# Boosting Knowledge Graph Generation from Tabular Data with RML Views

Julián Arenas-Guerrero(✉), Ahmad Alobaid, María Navas-Loro, María S. Pérez, and Oscar Corcho

Ontology Engineering Group, Universidad Politécnica de Madrid, Madrid, Spain
{julian.arenas.guerrero,ahmad.alobaid,m.navas, maria.s.perez,oscar.corcho}@upm.es

**Abstract.** A large amount of data is available in tabular form. RML is commonly used to declare how such data can be transformed into RDF. However, RML presents limitations that lead, in many cases, to the need for additional preprocessing using scripting. Although some proposed extensions (e.g., FnO or RML fields) address some of these limitations, they are verbose, unfamiliar to most data engineers, and implemented in systems that do not scale up when large volumes of data need to be processed. In this work, we expand RML views to tabular sources so as to address the limitations of this mapping language. In this way, transformation functions, complex joins, or mixed syntax can be defined directly in SQL queries. We present our extension of Morph-KGC to efficiently support RML views for tabular sources. We validate our implementation adapting R2RML test cases with views and compare it against state-of-the-art RML+FnO systems showing that our system is significantly more scalable. Moreover, we present specific examples of a real use case in the public procurement domain where basic RML mappings could not be used without additional preprocessing.

**Resource type**: Software framework
**License**: Apache 2.0
**DOI**: 10.5281/zenodo.7385488
**URL**: https://github.com/morph-kgc/morph-kgc

**Keywords:** Knowledge Graph · RML · CSV · Data Integration

## 1 Introduction

An extensive amount of data is stored as CSV, Microsoft Excel spreadsheets, and other tabular formats such as Apache Parquet [3] or Apache ORC [2]. Many organizations are also transforming these data sources into RDF knowledge graphs [30] (KGs), given their potential to integrate, represent, and publish heterogeneous data according to the model given by one or several ontologies.

Data transformations from tabular sources into RDF are typically defined in a systematic manner using mapping languages [43]. These languages increase the maintainability of the data integration pipelines and prevent the use of external scripting [13]. In addition, mappings leverage specialized data integration systems that come with rich functionality and are optimized for large-scale use cases.

The RDF Mapping Language [23] (RML) is a popular language [10] that extends the W3C Recommendation RDB to RDF Mapping Language [17] (R2RML) to data formats beyond relational databases (RDBs). In real-world data integration scenarios, some computations, such as transformation functions, complex joins, or extraction of embedded values, need to be applied to the input data. R2RML enables these computations by wrangling the data using SQL queries in the mappings that are executed over RDBs. However, RML does not allow this for tabular sources, which limits the capabilities of the mapping language for these common scenarios.

Although RML has already been extended with additional constructs to enable complex operations (e.g., FnO [19] and FunUL [16] for transformation functions, or RML fields [21] and mixed-syntax paths [36] for nested data), relying on SQL may ease the development of mappings by data engineers who know this query language well and are generally unfamiliar with semantic web technologies. Moreover, current implementations of these RML extensions, such as RMLMapper [38], RocketRML [39] and RMLStreamer [40], do not scale to large volumes of data [10]. This may impact the adoption of RML and its associated systems in particular, and maintainability and scalability of data integration pipelines in the broader scope.

In this work, we (i) analyze the limitations of RML and its implementations for handling tabular data, (ii) address them with RML views, (iii) extend a state-of-the-art system, Morph-KGC [8], to use SQL to define computations over the tabular sources, (iv) validate it with test cases and two benchmarks in the literature, and (v) apply our implementation to a real-world use case in the public procurement domain.

The manuscript is structured as follows. Section 2 presents an overview of RML, analyzes its limitations for tabular data, and expands RML views to tabular sources. Section 3 introduces our implementation as an extension of Morph-KGC. Section 4 validates the implementation using test cases and compares it with current alternatives. Section 5 applies our extension to a real use case in the public procurement domain. Finally, Sect. 6 presents the related work, and Sect. 7 wraps up with some conclusions and future work lines.

## 2 RML Tabular Views

In this section, we introduce the main limitations of RML for handling tabular data, as well as extensions that address part of them. After analyzing these limitations, we present our approach and show how it solves them relying on SQL.

### 2.1 RML Overview

RML is an extension of R2RML, a mapping language recommended by the W3C to generate RDF from RDBs. It generalizes R2RML to any data source.

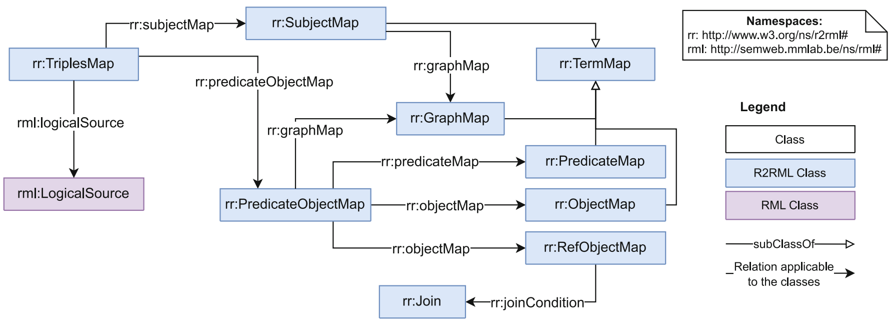


**Fig. 1.** RML overview (in Chowlk visual notation [24]).

Figure 1 depicts the structure of the RML mapping language, which includes `rml:LogicalSource` as an extension of R2RML's logical table.

