Semantic Conversion of Transport Data Adopting Declarative Mappings: an Evaluation of Performance and Scalability

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

Cefriel, Milan, Italy name.surname@cefriel.com

Abstract. The transportation domain is characterised by a multitude of different formats to represent data, thus creating a problem of (lack of) interoperability between systems and a need for data conversion. In order to cope with the specific requirements of production systems, special attention should be given to performance and scalability of conversion solutions. In this paper, we present a thorough evaluation of the Chimera framework for semantic data conversion through declarative mappings, in both a dataset and message conversion scenarios. We illustrate the experimental results and we offer our considerations and recommendations for the successful implementation of conversion pipelines exploiting Semantic Web technologies.

Keywords: Transport Data · Semantic Data Conversion · Knowledge Graph Construction

1 Introduction

Interoperability in the transportation domain is the main challenge to provide travellers with multi-modal door-to-door travel solutions involving different transport service providers. The Shift2Rail initiative, financed by the European Commission, has been addressing this challenge within the Innovation Program IP4[1] by defining an *Interoperability Framework* to enable a seamless data exchange between different transport stakeholders [14]. The proposed approach exploits a reference ontology, representing the shared conceptual model of the transport domain, to enable interoperability at a semantic level and any-to-any communications between different actors. In this scenario, each stakeholder is not forced to adopt a new format or standard and can enter the ecosystem by defining a set of rules that specify how the currently used data model can be mapped onto the reference ontology [8]. As a result, a technical artifact, henceforth referred as the *converter*, can be configured to translate messages from a standard A to a standard B, exploiting Semantic Web technologies and the reference ontology.

The main challenges in the adoption of this approach are the definition of the mapping rules and the performance and scalability requirements related to

[1] Cf. https://shift2rail.org/research-development/ip4/

the technical implementation of converters. In the SPRINT project (Semantics for PerfoRmant and scalable INteroperability of multimodal Transport) we addressed these two aspects [11,15]. This work reports the results obtained in developing a semantic converter and in assessing its performances and scalability.

The performance and scalability evaluation of a converter should consider the two main conversion scenarios in the transportation domain: the harmonization of static data, like timetables and scheduled transport, and the transformation of dynamic data, like journey planning messages. The *Dataset Conversion* (batch conversion) scenario considers the case where medium-sized to big archives of transportation data, usually static data, should be converted. Conversions are required with low frequency, but the conversion procedure should minimise the resource usage to obtain scalability with respect to the size of the dataset. The *Message Conversion* (service mediation) scenario considers the case where a message, usually dynamic data, should be converted to guarantee communication between two different systems at runtime. A small amount of data is converted for each request, but the conversion procedure should minimise the processing time to avoid introducing overhead and to obtain scalability with respect to a high frequency request rate.

To address these two scenarios, we designed and developed a generic, modular and flexible solution, the Chimera framework[2] [15], the converter component of the *Interoperability Framework*. The performed testing activities reported in this work, besides offering insights on the Chimera implementation, contribute to a general validation of the involved technologies and tools for the discussed use case in the transportation domain.

The reminder of the paper is organised as follows: Section 2 deals with preliminaries and related works; Section 3 describes the testing activities designed to evaluate converters; Section 4 discusses the results obtained and elicits a set of recommendations for the performance and scalability of converters; Section 5 draws the conclusions and defines potential future works.

2 Preliminaries and Related Works

A semantic data conversion procedure, following the any-to-one centralized mapping approach [16], transforms data in two steps exploiting the defined rules to/from the reference ontology: (i) input data are mapped onto the reference ontology (*lifting phase* from standard A to RDF), (ii) the obtained triples, specifying data through the reference ontology, are mapped onto the target data format (*lowering phase* from RDF to standard B).

