Reference Conditions: Relating Mapping Rules Without Joining

Els de Vleeschauwer[1], Sitt Min Oo[1], Ben De Meester[1] and Pieter Colpaert[1]

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

Abstract

Existing knowledge graph construction mapping languages have a legacy of mapping over relational databases. A such, current mapping language join constructs conflate securing referential integrity with relating concepts across data sources. This leads to a significant amount of operations (resulting in performance bottlenecks), and loss of additional context of potential linking triples that are not generated due to a lack of referential integrity. We propose a reference condition next to the traditional join condition, allowing to express a relation between sources without imposing any referential integrity. In this short research paper, we describe the concept, its applicability, how it could be integrated in existing and future mapping languages, and a proof-of-concept implementation. Our evaluation based on GTFS-Madrid-Bench confirms the assumption that removing these integrity checks leads to much faster generation times, but we also find that using reference conditions results in exactly the same graph output, i.e. these alternative semantics do not influence generation results for cases where referential integrity is assumed within the source system. Adding the reference conditions keeps the best of all worlds: generation time is shortened where possible, you have more context in the resulting RDF graph, and the mapping file still supplies relevant metadata about relations between triples maps. For future work, we will further research relations between triples maps, and expand our implementation to more complex reference conditions. This will allow us to investigate similar performance gains with other benchmarks and other mapping engines.

Keywords

R2RML, RML, relation, join, optimization

1. Introduction

Knowledge graph construction has been largely simplified by using established mapping languages and well-maintained mapping engines. The increasing maturity of this field is exemplified by the activities of W3C's Knowledge Graph Construction Community Group, and recent research detailing optimization strategies implemented in mapping engines.

Existing mapping languages historically extended W3C's recommended Relational to RDF Mapping Language (R2RML) [1]: the RDF Mapping Language (RML) being the most direct example [2], but we can safely assume that other languages such as SPARQL-Generate [3]

KGCW'23: 4th International Workshop on Knowledge Graph Construction, May 28, 2023, Crete, GRE
els.devleeschauwer@ugent.be (E. de Vleeschauwer); x.sittminoo@ugent.be (Sitt Min Oo); ben.demeester@ugent.be (B. De Meester); pieter.colpaert@ugent.be (P. Colpaert)
https://ben.de-meester.org/#me (B. De Meester); https://pietercolpaert.be/#me (P. Colpaert)
0000-0002-8630-3947 (E. de Vleeschauwer); 0000-0001-9157-7507 (Sitt Min Oo); 0000-0003-0248-0987 (B. De Meester); 0000-0001-6917-2167 (P. Colpaert)

and ShExML [4] have taken inspiration from (R2)RML given the timeline and references in the respective papers.

As R2RML aimed at mapping data from relational databases, the expression of a relation between mapping rules in (R2)RML (by a referencing object map) is represented as an operation that secures referential integrity in the resulting knowledge graph (by imposing join conditions). Hence, the concept of securing referential integrity (which is the responsibility of data sources) and linking concepts (which is the responsibility of mapping languages) got conflated.

This conflation has a limiting effect on the usability of relations in mapping files. Join conditions typically result in performance bottlenecks which in turn require joining optimization strategies such as: use predicate-join index tables [5], partition triples maps with joins into disjoint sets to parallelize execution [6, 7], and eliminate self-joins [7]. Moreover, join conditions can lead to loss of additional context: potential triples that are not generated due to a lack of referential integrity across data sources.

In this short research paper we argue that the default link between join conditions and the expression of relations between data sources is too restrictive for the task at hand, namely, the generation of RDF graphs.

We propose a *reference condition* next to the traditional join condition, allowing to express a relation between sources without imposing any referential integrity. We provide a proof-of-concept implementation that replaces the join conditions which do not rely on external information by reference conditions, and can be used as a first step of the knowledge graph generation process by any (R2)RML engine. We evaluate this proof-of-concept implementation and show the positive effects on the generation time when replacing join conditions by reference conditions when this is possible. This change in semantics resulted in exactly the same output when executing GTFS-Madrid-Bench [8], showing the potential impact of this solution in real-world use cases.

After presenting the effects of conflating referential integrity checks with linking concepts during mapping (section 2), we describe the reference condition semantics and implementation (section 3), evaluate (section 4), and conclude (section 5).

2. Motivation

After describing the benefits of being able to specify relations between triples maps (section 2.1), we explain the consequences of their conflation with referential integrity (section 2.2) and the effects of using an alternative mapping construction without explicit relations (section 2.3).