An *RML mapping document* is an RDF document consisting of one or more *triples maps*. A triples map has one *logical source* which can be (according to the latest version of the RML specification [22]):

– A base source (any input source or a base table).
– A view (in the case of databases) given by a query.

For RDBs and tabular data sources, logical tables are iterated on a row basis to generate triples. However, for other data models such as XML or JSON, it is necessary to specify how this iteration occurs. This can be defined in RML via the property `rml:iterator`, which can be optionally accompanied by a `rml:referenceFormulation` specifying how the data is referred to, e.g., with XPath or JSONPath.

RDF terms in RML are generated through functions known as *term maps*, which describe how to generate them (using constants, references, or templates). Term maps can be subject, predicate, object, or graph maps determined by the position that the generated RDF terms take in the output RDF triples (or quads). Two triples maps can be joined with a *referencing object map*, which uses the subject map of the *parent triples map* to generate the object RDF terms.

## 2.2 Limitations of RML for Tabular Data Sources

As noted in the R2RML Recommendation, sometimes specific computations need to be performed over the input data, such as transformations or filtering. This can be achieved with R2RML views and the SQL query language by pushing down the computations to relational database management systems (RDBMSs). However, views in RML do not cover tabular data sources, which are restricted to RML's base source [22]. This reduces the capabilities of the mapping language and led to a number of proposals extending RML using additional constructs. In the following, we analyze the limitations of RML for tabular data along with some extensions in the literature that address them.

***Transformation Functions*** [19]**.** Transformations of the data need to be defined in the mapping to handle data cleaning and computations such as reshaping, aggregating, or filtering. RML's base source uses the data source as is, without additional modifications. Projections of the source may still be computed using the references in the term maps [31] to avoid processing unnecessary data. To allow declarative transformation functions in mappings, RML has been extended with additional constructs, such as FnO [18,19] or FunUL [16].

***Joins.*** RML is restricted regarding join operations over tabular sources. Referencing object maps join two triples maps (with their associated tabular base sources) for which join conditions can be defined using `rr:Join`. Some of the limitations of referencing object maps are:

– *Multiple joins.* A referencing object map involves two triples maps, consequently, RML does not support joining three or more tabular sources. To the best of our knowledge, no specific solution addressed this. A workaround to enable multiple joins is to create a relational schema for the tabular sources, load the data to an RDBMS and use an RML mapping with views to encode the joins in SQL queries. This implies increased complexity, due to the cost of defining the SQL schema, the addition of an RDBMS to the data integration pipeline, and the overhead of loading the data to it.
– *Theta joins* [32]. The class of join conditions in RML (i.e., `rr:Join`) only allows for equality conditions. This is shown in the R2RML Recommendation, where the *joint SQL query*[1] resulting from a referencing object map is an equijoin. However, some data integration scenarios require theta joins (or inequality joins). We are not aware of specific proposals tackling this limitation, for which the workaround described for multiple joins could be used.
– *Literal generation with joins* [20]. Referencing object maps use the subject map of parent triples maps to generate the object RDF terms of the output triples. Given that the term type of a subject map cannot be a literal (enforced by the RDF data model), it is not possible in RML to generate literals with RML's base source. Recently, Debruyne [20] has proposed an extension of RML enabling the generation of literals with joins.

***Mixed Content*** [36]**.** Tabular sources in real data integration use cases usually present composite data values: values such as JSON or lists are embedded in cells. This has been referred to as *mixed content* [36]. RML does not allow for mixed content, although solutions such as *fields* [21] or *mixed-syntax paths* [36] addressed this limitation.

## 2.3 RML Views over Tabular Data

The approach of R2RML to solve the limitations above is R2RML views, which uses the SQL query language to push down computations to RDBMSs. It must be noted that R2RML views are different from *materialized views* [26]; the former

[1] https://www.w3.org/TR/r2rml/#dfn-joint-sql-query.

is an SQL query that is executed once and whose results are not persisted in the RDBMS, the latter is a table in the database resulting from the execution of a query for which refreshing policies apply (incremental, at regular time intervals, or on demand).

The RML specification [22] currently limits the scope of RML views to databases (the `rr:R2RMLView` class is not extended). The seminal work of RML [23] already devised that logical sources could be extended to support views over other data sources to allow data cleaning and transformation. However, views have not yet been considered for tabular sources.

We extend RML views to tabular data, which address the limitations of RML's base source. An RML view over tabular sources is a logical table populated with data resulting from the execution of an SQL query against the input tabular sources. It is represented by a resource with:

- One `rml:query` property (which extends R2RML's `rr:sqlQuery`), whose value is a literal with a lexical form that is a valid SELECT SQL query. The query result set cannot have duplicate column names, and projected columns resulting from an expression (e.g., aggregates) must be aliased to allow for referenciation from term maps.
- Zero or one `rml:referenceFormulation` property. Because of backwards compatibility with R2RML the property `rr:sqlVersion` can also be used. RML predefines `ql:CSV` to refer to CSV files using columns. In the case of RML views, the default is `rr:SQL2008` but others could be used[2]. This reference formulation applies to CSV and other tabular data formats such as Apache Parquet.
- Zero or one `rml:iterator` property. This is optional, since the default per-row iteration is assumed.

R2RML processors require an SQL connection[3] to access the input database. RML views over tabular sources do not need this connection, being the input sources directly referenced in the SQL query (using absolute or relative paths to the tabular files in the system or a URL for a remote file), which can be conveniently aliased. Since tabular sources are referenced as they are (in an RDBMS two tables with the same name can coexist in different databases), a default catalogue and schema are used.