A possible implementation of the described conversion procedure is based on a Object-Relational Mapping (ORM) approach using unmarshalling/marshalling libraries to obtain an in-memory representation of data as objects, and then exploiting annotations in the code to map each class and attribute to class and properties of the reference ontology. This approach is implemented in RDF-

[2] `https://github.com/cefriel/chimera`

Beans[3], Empire[4], and was studied for the definition of the ST4RT converter. This implementation improved the pre-existing approaches providing more flexibility in the definition of the annotations to address complex mappings in both the lifting and the lowering phase [3]. However, while this method allows using classical object-oriented programming techniques, its application showcased several drawbacks: (i) an object-oriented representation of the source/target data format is required, (ii) related performance and scalability issues arise for conversion procedure with complex annotations and/or handling large files. For these reasons, in the SPRINT project, we implemented and tested an alternative converter relying on declarative mappings for the lifting and lowering phases.

Chimera is an open-source framework based on Apache Camel[5] and adopts a modular approach to build flexible pipelines for conversions based on Semantic Web technologies. The main goal of Chimera is to minimise the amount of code to be written, allowing to create a converter just by configuring the various blocks offered by the framework. In particular, this paper evaluates conversion procedures adopting the following Chimera blocks: (i) a lifting block for materialization through a custom version of the RML-Mapper library[6] and employing RML [10] mappings to generate RDF triples, and (ii) a lowering block based on Apache Velocity templates to query the RDF graph with SPARQL queries and to define the logic to place the query results in the proper output format[7].

RML is a mapping language that extends the R2RML recommendation [9] to support heterogeneous data sources. RML mapping rules are defined on a set of *logical sources* representing the input data sources; each rule is represented using a *triple map* that defines how to extract a *record* from the *logical source* and generate a set of associated RDF triples; a *join* condition allows to create triples involving entities generated by two *triple maps*.

A real-world use case and evaluation of this approach for batch data conversion in the transportation domain is presented in [15] exploiting an initial release of Chimera. This paper discusses the results of a broader evaluation of the converter: we adopt a transportation domain benchmark for the batch data conversion scenario [6], discuss how different configurations and parameters [5] can affect the performances and scalability of the conversion, and evaluate the converter also in the service mediation scenario.

A comparison of Chimera with others state-of-the-art tools for knowledge graph materialisation is available in [1]. The advantages of adopting a semantic data conversion procedure based on lifting and lowering mechanisms in a different domain, i.e. the Web of Things, is presented in [2].

---

[3] RDFBeans, cf. `https://github.com/cyberborean/rdfbeans`
[4] Empire, cf. `https://github.com/mhgrove/Empire`
[5] `https://camel.apache.org/`
[6] `https://github.com/RMLio/rmlmapper-java`
[7] `https://github.com/cefriel/rdf-lowerer`. A demo example of the lowering approach is available in the repository.

3 Test Design

In this section we present the performance and scalability evaluation, for both dataset and message conversion, and the testing infrastructure.

Dataset Conversion For the evaluation in the dataset conversion scenario, we chose the GTFS-Madrid-Bench[8] [6]. The benchmark is based on GTFS[9] (General Transit Feed Specification) data on the Madrid city metro published from the Consorcio Regional de Transporte de Madrid (CRTM[10]). GTFS is composed of a set of CSV files where each one represents some information about static transit information. Based on the original data and according to the benchmark, we generated datasets of increasing scale (1, 5, 10, 50, and 100), and in different formats (CVS, JSON and XML). It is important to point out that we generated those datasets for performance testing, but a typical GTFS feed size is in the order of tens of megabytes and rarely overcomes the 100 MB when unzipped.

The planned testing activities considered a roundtrip conversion GTFS → Linked GTFS [7] → GTFS. The GTFS-to-LinkedGTFS RML lifting mappings are from the GTFS-Madrid-Bench. For the lowering, we defined a set of custom Apache Velocity templates. The lifting phase considers different input formats (CSV, JSON or XML), while the lowering phase always produces CSV files. In the conversion pipeline we used an in-memory RDF repository to store and query the materialised RDF graph. However, we also evaluated the impact on performances of using an external triplestore.

