RMLMapper supported by RML-view-to-CSV in the KGCW Challenge 2025

Els de Vleeschauwer[1], Dylan Van Assche[1] and Ben De Meester[1]

[1]*IDLab, Department of Electronics and Information Systems, Ghent University – imec, Technologiepark-Zwijnaarde 122, 9052 Ghent, Belgium*

Abstract

This paper presents the results of the Knowledge Graph Construction Workshop 2025 Challenge with RMLMapper, a reference in-memory JAVA implementation of the RML specification (RMLio) before RML got taken up by the W3C Knowledge Graph Construction (KGC) Community Group (RMLkgc). We updated RMLMapper to translate R2RML and RMLio into the latest version of RMLkgc. We participated in the challenge for the modules RML-Core, RML-IO and RML-LV. RMLMapper largely supports RML-Core and RML-IO (90% and 65%). For RML-LV, we added RML-view-to-CSV as a first step to our knowledge graph construction pipeline to materialize RML logical views (RML-LV) as CSV files. This way, RMLMapper largely supports RML-LV (81%). We currently cannot claim full coverage of the RMLkgc specification, however, this challenge enabled us to uncover the current gaps in our implementation, and allowed us to provide basic RMLkgc support in a mature JAVA implementation. Our continued participation in the community group will allow us to remain up to date with the latest developments in RMLkgc and further identify future implementation priorities for the RMLMapper.

Keywords

RMLMapper, RML-view-to-CSV, challenge, knowledge graph construction

1. Introduction

The third edition of the Knowledge Graph Construction Workshop (KGCW) challenge[1] is a next iteration of the KGCW2024 challenge's conformance track[2]. Its continued aim is to assess the compliance with the RML specification as established by the W3C Community Group on Knowledge Graph Construction (from hereon dubbed "RMLkgc") [1]. The RML specification is made modular and extends both the RDB to RDF Mapping Language (R2RML) specification [2] and the initial RML specification [3] (from hereon dubbed "RMLio").

RML's modules encompass (i) RML-Core, an iteration of the original R2RML functionality (exempt database connection and query functionality); (ii) RML-IO, an iteration of the RMLio data source reference functionality; (iii) RML-IO-Registry, a living document of reference formulations; (iv) RML-FNML, an iteration of the RMLio FnO integration for data transformation functionality; (v) RML-LV, a new module that provides an abstraction layer between RML-IO and RML-Core; (vi) RML-CC, a new module that provides support for RDF Collection and RDF Container generation; and (vii) RML-Star, a new module that provides support for RDF 1.2 triple term generation. The challenge contains test cases for each of these modules, to determine their feasibility and correctness by applying them in implementations.

In this paper, we present the results for RMLMapper, a reference in-memory JAVA implementation of the RMLio specification. Since our previous submission [4], we further updated RMLMapper to translate R2RML and RMLio into the latest version of RMLkgc, and combined it with RML-view-to-CSV [5], a

KGCW'25: 6th International Workshop on Knowledge Graph Construction, June 1st, 2025, Portorož, SLO

*Corresponding author.

†These authors contributed equally.

els.devleeschauwer@ugent.be (E. de Vleeschauwer); dylan.vanassche@ugent.be (D. V. Assche); ben.demeester@ugent.be (B. De Meester)

0000-0002-8630-3947 (E. de Vleeschauwer); 0000-0002-7195-9935 (D. V. Assche); 0000-0003-0248-0987 (B. De Meester)

[1]https://doi.org/10.5281/zenodo.14970817

[2]https://doi.org/10.5281/zenodo.10721875

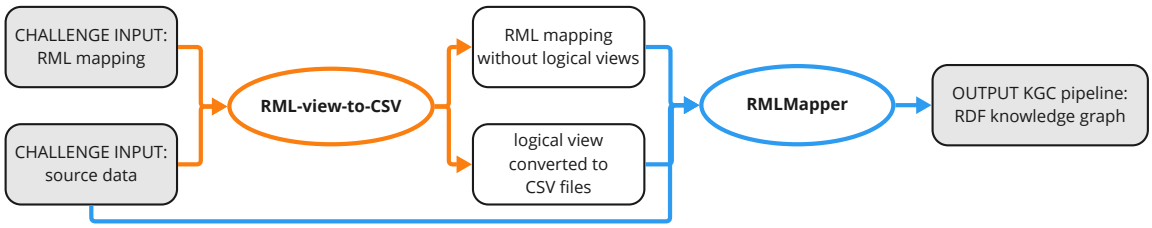

Figure 1: Knowledge graph construction pipeline: logical views are materialized as CSV files by RML-view-to-CSV.

preprocessing library for RML-LV that materializes Logical Views as CSV files and produces an adapted mapping document that can be processed by any RMLkgc engines supporting CSV files as source data format (Figure 1). With release v1.1.0[3], RML-view-to-CSV has been updated to support most features of the latest version of the RML-LV specification.

2. Results

We participated in the challenge for the modules RML-Core, RML-IO and RML-LV. We employed RMLMapper v7.3.3[4] and RML-view-to-CSV v1.1.0[3].

RMLMapper supports 53 out of 59 RML-Core test cases (89.83%). Failing cases are due to (i) using JSONPath expressions for keys that contain special characters (RMLTC010a-JSON, RMLTC0010b-JSON, RMLTC0010c-JSON), whereas the employed RMLMapper library does not follow these latest IETF draft specifications yet; (ii) no support for the shortcut property rml:subject in combination with a blank node (RMLTC0025a-JSON); (iii) being too lenient in wrongfully ignoring the rml:termType in a constant-valued term map (RMLTC0025b-JSON); and

Considering the failing of test case RMLTC0023e-JSON, this is deemed an incorrect test case, and discussion is ongoing[5].