## 3 Morph-KGC: An Extension for RML Tabular Views

Our implementation of RML views over tabular data is based on Morph-KGC [8]. This data integration system is optimized to process large volumes of data and supports the R2RML, RML, and RML-star [9] mapping languages. Due to the latter, our extension also generates graphs in the emerging RDF-star [27] data model. The system is implemented in Python and is built on top of the Pandas library [35].

[2] https://www.w3.org/2001/sw/wiki/RDB2RDF/SQL_Version_IRIs.
[3] https://www.w3.org/TR/r2rml/#dfn-sql-connection.

The mapping parser component was expanded to RML views. A logical source is now defined internally by two variables: a type and a value. The type can be a view, a source, or a table name, and the values are an SQL query, a path to a data file, or a table in an RDB, respectively. Additionally, the source format for a view (RDB or tabular, needed by the data loader component) is determined by the presence or lack of presence of a database connection. If the RML view is accompanied by a database connection, it is associated to the RDB format, otherwise it is associated to the tabular format.

Tabular data is ingested into Morph-KGC using Pandas for RML's base source and DuckDB [37] for RML views, for which a new connector to this state-of-the-art embedded analytical database has been implemented. The connector currently supports CSV (with any delimiter, inferred on the fly) and Apache Parquet, which are accessed locally or remotely (hosted by cloud providers). After evaluating a view, the query result set is transformed into a Pandas DataFrame, which is the internal structure used by Morph-KGC for processing data. If the logical source is related to a referencing object map or a star map, it will be further joined internally. Given the modular design of the system and the use of a source-independent internal data structure, the materialization procedure is not affected.

This extension of Morph-KGC allows pushing down some operations in the mappings, such as duplicate removal (using the DISTINCT clause), NULL elimination (IS NOT NULL statement) or joins (replacing referencing object maps) that can improve its performance. To the authors' knowledge, Morph-KGC is the first system that implements RML views over tabular data, solving the issues of RML's base source presented in Sect. 2.2, and avoiding the use of additional constructs such as RML+FnO or RML fields. The flexibility of views also enables the creation of identifiers when they are missing from tabular sources[4], and can potentially solve more limitations of base RML that may arise. The supported SQL syntax in the mapping is that of DuckDB (it can be consulted in its documentation[5]), which is derived from PostgreSQL. The main limitation of the system is the lack of user-defined functions; however, built-in SQL functions cover most data integration use cases [13] and we are already working to support them[6].

Regarding scalability, the core optimization of Morph-KGC is based on *mapping partitioning* [8]. This technique consists in creating groups of mapping rules that produce disjoint (i.e., non-overlapping) sets of triples. Each group of mappings can then be independently processed, generating a KG which is free of duplicate triples. If a parallel execution of the mapping groups is used, then the materialization time is minimized, and when they are sequentially processed, memory consumption is reduced. RML+FnO prevents good mapping partitioning (i.e., obtaining a mapping partition with a high number of groups), while our extension does not affect the partitioning. This is because to obtain a mapping

[4] https://github.com/morph-kgc/morph-kgc/discussions/102.
[5] https://duckdb.org/docs/sql/introduction.
[6] https://github.com/morph-kgc/morph-kgc/issues/117.

partition, the constant part of term maps (in `rr:constant` and `rr:template`) is used; however, for function maps it is not possible to make assumptions of constant parts of the generated RDF terms. By encoding transformation functions in RML views, we avoid the need of function maps in Morph-KGC, thus obtaining better mapping partitions and consequently, ensuring scalability.

***Availability.*** The source code of Morph-KGC is actively maintained in a public GitHub repository[7]. The releases are archived in Zenodo [5] with their corresponding DOIs and distributed through the Python Package Index[8] (PyPi). The system is available under the Apache License 2.0 and its documentation is licensed under CC BY-SA 4.0.

***Reusability.*** Morph-KGC is accompanied by detailed documentation hosted in Read the Docs[9]. A tutorial of the system is available on the Google Collaboration[10] platform using Python Notebooks with guide descriptions for users, and it has also been presented in tutorials. Morph-KGC is also used in semantic web courses in Universidad Politécnica de Madrid at the undergraduate and postgraduate levels and could be reused for similar courses by other universities. Furthermore, Morph-KGC is currently being used in several projects where the Ontology Engineering Group is involved, including domains such as public procurement (presented in Sect. 5) or labour (the EU project AI4LABOUR[11], where occupations and skills are linked to training courses for employees). In addition, we are also supporting large private organizations (in the insurance and manufacturing sectors[12]) in their data integration pipelines with tabular sources, and the issues in the GitHub repository show that Morph-KGC is being used by external organizations.

***Design and Technical Quality.*** We carried out extensive evaluations validating that our RML views extension of Morph-KGC performs similar to the original system. As it will be explained in Sect. 4.1, we developed test cases for RML views and added them to the continuous integration pipeline of the system.

## 4 Evaluation

In this section, we present the evaluation of RML views in Morph-KGC. First, we extend the R2RML test cases using R2RML views to tabular sources and RML. Next, we compare the performance of our system with respect to state-of-the-art RML+FnO engines using GTFS-Madrid-Bench [14]. Finally, we use the LUBM4OBDA benchmark to validate that our system performs similarly to using an RDBMS populated with the tabular data. All experiments are run

[7] https://github.com/morph-kgc/morph-kgc.
[8] https://pypi.org/project/morph-kgc/.
[9] https://morph-kgc.readthedocs.io.
[10] https://github.com/morph-kgc/morph-kgc/tree/main/examples/tutorial.
[11] https://doi.org/10.3030/101007961.
[12] Names are not disclosed for confidentiality reasons.

using Morph-KGC v2.3.1. The evaluation was performed on an Intel® Core$^{\text{TM}}$ i7-1165G7 (2.80 GHz) and a memory of 40 GB RAM DDR4 (3200 MHz). All the times reported are the average time of 5 executions, and we set the timeout to 24 h.