Message Conversion To evaluate the message conversion, we selected a realistic journey planning test case involving a deployed web service, converting a response message from the HaCon VBB journey planning endpoint[11] to the TRIAS format[12]. During the lifting, a VBB *TripList* message (representing travel solutions for a requested itinerary, dimension 43KB) is mapped onto the IP4 IT2Rail ontology[13] through RML mappings. The materialised graph is stored through an in-memory RDF repository. During the lowering, the data modelled through the IT2Rail ontology are mapped onto a TRIAS *TripResponse* message using a Apache Velocity template with specific SPARQL queries. We employed the JMeter tool[14] to test the performances of the converter with a increasing size of concurrent requests (number of threads: 10, 50, 100, 150, 500, 1000, 2500, 5000; ramp-up period: 1 second; loop count: 1).

[8] `https://github.com/oeg-upm/gtfs-bench`
[9] `https://developers.google.com/transit/gtfs`
[10] `https://www.crtm.es`
[11] `http://fahrinfo.vbb.de`
[12] `https://github.com/VDVde/TRIAS`
[13] `http://www.it2rail.eu/`
[14] `https://jmeter.apache.org/`

Testing infrastructure All tests were run using Docker[15] Containers to guarantee reproducibility on a machine running CentOS Linux 7, with Intel Xeon 8-core CPU and 64 GB Memory. We set a memory limit to 24GB, a timeout of 24 hours and no limits on CPU usage. We run each test 5 times averaging the obtained results. We also monitored resource usage of containers in execution.

4 Test Evaluation

In this section, we discuss the performance and scalability test results and illustrate the identified bottlenecks and their possible solutions. Additional details and data are available in [4]. We compare the conversion results obtained with two different releases of Chimera, *core* and *final*. The *final* release implements a set of optimisations, discussed in this section, that were developed within the SPRINT project to improve the performance and scalability of the lifting and lowering components.

4.1 Dataset Conversion: Performance and Scalability

In Table 1, we provide the complete results for the dataset conversion scenario considering different sizes and different formats. Execution times are measured in seconds and averaged on the 5 runs of each conversion; *TO* stands for timeout (>24-hours), *OoM* stands for out-of-memory (>24GB). We also report the input size and the number of triples generated at the end of the lifting phase.

We also tested different configurations of the pipeline (not fully reported here). The results in Table 1 refer to the best-performing configuration for each format and size. As preliminary comment, we notice that the overall conversion time is mainly influenced by the lifting phase (as also shown later in Table 2).

Scale	**1**		**5**		**10**		**50**		**100**	
Input Size	4.9 MB		10.42 MB		23.64 MB		106.1 MB		247.5 MB	
Num. Triples	565,489		1,800,911		3,663,380		18,009,100		36,633,800	
Release	core	final	core	final	core	final	core	final	core	final
CSV	22.77	10.83	164.95	55.37	544.41	154.13	11,624.45	3,441.67	TO	OoM
JSON	50.41	30.89	659.11	394.21	2,471.29	1,467.70	66,003.36	34,901.65	TO	TO
XML	TO	16.26	TO	123.29	TO	434.05	TO	12,648.65	TO	OoM

Table 1: Dataset conversion execution times (in seconds) for 1,5,10,50,100-scale size GTFS dataset, comparing different formats and Chimera releases.

The results show a consistent performance improvement in the conversion time of the *final* release with respect to the *core* one, respectively for CSV ($2x$), JSON ($1.6x$) and XML ($> 1,000x$) data sources. The *final* version was able to convert CSV, JSON and XML datasets up to 100 MB and generating 18 M triples with the available resources, thus demonstrating its capability to process even large dataset of static transportation data.