2.1. Benefits of relations

The referencing object map is the only construction in (R2)RML that explicitly expresses the relation between triples maps. It allows using the subjects of another triples map (the parent map) as the objects generated by a predicate-object map (the child map)[1]. As a running example we use a part of the widely used GTFS-Madrid-Bench [8] (fig. 1)[2].

[1]https://rml.io/specs/rml/#logical-join and https://www.w3.org/2001/sw/rdb2rdf/r2rml/#foreign-key
[2]Prefixes are omitted but can be found on https://prefix.cc

```
<#routes> a rr:TriplesMap;
  rml:logicalSource [rml:source "ROUTES.csv";rml:referenceFormulation ql:CSV];
  rr:subjectMap [rr:template "http://transport.ld.es/madrid/metro/routes/{id}";rr:class gtfs:Route];
  rr:predicateObjectMap [
    rr:predicate gtfs:agency;
    rr:objectMap [rr:parentTriplesMap <#agency>;rr:joinCondition[rr:child "agency_id";rr:parent "id"]]
  ].
<#agency> a rr:TriplesMap;
  rml:logicalSource [rml:source "AGENCY.json";rml:referenceFormulation ql:JSONPath;rml:iterator "$.[*]"];
  rr:subjectMap [rr:template "http://transport.ld.es/madrid/agency/{id}";rr:class gtfs:Agency].
```

(a) mapping.ttl

```
id,agency_id,route_short_name
1d,4,name1d
14,5,name14
15,3,name15
```

(b) ROUTES.csv

```
[
    {"id": "4","name": "name4"},
    {"id": "5","name": "name5"}
]
```

(c) AGENCY.json

Figure 1: GTFS-Madrid-Bench referencing object map example

Expressing such relations between triples maps (i) provides relevant metadata and (ii) increases the consistency and maintainability of the mapping file. (i) The relation between triples maps is relevant metadata for the generation of documentation about the related knowledge graph, e.g. deducting a richer SHACL shape from an (R2)RML mapping file [9]). A (visualization of a) SHACL file can help to spot errors in the mapping file or to understand the content of the related knowledge graph. To showcase the importance of this additional metadata (especially for large mappings files), we published diagrams automatically derived from the GTFS-Madrid-Bench mapping file at https://github.com/RMLio/rml-loose-generator/tree/main/diagrams. (ii) Using a referencing object map also ensures that the URI of the parent triples map is only specified in one place: when the developer makes a change to the URI template in the parent triples map, this change applies also to the child map.

2.2. Conflation with referential integrity

According to the (R2)RML specification at least one join condition must be added to the referencing object map if the logical sources of the parent map and the child map are not identical[3]. Due to this join condition the expression of relating triples maps in (R2)RML conflates with securing referential integrity in the knowledge graph. This conflation comes with following disadvantages: (i) a potential loss of knowledge and (ii) a large performance hit. (i) The output of the running example (fig. 2) is limited to seven triples. Although the id of agency 3 is available in the `ROUTES.csv`, the triple expressing the agency of route 15 (fig. 3) will not be generated. However, the missing triple can contain valuable information and can be used to solve queries which do not require additional agency information: generating links between resources, even if you do not have any additional information about them, aligns with the original vision of

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

```
@prefix ex: <http://transport.ld.es/madrid/> .
ex:metro/routes/1d a gtfs:Route; gtfs:agency ex:agency/4 .
ex:metro/routes/14 a gtfs:Route; gtfs:agency ex:agency/5 .
ex:metro/routes/15 a gtfs:Route .
ex:agency/4 a gtfs:Agency .
ex:agency/5 a gtfs:Agency .
```

Figure 2: Output of the running example

```
ex:metro/routes/15 gtfs:agency ex:agency/3 .
```

Figure 3: Triple which is not available in the output

```
<#routes> a rr:TriplesMap ;
  rml:logicalSource [rml:source "ROUTES.csv";rml:referenceFormulation ql:CSV];
  rr:subjectMap [rr:template "http://transport.ld.es/madrid/metro/routes/{id}";rr:class gtfs:Route];
  rr:predicateObjectMap [
    rr:predicate gtfs:agency;
    rr:objectMap [rr:template "http://transport.ld.es/madrid/agency/{agency_id}]].
<#agency> a rr:TriplesMap;
  rml:logicalSource [rml:source "AGENCY.json";rml:referenceFormulation ql:JSONPath;rml:iterator "$.[*]"];
  rr:subjectMap [rr:template "http://transport.ld.es/madrid/agency/{id}";rr:class gtfs:Agency].
<#agency2> a rr:TriplesMap;
  rml:logicalSource [rml:source "ROUTES.csv";rml:referenceFormulation ql:CSV];
  rr:subjectMap [rr:template "http://transport.ld.es/madrid/agency/{agency_id}";rr:class gtfs:Agency].
```

Figure 4: Crafted URI templates as alternative

designing RDF (”Anyone Can Make Statements About Any Resource”[4]). Use cases illustrating the value of those missing triples are: mapping sample data, at the beginning of a project or for testing a mapping file during the development process, and mapping data streams, where not all data is simultaneously available. (ii) The performance hit when joining and checking referential integrity is substantial, and is one of the reasons why the referential integrity constraint is typically dropped in data warehouses [10].