## 4.1 Validation with Test Cases

The R2RML test cases [44] are a companion of the R2RML Recommendation to validate the compliance of systems with respect to the mapping language. Later, they were extended to RML [28]. However, given the lack of RML views for tabular data sources at that time, test cases with R2RML views were only considered for RDBs and excluded for the CSV data format. In order to validate the compliance of Morph-KGC, we extended these R2RML test cases to RML views. Table 1 lists them along with a description and an alternative solution using RML's base source with additional constructs. It must be noted that the test cases RMLTVTC0015a, RMLTVTC0015b and RMLTVTC0019a were included in the RML test cases for CSV [28], but the tabular files were preprocessed to enable RML's base source. Here, we maintain the original structure of the data in the R2RML test cases. We also created four additional test cases (RMLTVTC0026a, RMLTVTC0027a, RMLTVTC0028a, RMLTVTC0029a) to further validate some of the limitations described in Sect. 2.2.

Morph-KGC successfully passes all test cases, hence validating the compliance of the system with respect to RML views. Test cases are publicly available in the GitHub repository[13] and Zenodo [6]. They are used for automated and continuous testing of Morph-KGC with GitHub Actions.

## 4.2 Transformation Functions with GTFS-Madrid-Bench

In this experiment, we compare RML views to RML+FnO, and also evaluate the performance of the RML view extension of Morph-KGC to state-of-the-art RML+FnO systems. Materials are publicly available in Zenodo [4].

We use GTFS-Madrid-Bench, a benchmark in the transport domain which is widely used to evaluate ontology-based data integration systems. The benchmark consists of 10 CSV files and the materialized KG for scaling factor 100 contains more than 35 millions of triples. The target data model includes 9 data properties with `xsd:boolean` datatypes. Given that the benchmark produces 1s and 0s as boolean values, the CSV data needs to be transformed to "true" and "false" respectively to prevent the generation of *ill-typed literals*[14]. However, the mappings provided by the benchmark do not take this into account, so we extended them to address this issue using RML views and RML+FnO. Listing 1.1 shows an example mapping rule (in the human-readable YARRRML [29] syntax) for RML views, which employs two replace functions (other alternatives, such as casting to boolean, are possible). Listing 1.2 shows the same example

[13] https://github.com/morph-kgc/morph-kgc/tree/main/test/rmltv.

[14] https://www.w3.org/TR/r2rml/#dfn-ill-typed.

**Table 1.** Test cases that are not supported by RML's base source.

| Test Case | Description | RML Support |
|---|---|---|
| RMLTVTC0002d | Concatenation of a column and a string | RML+FnO |
| RMLTVTC0002g | Tests the presence of an invalid SQL query | N/A |
| RMLTVTC0002h | Tests the presence of duplicate column names in the SELECT list of the SQL query | N/A |
| RMLTVTC0002i | Two columns mapping, SQL version identifier | N/A |
| RMLTVTC0002j | Two columns mapping, qualified column names | N/A |
| RMLTVTC0003b | Concatenation of two columns and a string | RML+FnO |
| RMLTVTC0009c | Concatenation of two columns and a string | RML+FnO |
| RMLTVTC0009d | Aggregation of a column | RML+FnO |
| RMLTVTC0011a | M to M relation, by using an SQL query | Yes (by using an additional Triples Map) |
| RMLTVTC0014d | Replacement of data values | RML+FnO |
| RMLTVTC0015a | Filtering | RML+FnO |
| RMLTVTC0015b | Filtering | RML+FnO |
| RMLTVTC0019a | Filtering | RML+FnO |
| RMLTVTC0026a | Embedded list in a column | RML+FnO |
| RMLTVTC0027a | Embedded JSON in a column | RML+Fields |
| RMLTVTC0028a | Generation of literals with joins | RML+ [20] |
| RMLTVTC0029a | Join of multiple sources | No |

mapping with RML+FnO using a composite function: a condition for filtering the 1s/0s and a replace function to transform the values to true/false. As can be observed, RML+FnO results in a more verbose mapping that may impact their maintainability.

For the performance evaluation of both approaches, we compare Morph-KGC v2.3.1 to RMLMapper v6.0.0 [38] and RocketRML 2.1.0 [39]. For RMLMapper we employed predefined functions (as shown in Listing 1.2), and for RocketRML we implemented a user-defined function (since the provided function set of the system is more limited). Figure 2(a) depicts the materialization times of the systems for data scaling factors 1, 10 and 100 of GTFS-Madrid-Bench. Morph-KGC is one order of magnitude faster than RocketRML, and the difference increases even more with respect to RMLMapper. In fact, the former yields an out-of-memory error and the latter produces timeouts when materializing the KG for scaling factor 100.

