RMLStreamer supported by RML-view-to-CSV in the performance track of the KGCW Challenge 2024

Els de Vleeschauwer[1], Ben De Meester[1]

[1]IDLab, Dept. Electronics & Information Systems, Ghent University – imec, Belgium

Abstract

This paper presents the results of the performance track of the Knowledge Graph Construction Workshop 2024 Challenge with RMLStreamer, an RML mapping engine that processes all data in a streaming fashion. On mappings without joins, RMLStreamer scales well regarding execution time and CPU usage, while maintaining a constant memory usage. To optimize the processing of the joins, we added RML-view-to-CSV as a first step to our knowledge graph construction pipeline. RML-view-to-CSV is a proof-of-concept implementation for RML Logical Views, i.e. flattened, source format-agnostic views over one or more existing data sources. RML-view-to-CSV can additionally rewrite referencing object maps as logical views, before it materializes the logical views as CSV files. The combination of RML-view-to-CSV and RMLStreamer emerges as an efficient approach, showcasing the potential of modular mapping engines that delegate each task to the most suitable framework.

Keywords

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

1. Introduction

The Knowledge Graph Construction Workshop (KGCW) 2024 Challenge[1] consists of two tracks: (i) a conformance track, that aims to spark development of implementations for the new RML specifications and improve the test-cases, and (2) a performance track, that wants to encourage the implementation of optimizations not only for execution time but also for CPU and memory usage. The conformance track covers the same experiments as the KGCW 2023 Challenge[2], consisting of two parts: (i) knowledge graph construction (KGC) parameters to evaluate individual parameters, e.g. joins and duplicates, with artificial data, and (ii) GTFS-Madrid-Bench [1] to focus on real-life use cases based on public transport data from Madrid. In contrast to the previous edition, all participants now conduct the experiments on identical virtual machines provided by Orange[3]. This ensures that the results from all participants can be directly compared.

KGCW'24: 5th International Workshop on Knowledge Graph Construction, May 26-27, 2024, Crete, GRE

els.devleeschauwer@ugent.be (E. de Vleeschauwer); ben.demeester@ugent.be (B. De Meester)

https://ben.de-meester.org/#me (B. De Meester)

0000-0002-8630-3947 (E. de Vleeschauwer); 0000-0003-0248-0987 (B. De Meester)

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

[2]https://doi.org/10.5281/zenodo.7689310

[3]https://hellofuture.orange.com/en/

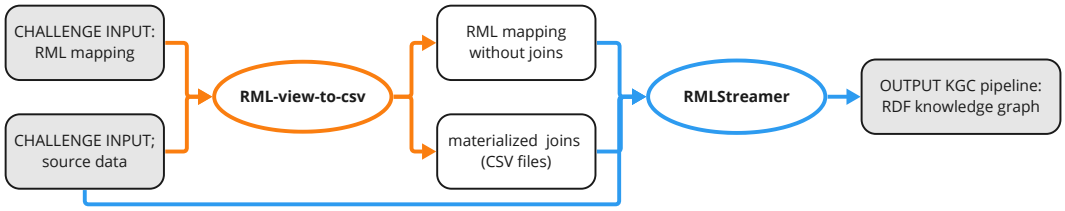


Figure 1: Knowledge graph construction pipeline: joins are resolved by RML-view-to-CSV to reduce the knowledge graph construction execution time and the size of the resulting RDF knowledge graph.

In this paper, we present the results of the performance track for RMLStreamer [2], an RML mapping engine which processes all data in a streaming fashion, in combination with RML-view-to-CSV[4], a proof-of-concept implementation for RML Logical Views [3].

Section 2 describes the components of our knowledge graph construction pipeline. Section 3 discusses the setup used to execute the challenge's experiments. We present our results in Section 4 and our conclusion in Section 5.

2. Knowledge Graph Construction Pipeline

Our knowledge graph construction pipeline (Figure 1) consists of two components: (i) RMLStreamer executes the RML mapping rules in a streaming fashion [2], and (ii) RML-view-to-CSV is a proof-of-concept implementation for RML Logical Views, that can resolve joins between data sources [3].

2.1. RMLStreamer

RMLStreamer executes RML mapping rules to generate high quality Linked Data from multiple originally (semi-)structured data sources in a streaming way. It handles big input files and continuous data streams like sensor data without consuming more memory when the input data size increases. RMLStreamer leverages Apache Flink to scale vertically across multiple CPU cores and horizontally across multiple machines. In the challenge we use RMLStreamer version v2.5.0 with an embedded Flink version in a Docker container[5].