2.3. Alternative without relations

Another way to relate triples is via carefully crafted URI templates (fig. 4): as long as the URI templates are consistent between an object map of a first triples map and a subject map of a second triples map, the nodes will match and triples will be related. This way of mapping related triples does not involve any referential integrity checking and avoids performance bottlenecks. Additionally, the notation is less verbose. However, the synchronization of the mapping rules becomes a manual process, and to get similar relevant metadata from the mapping file as with a referencing object map, an additional triples map specifying the class of the object of the

[4]https://www.w3.org/TR/rdf-concepts/#section-anyone

first triples map is needed (triples map `<#agency2>`). Consequently, the manual synchronization efforts increase and the notation is again more verbose.

3. Reference conditions

We introduce a *reference condition* for a referencing object map as alternative to the traditional join condition. A reference condition allows the use of a subject map of another logical source without imposing any referential integrity: the object of the child map is generated following the structure of the subject of the parent map without verifying or using data from the parent source. After describing its scope (section 3.1), we propose a language construct (section 3.2) and a proof-of-concept implementation (section 3.3).

3.1. Scope

Not every join condition can be re-interpreted as a reference condition. We identify two cases for which join conditions remain needed: (i) data from the parent source is needed to build the child object URI, or (ii) the parent source is a selective dataset. Referring back to our running example, case (i) applies if the template of an agency uses the agency name: "`http://transport.ld.es/madrid/agency/{agency_name}`". Case (ii) applies when the parent source `AGENCY.json` is a selective dataset, e.g. there are two kinds of agencies, listed in two logical sources with each their own triples map and URI template. In that case the triple of fig. 3 would be wrong if agency 3 does not appear in AGENCY.json (fig. 1c).

3.2. Language Construct

The introduction of a reference condition (fig. 5) would require following adaptations to the (R2)RML specifications (emphasizing our proposed changes in bold)[5]. (i) A referencing object map allows using **the subject maps** of another triples map **for generating the objects of** a predicate-object map. (ii) If the logical source of the child triples map and the logical source of the parent triples map of a referencing object map are not identical, then the referencing object map must have at least **one condition, where all conditions are either reference conditions or join conditions**. (iii) The joint SQL query of a referencing object map is: [...] **If the referencing object map has at least one reference condition:** `SELECT child-column1 AS parent-column1, child-column2 AS parent-column2, ... FROM ({child-query}) AS tmp.`

If a reference condition is used and the URI template requires data from the parent source (section 3.1 case (i)), the mapping engine should throw an error, similar to errors thrown when an unknown reference is used in the mapping file (e.g. test case R2RMLTC0002c[6]).

3.3. Proof-of-concept Implementation

Reference conditions enable the deduction of crafted URI templates, which can be handled by default by any (R2)RML mapping engine. This allows introducing reference conditions without

[5]https://rml.io/specs/rml/#logical-join and https://www.w3.org/2001/sw/rdb2rdf/r2rml/#foreign-key
[6]https://www.w3.org/2001/sw/rdb2rdf/test-cases/#R2RMLTC0002c

```
<#routes> a rr:TriplesMap ;
  rml:logicalSource [rml:source "ROUTES.csv";rml:referenceFormulation ql:CSV];
  rr:subjectMap [rr:template "http://transport.ld.es/madrid/metro/routes/{id}";rr:class gtfs:Route];
  rr:predicateObjectMap [
    rr:predicate gtfs:agency;
    rr:objectMap [rr:parentTriplesMap <#agency>;rr:refCondition [rr:child "agency_id";rr:parent "id"]]
  ].
<#agency> a rr:TriplesMap;
  rml:logicalSource [rml:source "AGENCY.json";rml:referenceFormulation ql:JSONPath;rml:iterator "$.[*]"];
  rr:subjectMap [rr:template "http://transport.ld.es/madrid/agency/{id}";rr:class gtfs:Agency].
```

Figure 5: Alternative mapping file, with a reference condition

adding any overhead to the existing (R2)RML mapping engines. We published a proof-of-concept implementation converting referencing object maps with reference conditions to crafted URIs at https://github.com/RMLio/rml-loose-generator.