**Listing 1.1.** RML views mapping example for GTFS-Madrid-Bench and Morph-KGC.

```
mappings:
  calendar_date_rules:
    sources:
      - query: |
         SELECT service_id, date, REPLACE(REPLACE(
             exception_type, '1', 'true'), '0', 'false') AS
             exception_type
         FROM 'data/CALENDAR_DATES.csv'
    s: \url{http://transport.linkeddata.es/madrid/metro/
        calendar_date_rule/{service_id}-{date}}
    po:
      - [gtfs:dateAddition, $(exception_type), xsd:boolean]
```

**Listing 1.2.** RML+FnO mapping example for GTFS-Madrid-Bench and RMLMapper.

```
mappings:
  calendar_date_rules:
    sources:
      - [data/CALENDAR_DATES.csv~csv]
    s: \url{http://transport.linkeddata.es/madrid/metro/
        calendar_date_rule/$(service_id)-$(date)}
    po:
      - predicates: gtfs:dateAddition
        objects:
          datatype: xsd:boolean
          function: grel:string_replace
          parameters:
            - [grel:valueParameter, $(exception_type)]
            - [grel:p_string_find, "1"]
            - [grel:p_string_replace, "true"]
        condition:
          function: idlab-fn:stringContainsOtherString
          parameters:
            - [idlab-fn:str, $(exception_type)]
            - [idlab-fn:otherStr, "1"]
            - [idlab-fn:delimiter, ""]
      - predicates: gtfs:dateAddition
        objects:
          datatype: xsd:boolean
          function: grel:string_replace
          parameters:
            - [grel:valueParameter, $(exception_type)]
            - [grel:p_string_find, "0"]
            - [grel:p_string_replace, "false"]
        condition:
          function: idlab-fn:stringContainsOtherString
          parameters:
            - [idlab-fn:str, $(exception_type)]
            - [idlab-fn:otherStr, "0"]
            - [idlab-fn:delimiter, ""]
```

### 4.3 Multiple Joins with the LUBM4OBDA Benchmark

Real data integration use cases over tabular data usually involve performing complex joins. In these cases, RML views is the only solution that does not require preprocessing, since RML's base source cannot deal with them even with extensions. The aim of this experiment is to show how our extension of Morph-KGC can handle complex joins efficiently even in the presence of large volumes of data.

To evaluate Morph-KGC in data integration scenarios with multiple joins over tabular data, we used the LUBM4OBDA benchmark[15]. LUBM4OBDA is an ontology-based data access benchmark (in the university domain) over RDBs that involves R2RML views with up to four joins in the SQL queries. Since the benchmark provides the data as SQL dumps, we exported the tables as tabular data in CSV and Apache Parquet (in a similar manner as done by GTFS-Madrid-Bench) formats. The benchmark consists of 14 tabular files, that result in an output KG of more than 150 million triples for scaling factor 1000. This data and the mappings are openly available in Zenodo [7]. We exclude RMLMapper and RocketRML from this experiment since they do not allow multiple joins. As an alternative, we considered a setup in which a relational representation of the tabular sources is created, the tabular data is loaded into an RDBMS, and mappings using standard R2RML views are used (as explained in Sect. 2.2). We just take into account the materialization times, ignoring the additional cost of creating the relational representation, and the overhead of loading the data into an RDBMS. We use PostgreSQL 15.0, MySQL 8.0.31 and data scaling factors 1, 10, 100 and 1000 of LUBM4OBDA.

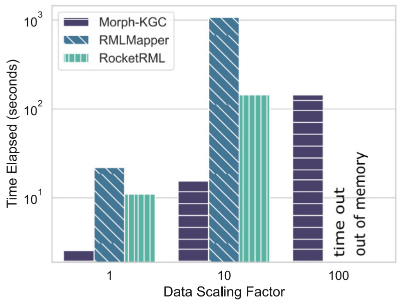
Figure 2(b) shows the materialization times obtained. It is observed that Morph-KGC is faster when materializing directly from tabular sources compared to relying on RDBMSs. Differences are appreciated between RDBMSs, in particular, while PostgreSQL is not far from the materialization times obtained for tabular data, times significantly increase for MySQL when the scaling factor is large. This proves that our implementation supports multiple joins and that it is even more efficient than relying on RDBMSs.

## 5 A Real World Use Case in Public Procurement

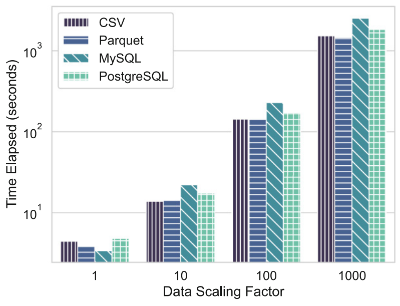
Public procurement represents a relevant budget expense of many states worldwide. For example, the European Union spends around 14% of its annual gross domestic product on the purchase of services, utilities, and supplies[16]. Free access to this data facilitates accountability and transparency. Therefore, many public administrations (at the local, regional, and international levels) provide these data on their own open data portals [41].

[15] https://github.com/oeg-upm/lubm4obda.

[16] https://ec.europa.eu/growth/single-market/public-procurement_en.



(a) **GTFS-Madrid-Bench.**



(b) **LUBM4OBDA benchmark.**

**Fig. 2.** Execution times for GTFS-Madrid-Bench in CSV format (Morph-KGC with RML views and RMLMapper and RocketRML with RML+FnO), and the LUBMM4OBDA benchmark (Morph-KGC over different tabular formats and RDBMSs). Times are reported using a logarithmic scale.

In NextProcurement[17] we are developing an open, harmonized, and enriched public procurement data platform for Europe. In this case, the extension of Morph-KGC with RML views has been successfully used to transform Spanish public procurement data available in Apache Parquet format (mainly obtained from the national portal PLACE/PLASCP[18], together with some regional contracting platforms) into RDF according to the Open Contracting Data Standard (OCDS) ontology [41]. Prior to the use of RML views, the RDF was being generated by applying programmatic preprocessing in Python, and then using base sources in the mappings. The final public procurement service will be deployed on an external server, and the fact of directly using Morph-KGC without additional preprocessing simplifies the deployment and its maintainability.

In the following we introduce two specific situations where RML's base source is not enough to generate the output RDF, and how RML views have been used to overcome this.