RMLMapper supports 41 out of 73 RML-IO test cases (56.16%). Failing cases are due to (i) missing support for compressed data sources (RMLSTC0002b, RMLSTC0002c, RMLSTC0002d, RMLSTC0002e); (ii) missing support for logical source serialization n-triples (RMLSTC0003); (iii) incorrectly handling CSWV null values (RMLSTC0004a, RMLSTC0004c); (iv) testing implementation of RMLMapper misses support for templated values for SQL test cases (RMLSTC0006a); (v) missing support for `rml:CurrentWorkingDirectory` (RMLSTC0006b); (vi) missing support for XML reference formulation with namespaces (RMLSTC0007d); (vii) returning warnings instead of errors (RMLSTC0010a, RMLSTC0010b); (viii) missing support for targets in `rml:DatatypeMap` (RMLTTC0001f, RMLTTC0002q); and (ix) missing support for logical target serializations N3, RDF/JSON, and RDF/XML (RMLTTC0004b, RMLTTC0004e, RMLTTC0004f), UTF-16 encoding (RMLTTC0005b), and compression (RMLTTC0006b, RMLTTC0006c, RMLTTC0006d, RMLTTC0006e).UTF-16 encoding

Considering the failing of test cases RMLTTC0002f, RMLTTC0002g, RMLTTC0002h, RMLTTC0002i, RMLTTC0002k, RMLTTC0002l, RMLTTC0002m, RMLTTC0002n, RMLTTC0002r: mistakes in the test cases were discovered concerning correct graph assignment[6]. Considering the failing of test case RMLTTC0004g, a mistake in the test case was discovered concerning incorrect test data[7]. By fixing these mistakes[8], RMLMapper supports 51 out of 73 RML-IO test cases (69.86%).

RMLMapper in combination with RML-view-to-CSV supports 26 out of 32 RML-LV test cases (81.25%). Failing cases are due to (i) missing support for constant-valued and tempate-valued

[3]https://github.com/RMLio/rml-view-to-csv/releases/tag/v1.1.0
[4]https://github.com/RMLio/rmlmapper-java/releases/tag/v7.3.3
[5]https://github.com/kg-construct/rml-core/issues/203
[6]https://github.com/kg-construct/rml-io/issues/129, https://github.com/kg-construct/rml-io/issues/130, and https://github.com/kg-construct/rml-io/issues/131
[7]https://github.com/kg-construct/rml-io/issues/132
[8]https://github.com/kg-construct/rml-io/pull/133

`rml:ExpressionFields` in RML-view-to-CSVW (RMLLVTC0001b, RMLLVTC0001c); and (ii) missing support for natural data type mapping in RML-view-to-CSV (RMLLVTC0004a, RMLLVTC0004b, RMLLVTC0004c, RMLLVTC0004d).

RML-view-to-CSV can not support natural data type mapping because it materializes logical views as CSV files. CSV does not provide any native data types, therefore there is no natural RDF mapping of CSV values upon XSD data types[9]. A mistake in RML-LV test case RMLLVTC0008b was discovered and corrected[10].

3. Conclusion

We currently can not claim full coverage of the RMLkgc specification for modules RML-Core, RML-IO, or RML-LV, however, this challenge enabled us to uncover the current gaps in our implementation, and allowed us to provide basic RMLkgc support in a mature JAVA implementation. Finally, some errors in the test cases have been discovered. Our continued participation in the community group will allow us to remain up to date with the latest developments in RMLkgc and further identify future implementation priorities for the RMLMapper.

Acknowledgments

The described research activities were supported by SolidLab Vlaanderen (Flemish Government, EWI and RRF project VV023/10), and the European Union's Horizon Europe research and innovation program under grant agreement no. 101058682 (Onto-DESIDE).

The author(s) have not employed any Generative AI tools.

References

[1] 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: Proceedings of the International Semantic Web Conference (ISWC), Lecture Notes in Computer Science, Springer, Cham, 2023. doi:10.1007/978-3-031-47243-5_9.

[2] S. Das, S. Sundara, R. Cyganiak, R2RML: RDB to RDF Mapping Language, Working Group Recommendation, World Wide Web Consortium (W3C), 2012. URL: http://www.w3.org/TR/r2rml/.

[3] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, R. Van de Walle, RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data, in: Proceedings of the 7th Workshop on Linked Data on the Web, volume 1184, 2014.

[4] Van Assche, Dylan and Jankaj, Jozef and De Meester, Ben, Backwards or forwards? [R2]RML backwards compatibility in RMLMapper, in: KGCW 2024 : Knowledge Graph Construction 2024 : Proceedings of the 5th International Workshop on Knowledge Graph Construction co-located with 21th Extended Semantic Web Conference (ESWC 2024), volume 3718, 2024, p. 7. URL: https://ceur-ws.org/Vol-3718/.

[5] E. de Vleeschauwer, B. De Meester, 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 (KGCW 2024) co-located with 20th Extended Semantic Web Conference (ESWC 2024), 2024.

[9] ttps://kg-construct.github.io/rml-io-registry/csv/index.html#natural-rdf-mapping

[10] https://github.com/kg-construct/rml-lv/issues/72