[15] https://www.docker.com/

The *final* version mainly improved with respect to the lifting phase, due to the adoption of different libraries to access the input data sources, a simplified mechanism to handle the generated triples (triples are directly written to the RDF repository of the Chimera pipeline during the lifting procedure) and the introduction of different concurrency strategies in the RML block (at the *record* level and/or *triple map* level). However, concurrency naturally increases CPU usage and memory consumption, thus, in some cases, it may be preferable to use single-threading, with a longer conversion time but lower resource usage.

Additional tests, available in [4], show how concurrency in the lifting process performs better at the *triple map* level if *triple map*s associated with the same logical source are performed within the same thread (avoiding different threads to concurrently access the same input source). However, different RML mappings may result in different performance results (e.g., number of *triple map*s defined for each logical source, join conditions among them) considering the same concurrency configuration.

CSV conversion time outperforms the JSON/XML one because of the impact of the libraries to access the input datasources in the lifting procedure. Also intuitively, accessing rows in a CSV file and iterating over them is less expensive than querying a JSON/XML file resolving a JSONPath/XPath query and iterating over nodes retrieved. This aspect affects the execution and worsens with the size of the file to query, hence the differences in timings. Considering the XML data format, the *core* version did not complete the conversion within the 24 hours timeout even for the 1-xml dataset, while the *final* converter completed it up to the 50-scale size, performing even better than the JSON conversion thanks to the new XML parser. The conversion of the JSON dataset obtained more limited improvements between the *core* and *final* version, mainly because the performances of the JSON parser limit the advantages of introducing concurrency. The results obtained for JSON and XML show not only the importance of lifting mapping optimization via concurrency, but also the impact on performances of the parsing procedure.

Finally, we checked the impact of *join conditions* in the RML mappings on the conversion time. Mappings defined in the GTFS-Madrid-Bench maximize the number of joins among the different files composing a GTFS feed. As in SQL queries, a growing number of items (in this case, the scale of the input dataset) increases non linearly the number of needed comparisons for a non-optimised join operation. In the RML mappings it is often possible to avoid the usage of join conditions by adopting the same IRI generation pattern in different *triple map*s. With this approach, we managed to optimize the RML mappings for the GTFS-Madrid-Bench dataset. Two RML mappings producing exactly the same knowledge graph were used to compare conversion times in Chimera with and without joins conditions. The results showed that the optimized mappings reduced the conversion time up to 2/3 in the case of 50-scale CSV (6205.25s with joins, 2269.62s without joins).

Table 2 compares, for the 50-scale CSV dataset, the results obtained considering an in-memory and an external RDF repository (triplestore). It is important

	Conversion time (s)	Lifting time (s)	Lowering time (s)	Max Mem (GB)	Max CPU Usage (%)
core-csv	11,624.45	11,583.49	40.97	18.84	185.56
final-csv	3,441.67	3,407.39	34.28	18.58	516.51
final-csv-ext	3,784.34	3,659.39	88.95	9.63	314.61
final-json	34,901.65	34,861.97	39.69	18.45	153.97
final-xml	12,648.65	12,614.62	34.03	18.44	540.01

Table 2: Dataset conversion scenario: 50-scale size GTFS dataset detailed results comparing different configurations and formats.

to point out that the performance strictly depends also on the employed triplestore, in this case *GraphDB Free v9.0.0*[16]. On one hand, the usage of an external repository with incremental writes (*final-csv-ext*) drastically reduces the memory consumption ($2x$ reduction) with respect to the same configuration run using an in-memory repository (*final-csv*). The conversion time using incremental writes is higher, but it can be acceptable because it comes with a substantial decrease in resource usage; the adoption of a triplestore without concurrency limitations[17] would likely bring an even better time-resource trade off. In any case, it is important to take into account that an external repository implies also its own resource usage.