## 5.1 Translating Spanish Codes to the Range Represented in the OCDS Ontology

Spanish public procurement procedures are usually categorized following a numeric typology[19], whose codes are defined upon the European Directive 2004/18/CE[20] and by the IDABC (Interoperable Delivery of Pan-European eGovernment Services to Public Administrations, Business and Citizens) Functional Requirements. However, these codes do not always have a direct mapping

[17] http://nextprocurement-project.com/.
[18] https://contrataciondelestado.es.
[19] https://contrataciondelestado.es/codice/cl/2.04/SyndicationTenderingProcessCode-2.04.gc.
[20] https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A32004L0018.

to the types of procedures detailed in the OCDS ontology. In the code excerpt below (Listing 1.3) it is shown how this was preprocessed before, using Python scripting to map the values in a Pandas DataFrame to the expected values.

**Listing 1.3.** Translating spanish codes to the OCDS ontology with Python.

```
tipos_contrato = { 1:  "goods",
                   2:  "services",
                   3:  "works",
                   21: "services",
                   31: "works" }
df["Tipo␣de␣Contrato"] = df["Tipo␣de␣Contrato"].map(
    tipos_contrato).fillna("other")
```

When we shifted to RML views, this mapping could be solved with the CASE statement in the SQL query (Listing 1.4). A similar solution[21] is exemplified in the R2RML Recommendation, which until now was only possible for RDBs.

**Listing 1.4.** Translating spanish codes to the OCDS ontology with SQL.

```
SELECT NextProcurement.*,
       CASE "Tipo␣de␣Contrato"
           WHEN 1 THEN 'goods'
           WHEN 2 THEN 'services'
           WHEN 3 THEN 'works'
           WHEN 21 THEN 'services'
           WHEN 31 THEN 'works'
           ELSE 'other'
        END AS TipoContrato
FROM 'NextProcurement.parquet' AS NextProcurement
```

## 5.2 Handling Embedded Lists of Lots in Procedures

Public procurement procedures may involve several tasks of different types (e.g., when a school starts operating, both the new materials and vacancies for the employees must be tendered). To facilitate organizations to apply only for the parts of the service that they are interested in, public procurement procedures are usually divided into lots. In our case, the input data associated with lots are in the form of lists embedded in the cells of an Apache Parquet file. RML views and Morph-KGC enable the processing of these lists by using unnesting, casting and splitting operations, as shown in the code excerpt below (Listing 1.5).

**Listing 1.5.** Processing embedded lists of lots in procedures with SQL.

```
SELECT "Número␣de␣Expediente",
    UNNEST("Lote"::DOUBLE[]) AS Lotes,
    UNNEST(
       (string_split
```

[21] https://www.w3.org/TR/r2rml/#example-translationtable.

```
            (replace("Número␣de␣Licitadores␣Participantes", '
               NULL', '0')[2:-1], ',')
             ::DOUBLE[])
        ::INT[]) AS NumTenders
    FROM 'NextProcurement.parquet'
```

## 6 Related Work

The SPARQL query language has been extended in several works, such as SPARQL-Anything [11], SPARQL-Generate [33] or Tarql [1], to generate RDF KGs from tabular data. Similarly to SQL in RML views, SPARQL functions allow for data transformation using the GENERATE clause in SPARQL-Generate and the CONSTRUCT query form in SPARQL-Anything (by overloading the SERVICE clause) and Tarql (via the FROM clause). Complex joins are enabled through nested GENERATE and CONSTRUCT clauses, but mixed content is not supported. The main difference with respect to our work is that SPARQL-based approaches use a query language over the target ontology, while RML views use a query language over the tabular sources (these approaches are known as local and global as view [34], respectively). Semantic web practitioners may prefer these alternatives, since they are familiar with SPARQL, while data engineers who are used to SQL may lean towards RML views.

García-González et al. [25] proposed using Shape Expressions [12] to map heterogeneous data to RDF. The Shape Expressions Mapping Language (ShExML) relies on a data validation language instead of a query language. ShExML is more limited regarding transformation functions, for which only matchers and string operations are supported, and filtering or mixed content is not possible. Also, limited join functionality can be achieved with shape linking and the JOIN clause.

Szekely et al. [42] proposed the T2WML language to allow layouts beyond the canonical tabular representation (one column for each variable). T2WML maps on a cell-centric basis rather than the row-centric model of RML's base source. Although T2WML allows transformation functions, it does not support joins between different tabular sources or mixed content. YAML is used to write T2WML rules, similar to YARRRML [29], a popular human-readable serialization of RML.

As already discussed during the paper, several mechanism extending RML have been proposed to increase the flexibility of the mapping language. RML has been aligned to FnO [19] and FunUL [16], which define functions in a generic and reusable way. While these approaches define functions within term maps, RML views define them directly in the logical source. RML fields introduces a nested iteration model to handle mixed-content, and the work presented in [36] proposes the concatenation of path expressions using the *mixed-syntax paths* constructs. RML+FnO and RML fields are now under the hood of the W3C Knowledge

Graph Construction Community Group[22]. Chaves-Fraga et al. [15] studied how to efficiently load tabular sources to an RDBMS, and perform SPARQL-to-SQL query translation to enable virtualization over tabular sources. This also allows complex joins similarly to RML views by delegating on the RDBMS. However it tackles virtualization, while we focus on the generation of the KG.

## 7 Conclusions and Future Work

This paper presents an open-source extension of Morph-KGC for KG generation from tabular data with RML views. Our implementation enables transformation functions, complex joins, and mixed content using SQL queries within RML mappings. In this way, Morph-KGC can potentially boost the adoption of RML, especially by data engineers, since they are usually more familiar with SQL than with RML extensions used so far for these purposes.

