesendonck, G. De Mulder, T. Delva, P. Heyvaert, B. De Meester, A. Dimou, Leveraging web of things w3c recommendations for knowledge graphs generation, in: M. Brambilla, R. Chbeir, F. Frasincar, I. Manolescu (Eds.), Web Engineering, Springer International Publishing, Cham, 2021, pp. 337–352.

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

[23] J. Arenas-Guerrero, A. Alobaid, M. Navas-Loro, M. S. Pérez, 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.

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

[25] O. Hartig, et al., RDF-star and SPARQL-star, 2021. URL: https://w3c.github.io/rdf-star/cg-spec.

[26] T. Delva, D. V. Assche, P. Heyvaert, B. D. Meester, A. Dimou, Integrating Nested Data into Knowledge Graphs with RML Fields, in: D. Chaves-Fraga, A. Dimou, P. Heyvaert, F. Priyatna, J. Sequeda (Eds.), Proceedings of the 2nd International Workshop on Knowledge Graph Construction, volume 2873 of *CEUR Workshop Proceedings*, CEUR, Online, June, 2021. URL: https://ceur-ws.org/Vol-2873/#paper9, iSSN: 1613-0073.

[27] D. Chaves-Fraga, F. Priyatna, A. Cimmino, J. Toledo, E. Ruckhaus, 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.

[28] S. Jozashoori, D. Chaves-Fraga, E. Iglesias, M. Vidal, Ó. Corcho, Funmap: Efficient execution of functional mappings for knowledge graph creation, in: J. Z. Pan, V. A. M. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne, L. Kagal (Eds.), The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part I, volume 12506 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 276–293. doi:10.1007/978-3-030-62419-4_16.

[29] S. Jozashoori, E. Iglesias, M. Vidal, Dragoman: Efficiently evaluating declarative mapping languages over frameworks for knowledge graph creation, CoRR abs/2210.15645 (2022). doi:10.48550/arXiv.2210.15645. arXiv:2210.15645, (pre-print).

[30] E. Iglesias, S. Jozashoori, D. Chaves-Fraga, D. Collarana, M. Vidal, Sdm-rdfizer: An RML interpreter for the efficient creation of RDF knowledge graphs, in: M. d'Aquin, S. Dietze, C. Hauff, E. Curry, P. Cudré-Mauroux (Eds.), CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020, ACM, 2020, pp. 3039–3046. doi:10.1145/3340531.3412881.

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

[32] J. Arenas-Guerrero, D. Chaves-Fraga, J. Toledo, M. S. Pérez, O. Corcho, Morph-KGC: Scalable knowledge graph materialization with mapping partitions, Semantic Web Preprint (2022) 1–20. doi:10.3233/SW-223135, publisher: IOS Press.

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

[34] O. Corby, 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. doi:10.1007/978-3-319-30996-5_16.

[35] C. Allocca, 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.

[36] M. Hert, G. Reif, 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, p. 25–32. doi:10.1145/2063518.2063522.

[37] A. Makinouchi, A consideration on normal form of not-necessarily-normalized relation in the relational data model., in: VLDB, volume 1977, Citeseer, 1977, pp. 447–453.

[38] P. Heyvaert, B. De Meester, A. Dimou, R. Verborgh, Declarative rules for linked data generation at your fingertips!, in: European Semantic Web Conference, Springer, 2018, pp. 213–217.

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

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

[41] S. Gössner, G. Normington, C. Bormann, JSONPath: Query Expressions for JSON, RFC 9535, 2024. URL: https://www.rfc-editor.org/info/rfc9535. doi:10.17487/RFC9535.

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

[43] D. Chaves-Fraga, K. M. Endris, E. Iglesias, Ó. Corcho, 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.

[44] M. Comerio, A. Fiano, M. Grassi, 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).

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