Table 2 also details the lifting and lowering time considering the different formats for the 50-scale dataset in the *final* release. As previously commented, lifting times are influenced by the input data format whereas the lowering times are similar since the same lowering mappings are executed on the same knowledge graph. In general, our results show that lifting through materialization is a time-consuming approach in case of large datasets. However, it is important to highlight that the really good lowering performance is mainly due to the possibility of querying a materialized knowledge graph. The lowering time is higher when using an external repository (*final-csv-ext*) due to the concurrency limitations for the SPARQL queries in the template. However, complex queries in the templates applied to large knowledge graphs can effectively benefit of an external triplestore [15]. For example, when lifting and lowering steps have similar execution times, it may be beneficial to increase a little bit the lifting time writing triples to an external repository, to speed up the lowering phase thanks to the reading performances of the triplestore.

4.2 Message Conversion: Performance and Scalability

In Table 3 we report the results of the tests performed for the message conversion scenario. For the *final* release, we compare two configurations: *final-m-1* introducing concurrency in lifting only at the *record* level, and *final-m-2* introducing it also at the *triple map* level. The best performing configuration (*final-m-1*), resulted in a conversion times in the order of 100ms, thus perfectly acceptable

[16] `https://graphdb.ontotext.com/`
[17] Free version of GraphDB is limited to two concurrent queries.

	Conversion time (ms)	Lifting time (ms)	Lowering time (ms)	Max Mem (GB)	Max CPU Usage (%)	Num. Triples
core-m	739	711	28	0.09	40	466
final-m-1	138	107	31	0.04	25	466
final-m-2	166	125	41	0.04	30	466

Table 3: Message conversion scenario: VBB-TRIAS conversion comparing different configurations.

Number of concurrent requests	**Avg Time of processing N Requests** [ms]	**Interval between requests** [ms]
10	131	100
50	219	20
100	775	10
150	1,663	6.7
500	3,926	2
1,000	7,567	1
2,500	21,114	0.4

Table 4: Message conversion scenario: VBB-TRIAS conversion with increasing number of concurrent requests.

in a runtime scenario. Moreover, a $5x$ improvement in the conversion time was obtained from the *core* to the *final* release, thanks to the already mentioned optimisations, but also thanks to a specific configuration for the message conversion scenario that executes the lowering transformation avoiding input/output operations on the filesystem. The results obtained for *final-m-2*, demonstrate that for small messages it is preferable not to introduce excessive concurrency, since the structure initialization time (e.g., threads) is not compensated by an overall speedup. Finally, it is worth noting the very limited resource usage, especially memory, in the different tested configurations.

Table 4 reports the scalability test results with an increasing number of concurrent requests. A single instance of the converter managed to handle up to 100 concurrent requests per second (at 150 concurrent requests the processing time overcomes one second), handling successfully also a peak of 2,500 concurrent requests. After 3,000 pending requests the queue mechanism provided by Apache Camel starts dropping requests. The maximum length of the queue can be increased, however, a high number of pending requests is associated with noticeable performance degradation. The low resource usage and the optimizations resulted in very good scalability results for increasing workloads, even considering a single instance of the converter.

4.3 Recommendations for Performance and Scalability

The reported results show the improvements obtained in the development of the Chimera framework for semantic data conversion pipelines and its applicability to both the dataset and message conversion scenarios in the transportation domain. Moreover, from the performed testing activities, we derive a set of ad-

ditional recommendations for performance and scalability using the proposed approach.

Performance Considering the RML-based *lifting*, the specific RML mappings defined for a conversion pipeline (join conditions, number of triple maps, number of logical sources, path to access the records, ...) can influence the performances of the lifting portion. As a result, the choice of the pipeline configuration should take into account the trade-off between the conversion time and the resource usage, for example with different concurrency strategies. In particular, in some cases the gain in conversion time obtained does not justify the higher resource usage. Additional recommendations for the RML-based lifting are:

- to efficiently exploit concurrency, it is important to tune the different parameters, e.g., the number of concurrent threads adopted;
- concurrency should not be used in case of mappings defining blank nodes that are referenced by different triple maps (each thread would assign different random identifiers to the same blank node);
- the presence of many functions in the RML mappings (RML and FnO integration [13]) can cause concurrent access to the same data structures negatively impacting the conversion time;
- concurrency strategies can be implemented not only within the lifting procedure but also in the pipeline, for example, configuring different lifting blocks in parallel or exploiting concurrent consumers for routes in Chimera.