To validate the capabilities of our implementation, we extended some R2RML test cases for tabular data. We showed that our system significantly outperforms state-of-the-art RML+FnO for transformation functions and that it is the only one that allows complex joins over tabular sources. Furthermore, we demonstrated how Morph-KGC and RML views are being applied in a real use case in public procurement, replacing programmatic preprocessing.

We made publicly available (via Zenodo and GitHub) all the resources: the RML tabular views test cases, the RML+FnO mappings for GTFS-Madrid-Bench and the tabular data dumps and mappings for the LUBM4OBDA benchmark. The system is under active development and outlining its road map, we have already started working to support user-defined functions with RML+FnO and we plan to enhance its usability allowing the YARRRML human-friendly serialization of RML.

**Acknowledgments.** This work was funded partially by the project Knowledge Spaces: Técnicas y herramientas para la gestión de grafos de conocimientos para dar soporte a espacios de datos (Grant PID2020-118274RB-I00, funded by MCIN/AEI/ 10.13039/501100011033).

## References

1. Tarql: SPARQL for Tables (2019). https://tarql.github.io/
2. Apache Software Foundation: Apache ORC. https://orc.apache.org/
3. Apache Software Foundation: Apache Parquet. https://parquet.apache.org/
4. Arenas-Guerrero, J.: Evaluation of RML tabular views with GTFS-Madrid-Bench (2022). https://doi.org/10.5281/zenodo.7389828
5. Arenas-Guerrero, J.: oeg-upm/morph-kgc (2022). https://doi.org/10.5281/zenodo.5543552
6. Arenas-Guerrero, J.: RML tabular views test cases (2022). https://doi.org/10.5281/zenodo.7389760

[22] https://www.w3.org/community/kg-construct/.

7. Arenas-Guerrero, J.: The LUBM4OBDA benchmark for tabular sources (2022). https://doi.org/10.5281/zenodo.7389705
8. Arenas-Guerrero, J., Chaves-Fraga, D., Toledo, J., Pérez, M.S., Corcho, O.: Morph-KGC: scalable knowledge graph materialization with mapping partitions. Semant. Web (2022). https://doi.org/10.3233/SW-223135
9. Arenas-Guerrero, J., Iglesias-Molina, A., Chaves-Fraga, D., Garijo, D., Corcho, O., Dimou, A.: Morph-KGCstar: declarative generation of RDF-star graphs from heterogeneous data. Submitted to Semantic Web (2023). https://www.semantic-web-journal.net/system/files/swj3238.pdf
10. Arenas-Guerrero, J., et al.: Knowledge Graph Construction with R2RML and RML: an ETL System-based Overview. In: Proceedings of the 2nd International Workshop on Knowledge Graph Construction, vol. 2873. CEUR Workshop Proceedings (2021). http://ceur-ws.org/Vol-2873/paper11.pdf
11. Asprino, L., Daga, E., Gangemi, A., Mulholland, P.: Knowledge graph construction with a façade: a unified method to access heterogeneous data sources on the web. ACM Trans. Internet Technol. (2022). https://doi.org/10.1145/3555312
12. Boneva, I., Labra Gayo, J.E., Prud'hommeaux, E.G.: Semantics and validation of shapes schemas for RDF. In: d'Amato, C., et al. (eds.) ISWC 2017. LNCS, vol. 10587, pp. 104–120. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68288-4_7
13. Chaves-Fraga, D., Corcho, O., Yedro, F., Moreno, R., Olías, J., De La Azuela, A.: Systematic construction of knowledge graphs for research-performing organizations. Information **13**(12), 562 (2022). https://doi.org/10.3390/info13120562
14. 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. J. Web Semant. **65**, 100596 (2020). https://doi.org/10.1016/j.websem.2020.100596
15. Chaves-Fraga, D., Ruckhaus, E., Priyatna, F., Vidal, M.E., Corcho, O.: Enhancing virtual ontology based access over tabular data with Morph-CSV. Semant. Web **12**(6), 869–902 (2021). https://doi.org/10.3233/SW-210432
16. Crotti Junior, A., Debruyne, C., Brennan, R., O'Sullivan, D.: An evaluation of uplift mapping languages. Int. J. Web Inf. Syst. **13**(4), 405–424 (2017). https://doi.org/10.1108/IJWIS-04-2017-0036
17. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF mapping language. W3C Recommendation, World Wide Web Consortium (W3C) (2012). http://www.w3.org/TR/r2rml/
18. De Meester, B., Maroy, W., Dimou, A., Verborgh, R., Mannens, E.: Declarative data transformations for linked data generation: the case of DBpedia. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) ESWC 2017. LNCS, vol. 10250, pp. 33–48. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58451-5_3
19. De Meester, B., Seymoens, T., Dimou, A., Verborgh, R.: Implementation-independent function reuse. Futur. Gener. Comput. Syst. **110**, 946–959 (2020). https://doi.org/10.1016/j.future.2019.10.006
20. Debruyne, C.: Supporting relational database joins for generating literals in R2RML. In: Proceedings of the 3rd International Workshop on Knowledge Graph Construction, vol. 3141. CEUR Workshop Proceedings (2022). http://ceur-ws.org/Vol-3141/paper7.pdf
21. Delva, T., Van Assche, D., Heyvaert, P., De Meester, B., Dimou, A.: Integrating nested data into knowledge graphs with RML Fields. In: Proceedings of the