The challenge results of 2023 show that, on mapping tasks without joins, RMLStreamer scales well regarding execution time and CPU usage, while maintaining a constant memory usage [4].

However, joins can significantly hinder the performance of RMLStreamer, as RML-Streamer does not eliminate self-joins or duplicates. As a result, RMLStreamer needs more than three hours to execute the first scale of the GTFS-Madrid-Bench, generating an output of 105 GB. Therefore, we delegate the execution of joins to RML-view-to-CSV.

[4]https://github.com/RMLio/rml-view-to-csv/
[5]https://zenodo.org/records/7998156

2.2. RML-view-to-CSV

RML-view-to CSV is a proof-of-concept implementation for RML Logical Views [3], a new RML module that is still under development. RML Logical Views[6] allow specifying a logical view: a flattened, source format-agnostic view over one or more existing data sources. A view over multiple data sources can be created by joining a logical view with other logical views.

First, RML-view-to-CSVÍidentifies and optimizes redundant self-joins, and eliminates the remaining referencing object maps from the mapping, replacing them by equivalent logical views. Afterwards, it materializes the logical views as CSV files. During this proces RML-view-to-CSV also takes the related triples maps into account, eliminating redundant fields and duplicate logical iterations. Finally, it rewrites the mapping accordingly replacing the logical views as logical sources over the materialized logical views.

At this moment RML-view-to-CSV supports one nested source format (JSON) and one tabular source format (CSV). Slight adaptations to the experiment setup were needed to overcome this limitation.

3. Experiment setup

The KGCW 2024 Challenge provides CSV files as source data, mapping files, queries, baseline results (i.e. the expected set of triples and query results), an example pipeline based on the MySQL, RMLMapper, and Virtuoso for reaching those results, and a tool for executing the example pipeline.

We made the following adaptations to the provided experiments to enable execution with our KGC pipeline. (i) In the provided end-to-end pipelines, the CSV files are loaded into a relational database. As RML-view-to-CSV does not support SQL (yet), we used the CSV files directly to construct the knowledge graphs and adapted the mapping files accordingly. (iii) As RML-view-to-CSV does not support XML (yet), we replaced the XML files in the GTFS-Madrid-Bench heterogeneity experiments by JSON files. We conducted two heterogeneity experiments: one experiment with only JSON source data and one experiment with 50 % CSV and 50 % JSON source data, both on scale 100. (ii) We added a condition to the mapping files to recognize the string `NULL` in the provided CSV files as an empty value. The GTFS experiments were executed with our KGC pipeline as shown in Figure 1. For the KGC parameters experiments without joins, we skipped the preprocessing step with RML-view-to-CSV, as it is a redundant step for these experiments. For the KGC parameter experiments with joins, we tested both the pipeline with and without RML-view-to-CSV as preprocessor, to measure also the impact of RML-view-to-CSV for those experiments.

We compared our experiments' results to ensure that our output is correct with respect to the baseline results of the challenge. For the first part of the challenge (KGC parameters), where the output of RMLStreamer is not loaded into a triples store, we deduplicated the output results as RMLStreamer cannot eliminate duplicates by itself.

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

After deduplication, we compared the number of triples to the baseline results of the challenge. For the second part of the challenge (GTFS-Madrid-Bench) we compared the number of query results to the baseline.

We executed all experiments on the virtual machine provided by Orange, with following specifications: 4 vCPUs, 16 GB RAM and 140GiB SSD storage running on Ubuntu 22.04.3 LTS. The challenge execution tool configures the Java heap space to 50 % of the available memory. All experiments were performed 5 times, and the experiment with the median of the measurements is reported. All files needed to reproduce the conducted experiments are available on Zenodo[7].

4. Results

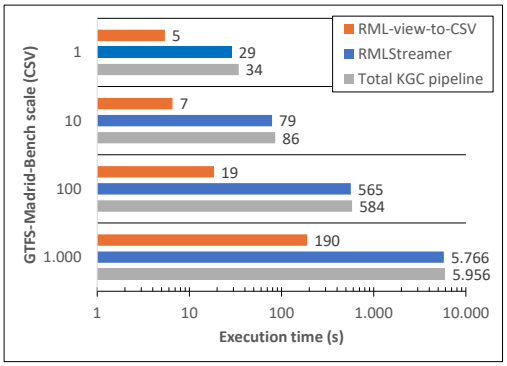
In Figure 2 and Table 1 we included the measured execution time, CPU time, and maximal memory usage of the knowledge graph construction pipeline for selected experiments. The complete overview of the results, as it was submitted to the KGCW 2024 Challenge, is available on Zenodo[8].