With respect to the *lowering* phase, the queries and the logic of the templates can heavily affect performance. Therefore, it is recommended to:

- *Optimize queries*, by accessing data using simple queries and avoiding expensive constructs or patterns. It is better to divide complex queries into sub-queries, if possible.
- *Optimize template logic*, by avoid nested loops and by using support data structures (e.g. maps) to efficiently access records in queries' result sets.

Finally, the stream option to process templates in-memory (avoiding input/output operations) is recommended only for the runtime data/message conversion use case. For large batch datasets, this option should be avoided, because the template engine is able to optimize memory consumption with incremental writes to the filesystem.

Scalability In the dataset conversion scenario, the scalability of the solution is limited by memory consumption due to the materialization of large knowledge graphs. A potential alternative could exploit virtualization techniques for the lifting phase, but the state-of-the-art tools are still not mature enough to be employed in a conversion scenario [6]. To address the materialization scalability, the Chimera framework allows using an external repository to store the materialized graph. In our tests, we showed that this approach can effectively reduce memory

consumption, but it still has some limits. In particular, the use of an external repository shifts the bottleneck to the triplestore. For this reason, for very large datasets, we recommend to split the conversion as follows, under the assumption that the materialized graph does not change very often: (i) execution of the lifting procedure (if required, splitting the mappings in different executions); (ii) bulk loading of the materialized graph(s) into the triplestore (thus avoiding incremental indexing issues); (iii) on-demand lowering of the materialized graph (possibly with a separate Chimera pipeline). This approach also allows to select different lifting tools. Indeed, the RML specification has several implementations that can be chosen on the basis of the specific scenario requirements [1].

In the message conversion scenario, to cope with a higher number of concurrent requests, it is possible to deploy more than one instance of the converter exploiting a load balancing mechanism. However, standards in the transportation domain usually require dealing with a large set of different message types which implies defining a high number of converters. In this situation, an efficient scalability strategy is to deploy (multiple instances of) a *universal converter* which is able to dynamically select and execute the relevant mappings with respect to the input/output message.

5 Conclusions and Future Work

Semantic interoperability in the transportation domain can be addressed effectively exploiting Semantic Web technologies to enable the communication between actors and the definition of new services for travellers. To guarantee the adoption by relevant stakeholders, it is however extremely important to address their performance and scalability requirements. Considering both the dataset and message conversion scenario, this paper identified two test cases from the transportation domain and evaluated performances and scalability of the semantic data conversion approach using the Chimera framework, with a declarative approach based on RML mappings for lifting and on Apache Velocity templates and SPARQL queries for lowering.

Our analysis showed the potential and flexibility of the Chimera solution: in the dataset conversion scenario, we managed to generate and handle knowledge graphs with millions of triples; in the message conversion scenario, we obtained very low conversion times and a proved robustness also with hundred of concurrent requests per second. Moreover, on the basis of our tests, we defined a set of recommendations to improve performance and scalability with the discussed approach.

As future work, we would like to investigate and implement in Chimera further optimisations for the lifting procedure, for example adopting the data structures defined in the SDM-RDFizer tool [12] and investigating concurrency strategy improvements. Moreover, we would like to setup a more comprehensive benchmark for the message conversion scenario (e.g., based on GTFS-RT[18]).

[18] `https://developers.google.com/transit/gtfs-realtime`

Acknowledgments

The presented research was supported by the SPRINT project (Grant Agreement 826172), co-funded by the European Commission under the Horizon 2020 Framework Programme.