2nd International Workshop on Knowledge Graph Construction, vol. 2873. CEUR Workshop Proceedings (2021). http://ceur-ws.org/Vol-2873/paper9.pdf
22. Dimou, A., Vander Sande, M.: RDF mapping language (RML). Technical report, World Wide Web Consortium (W3C) (2022). https://rml.io/specs/rml/
23. Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R.: RML: a generic language for integrated RDF mappings of heterogeneous data. In: Proceedings of the 7th Workshop on Linked Data on the Web, vol. 1184. CEUR Workshop Proceedings (2014). http://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf
24. Feria, S.C., García-Castro, R., Poveda-Villalón, M.: Chowlk: from UML-based ontology conceptualizations to OWL. In: Groth, P., et al. (eds.) ESWC 2022. LNCS, vol. 13261, pp. 338–352. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-06981-9_20
25. García-González, H., Boneva, I., Staworko, S., Labra-Gayo, J.E., Lovelle, J.M.C.: ShExML: improving the usability of heterogeneous data mapping languages for first-time users. PeerJ Comput. Sci. **6**, e318 (2020). https://doi.org/10.7717/peerj-cs.318
26. Goldstein, J., Larson, P.R.: Optimizing queries using materialized views: a practical, scalable solution. SIGMOD Rec. **30**(2), 331–342 (2001). https://doi.org/10.1145/376284.375706
27. Hartig, O.: Foundations of RDF* and SPARQL* (an alternative approach to statement-level metadata in RDF). In: Proceedings of the 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, vol. 1912. CEUR Workshop Proceedings (2017). http://ceur-ws.org/Vol-1912/paper12.pdf
28. Heyvaert, P., et al.: Conformance test cases for the RDF mapping language (RML). In: Villazón-Terrazas, B., Hidalgo-Delgado, Y. (eds.) KGSWC 2019. CCIS, vol. 1029, pp. 162–173. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21395-4_12
29. Heyvaert, P., De Meester, B., Dimou, A., Verborgh, R.: Declarative rules for linked data generation at your fingertips! In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 11155, pp. 213–217. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98192-5_40
30. Hogan, A., et al.: Knowledge graphs. ACM Comput. Surv. **54**(4) (2021). https://doi.org/10.1145/3447772
31. Jozashoori, S., Vidal, M.-E.: MapSDI: a scaled-up semantic data integration framework for knowledge graph creation. In: Panetto, H., Debruyne, C., Hepp, M., Lewis, D., Ardagna, C.A., Meersman, R. (eds.) OTM 2019. LNCS, vol. 11877, pp. 58–75. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33246-4_4
32. Khayyat, Z., Lucia, W., Singh, M., Ouzzani, M., Papotti, P., Quiané-Ruiz, J.-A., Tang, N., Kalnis, P.: Fast and scalable inequality joins. VLDB J. **26**(1), 125–150 (2016). https://doi.org/10.1007/s00778-016-0441-6
33. Lefrançois, M., Zimmermann, A., Bakerally, N.: A SPARQL extension for generating RDF from heterogeneous formats. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) ESWC 2017. LNCS, vol. 10249, pp. 35–50. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58068-5_3
34. Lenzerini, M.: Data integration: a theoretical perspective. In: Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS, pp. 233–246. Association for Computing Machinery (2002). https://doi.org/10.1145/543613.543644

35. McKinney, W.: Data structures for statistical computing in Python. In: Proceedings of the 9th Python in Science Conference, pp. 56–61 (2010). https://doi.org/10.25080/Majora-92bf1922-00a
36. Michel, F., Djimenou, L., Zucker, C.F., Montagnat, J.: Translation of relational and non-relational databases into RDF with xR2RML. In: Proceedings of the 11th International Conference on Web Information Systems and Technologies, vol. 1, pp. 443–454. SciTePress (2015). https://doi.org/10.5220/0005448304430454
37. Raasveldt, M., Mühleisen, H.: DuckDB: an embeddable analytical database. In: Proceedings of the 2019 International Conference on Management of Data, pp. 1981–1984. Association for Computing Machinery (2019). https://doi.org/10.1145/3299869.3320212
38. RMLio: RMLMapper (2022). https://github.com/RMLio/rmlmapper-java
39. Şimşek, U., Kärle, E., Fensel, D.: RocketRML - a NodeJS implementation of a use-case specific RML mapper. In: Proceedings of the 1st International Workshop on Knowledge Graph Building, vol. 2489, pp. 46–53. CEUR Workshop Proceedings (2019). http://ceur-ws.org/Vol-2489/paper5.pdf
40. Sitt Min, O., Gerald, H., Ben, D.M., Anastasia, D.: RMLStreamer-SISO: an RDF stream generator from streaming heterogeneous data. In: Sattler, U., et al. (eds.) ISWC 2022. LNCS, vol. 13489, pp. 697–713. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-19433-7_40
41. Soylu, A., et al.: TheyBuyForYou platform and knowledge graph: expanding horizons in public procurement with open linked data. Semant. Web **13**(2), 265–291 (2022). https://doi.org/10.3233/SW-210442
42. Szekely, P., Garijo, D., Bhatia, D., Wu, J., Yao, Y., Pujara, J.: T2WML: table to wikidata mapping language. In: Proceedings of the 10th International Conference on Knowledge Capture, pp. 267–270. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3360901.3364448
43. Van Assche, D., Delva, T., Haesendonck, G., Heyvaert, P., De Meester, B., Dimou, A.: Declarative RDF graph generation from heterogeneous (semi-)structured data: a systematic literature review. J. Web Semant. **75**, 100753 (2023). https://doi.org/10.1016/j.websem.2022.100753
44. Villazón-Terrazas, B., Hausenblas, M.: R2RML and direct mapping test cases. W3C Note, World Wide Web Consortium (W3C) (2012). http://www.w3.org/TR/rdb2rdf-test-cases/