First, we verify how RMLStreamer behaves when the size of the expected output increases. This is best illustrated by the GTFS-Madrid-Bench scale experiments. Figure 2a and Figure 2b show that both RML-view-to-CSV and RMLStreamer scale towards a linear trend. Note that the reported metrics include the startup time of RML-view-to-CSV and RMLStreamer (no separate measurements in the challenge execution tool), that is independent of data size and has a higher impact on the lower scales. The execution time and CPU time increases with the same factor as the data size for higher scales. The peak RAM memory (fig. 2c) measured is similar for all scales when using RMLStreamer. RMLStreamer has a constant memory usage independent of the data size, because it processes everything in a streaming way. This ensures a stable performance independent of the data size. As long as there is space to store the output, RMLStreamer can continue its knowledge graph construction process. For RML-view-to-CSV we note an increasing use of memory for the higher GTFS scales. Nevertheless, it was still able to handle scale 1000 without reaching the memory limitations of the provided hardware.

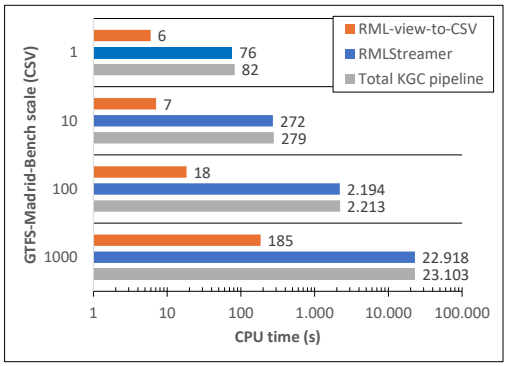
The measurements for the KGC parameter experiments confirm these observations (Table 1 section 1). RMLStreamer shows linear scaling of execution time and CPU usage, proportional to the size of the input data, in combination with a constant memory usage.

Second, we evaluate the impact of the format of the data input. Replacing the CSV source data by JSON data increases the execution time of RMLStreamer with a factor of two. The difference in execution time for RMLStreamer is the consequence of RMLStreamer chunking CSV files and processing the chunks in parallel. This is not the case for the JSON formats yet. The performance impact of nested data is higher for RML-view-to-CSV. The execution time and CPU usage of RML-view-to-CSV increases with a factor of ten, and its memory consumption with a factor of three (fig. 2d).

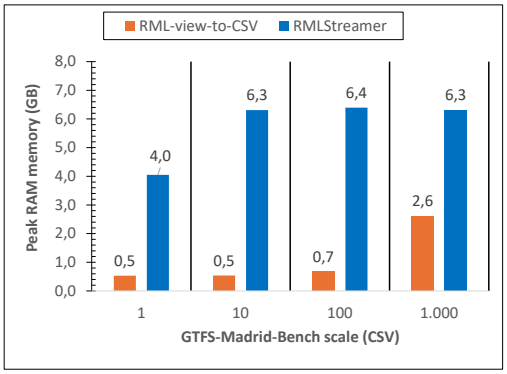
[7]https://doi.org/10.5281/zenodo.11100801
[8]https://doi.org/10.5281/zenodo.11100801



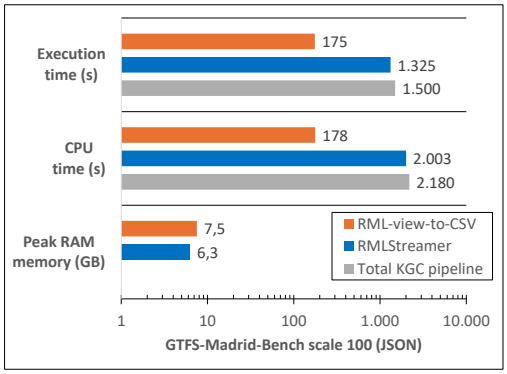
(a) Linear trend: the execution time of both RML-view-to-CSV and RMLStreamer increases with the same factor as the data size for higher scales.



(b) Linear trend: the CPU usage of both RML-view-to-CSV and RMLStreamer increases with the same factor as the data size for higher scales.



(c) The memory consumption of RML-view-to-CSV increases with RMLStreamer has a constant memory usage independent of data size.



(d) When using JSON data as source instead of CSV data, the performance impact is higher on RML-view-to-CSV than on RMLStreamer.

Figure 2: Metrics of the knowledge graph construction pipeline for the GTFS-Madrid-Bench experiments.