References

1. Arenas-Guerrero, J., Scrocca, M., Iglesias-Molina, A., Toledo, J., Pozo-Gilo, L., Dona, D., Corcho, O., Chaves-Fraga, D.: Knowledge graph construction with R2RML and RML: An ETL system-based overview. In: Proceedings of the 2nd International Workshop on Knowledge Graph Construction (2021)
2. Bennara, M., Zimmermann, A., Lefrançois, M., Messalti, N.: 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. pp. 149–153 (2020)
3. Carenini, A., Dell'Arciprete, U., Gogos, S., Pourhashem Kallehbasti, M.M., Rossi, M.G., Santoro, R., et al.: ST4RT – Semantic Transformations for Rail Transportation. In: Transport Research Arena TRA 2018. pp. 1–10 (2018)
4. Carenini, A., et al.: SPRINT project Deliverable D5.6 – Final report on the results of the validation of pilot implementation (2021), `http://sprint-transport.eu/`
5. Chaves-Fraga, D., Endris, K.M., Iglesias, E., Corcho, Ó., Vidal, M.: What are the parameters that affect the construction of a knowledge graph? In: On the Move to Meaningful Internet Systems. pp. 695–713. Springer (2019)
6. Chaves-Fraga, D., Priyatna, F., Cimmino, A., Toledo, J., Ruckhaus, E., Corcho, O.: GTFS-Madrid-Bench: A benchmark for virtual knowledge graph access in the transport domain. Journal of Web Semantics **65**, 100596 (2020)
7. Colpaert, P., Llaves, A., Verborgh, R., Corcho, O., Mannens, E., Van de Walle, R.: Intermodal public transit routing using linked connections. In: International Semantic Web Conference: Posters and Demos. pp. 1–5 (2015)
8. Comerio, M., Carenini, A., Scrocca, M., Celino, I.: Turn transportation data into EU compliance through semantic web-based solutions. In: Joint Proceedings of the 1st International Workshop On Semantics For Transport and the 1st International Workshop on Approaches for Making Data Interoperable co-located with 15th Semantics Conference (2019), `http://ceur-ws.org/Vol-2447/paper6.pdf`
9. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF mapping language. W3C recommendation, W3C (Sep 2012), http://www.w3.org/TR/2012/REC-r2rml-20120927/
10. Dimou, A., Sande, M.V., Colpaert, P., Verborgh, R., Mannens, E., de Walle, R.V.: 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`
11. Hosseini, M., Kalwar, S., Rossi, M.G., Sadeghi, M.: Automated mapping for semantic-based conversion of transportation data formats. In: 1st International Workshop On Semantics For Transport. vol. 2447, pp. 1–6 (2019)
12. Iglesias, E., Jozashoori, S., Chaves-Fraga, D., Collarana, D., Vidal, M.E.: SDM-RDFizer: An RML interpreter for the efficient creation of rdf knowledge graphs. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. pp. 3039–3046 (2020)

13. Meester, B.D., Maroy, W., Dimou, A., Verborgh, R., Mannens, E.: RML and FnO: Shaping DBpedia declaratively. In: The Semantic Web: ESWC 2017 Satellite Events. vol. 10577, pp. 172–177. Springer (2017). https://doi.org/10.1007/978-3-319-70407-4_32
14. Sadeghi, M., Buchníček, P., Carenini, A., Corcho, O., Gogos, S., Rossi, M., Santoro, R., et al.: SPRINT: Semantics for PerfoRmant and scalable INteroperability of multimodal Transport. In: 8th Transport Research Arena TRA 2020. pp. 1–10 (2020)
15. Scrocca, M., Comerio, M., Carenini, A., Celino, I.: Turning transport data to comply with EU standards while enabling a multimodal transport knowledge graph. In: The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part II. vol. 12507, pp. 411–429. Springer (2020). https://doi.org/10.1007/978-3-030-62466-8_26
16. Vetere, G., Lenzerini, M.: Models for semantic interoperability in service-oriented architectures. IBM System Journal **44**(4), 887–903 (2005)