Third, we investigate the impact of duplicates and empty values in the input data. As RMLStreamer does not eliminate duplicates and the duplicate tests all start with the same amount of input data, there is no noticeable difference in performance between the experiments with and without duplicates, whilst a duplication elimination could result in a much better performance for tests with duplicates (Table 1 section 2). In the source data of the experiments with empty values (Table 1 section 3), this string `NULL` is representing an empty value in a CSV file. We had to add a condition to the mappings of those experiments to recognise this string as an empty values. The execution of this condition increased the execution time and CPU usage with 50% when all rows contain empty values.

Last, we comment on the KGC parameter experiments including joins (Table 1 section 4). We executed these experiments without and with RML-view-to-CSV as preprocessor

	Execution (s)	CPU (s)	Peak RAM (GB)	Output (triples)
1. Records				
10K rows 20 columns	22	49	1,8	200.000
100K rows 20 columns	41	120	6,1	2.000.000
1M rows 20 columns	187	694	6,1	20.000.000
10M rows 20 columns	1.769	6.918	6,2	200.000.000
2. Duplicates				
100 percent	43	128	6,1	20
0 percent	44	130	6,1	2.000.000
3. Empty values				
100 percent	66	223	6,1	0
0 percent	46	147	6,1	2.000.000
4a. Joins (without support of RML-view-to-CSV)				
1-1 0 percent	41	120	6.3	0
5-5 100 percent	110	396	6.4	2.500.000
4b. Joins (with support of RML-view-to-CSV)				
1-1 0 percent	37	88	6,1	0
5-5 100 percent	88	242	6,1	2.500.000

Table 1
Metrics of the knowledge graph construction step for selected KGC parameter experiments

and noticed that RML-view-to-CSV reduced the total execution time and CPU usage with respectively 20% and 39% for the experiments containing most joins.

5. Conclusion

The KGCW 2024 Challenge results confirms the observations of the KGCW 2023 Challenge: RMLStreamer has a linear scaling of execution time and CPU usage, proportional to the size of the input data, while maintaining a constant memory usage. Scalability is the main strength of RMLStreamer.

The main weakness of RMLStreamer is its inefficient implementation of join operations (e.g. GTFS-Madrid-Bench experiments with joins cannot be handled properly by RMLStreamer). When delegating this task to RML-view-to-CSV as a preprocessor, we resolve this weakness and build a reliable and performing knowledge graph construction pipeline.

The challenge results reveal of slower performance of RML-view-to-CSV when handling nested data. Boosting performance was not the aim of this proof-of-concept implementation. As RML Logical Views enable the flattening of nested data, the implementation of RML-view-to-CSV is processing all JSON fields separately. A more efficient implementation for nested data sources is a challenge for future implementations, after the RML Logical View specification is finalized.

Comparing to the results of KGCW Challenge 2023 [4], we see a direct relation between the performance of RMLStreamer, and the available CPU cores and RAM memory of the virtual machines used for the experiments. RMLStreamer is now up to a factor of three

slower and uses 30% less memory. The virtual machine used for the KGCW Challenge 2023 had 12 CPU cores and 24 GB RAM; for the KGCW Challenge 2024, all experiments were conducted on a virtual machine with 4 CPU cores and 16 GB RAM. We concluded that RMLStreamer takes full advantage of the number of available CPU cores and of the available memory.

At the moment of writing, we have no insight in the results of the other engines participating in the performance track of the KGCW Challenge 2024. We are looking forward to the comparison of the challenge results.

Acknowledgments

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

References

[1] 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.

[2] Sitt Min Oo, G. Haesendonck, B. De Meester, A. Dimou, RMLStreamer-SISO: An RDF Stream Generator from Streaming Heterogeneous Data, in: U. Sattler, A. Hogan, M. Keet, V. Presutti, J. P. A. Almeida, H. Takeda, P. Monnin, G. Pirrò, C. d'Amato (Eds.), The Semantic Web – ISWC 2022, Springer, Springer International Publishing, Cham, 2022, pp. 697–713. doi:10.1007/978-3-031-19433-7_40.

[3] 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.

[4] E. de Vleeschauwer, G. Haesendonck, D. Van Assche, B. D. Meester, B. De Meester, RMLStreamer with Reference Conditions in the KGCW Challenge 2023, in: Proceedings of the 4rd International Workshop on Knowledge Graph Construction (KGCW 2023) co-located with 20th Extended Semantic Web Conference (ESWC 2023), 2023.