At this moment our implementation interprets every referencing object map that does not need data from the parent source with reference conditions instead of join conditions (i.e. it does not take case (ii) as described in section 3.1 into account) and is limited to referencing object maps with only one join condition, without use of functions, and where the subject of the parent is built with a template. Outside of these restrictions, the original semantics of join conditions are preserved.

4. Evaluation

We tested the effect of using reference conditions versus join conditions on GTFS-Madrid-Bench [8]. We used our proof-of-concept implementation to re-interpret all relevant join conditions from the GTFS-Madrid-Bench mappings as reference conditions, before sending the adapted mapping file to RMLMapper, RMLStreamer and Morph-KGC[7]. Although unanticipated, our implementation correctly detected *every* join condition as reference condition. We conclude that none of the joins in the GTFS-Madrid-Bench mappings require the use of data from the parent source (section 3.1 case (i)).

We tested these mapping engines for scales 1, 5, 50, and 100 (table 1), using a device with following specifications: 2 x Hexacore Intel E5645 (2.4GHz) CPU, 24GB RAM, 1x 250GB harddisk. RMLMapper and RMLStreamer cannot generate any output for the GTFS-Madrid-Bench within one hour when using join conditions. Using reference conditions, however, these mapping engines were able to generate correct output, with timings similar to using a state-of-the-art mapping engine like Morph-KGC. We note that RMLMapper cannot handle GTFS scales 50 and 100: the RMLMapper loads all data in memory during mapping, and the testing device ran out of memory during GTFS scales 50 and 100.

When comparing the resulting knowledge graphs from GTFS-Madrid-Bench we found *no difference* between the version generated with join conditions and the one generated with

[7] https://github.com/RMLio/rmlmapper-java, https://doi.org/10.5281/zenodo.3887065, and https://doi.org/10.5281/zenodo.5543552, respectively.

Table 1
Comparison of the knowledge graph generation time in seconds when using mapping files with reference conditions versus mapping files with join conditions, average of three runs, time out after 1 hour.

		GTFS scale 1	GTFS scale 5	GTFS scale 50	GTFS scale 100
RMLMapper	join conditions	-	-	-	-
	reference conditions	**6**	**18**	-	-
RMLStreamer	join conditions	-	-	-	-
	reference conditions	25	33	**136**	**229**
Morph-KGC	join conditions	13	20	163	746
	reference conditions	12	22	155	653

reference conditions. In hindsight, this makes sense: GTFS-Madrid-Bench's testing data is also used to fill a relational database, where referential integrity is needed. We conclude that data joined in the GTFS-Madrid-Bench has perfect referential integrity and that none of the join conditions in the GTFS-Madrid-Bench mapping file impose any limitation to replacing it by a reference condition.

5. Conclusion

The referencing object map construct from (R2)RML implies that the resulting RDF graph must be referential integer, limiting the usability of relations in mapping files. Meanwhile, in use cases where referential integrity is needed, the data sources typically already validate this integrity by design (e.g. in the case of relational databases). As such, referential integrity checks during the mapping process can become redundant. For other use cases, e.g., mapping data sources that are not under the same governance, or mapping incomplete data (e.g., sample data or streaming data), dropping the notion of referential integrity can improve the completeness of the resulting knowledge graph.

With this paper we introduce reference conditions, an additional condition type that enables the expression of relations between mapping rules without joining the related data sources and imposing referential integrity on the resulting triples. With a very limited overhead, the use of reference conditions can accomplish an instant improvement for any RML or R2RML mapping engine on knowledge graph generation time, while the mapping file still remains consistent and well maintainable, and supplies the valuable metadata which can be deducted from expressed relations. We argue that, unless data from the parent source is needed to build the child object URI, or the parent source is a selective dataset, a reference condition should be preferred over a join condition because this construction fits the Open World Assumption supported by RDF.

Due to the completely correct generation within the GTFS-Madrid-Bench, we believe that many use cases can benefit from this additional condition type, embracing the Open World Assumption followed by performance gain without any significant side-effects.

For future work, we will research how to further optimize reference conditions. First, our current implementation only takes equality operators into account, however, we can expand this to other types of reference conditions (e.g. comparing the lowercased values between sources) by transforming the child reference. Second, we will verify if we can achieve similar performance gains with other benchmarks and other mapping engines.

Acknowledgments

The described research activities were supported by SolidLab Vlaanderen (Flemish Government, EWI and RRF project VV023/10) and the imec ICON project AI4Foodlogistics (Agentschap Innoveren en Ondernemen project nr. HBC.2020.3097). The authors want to thank the KGCW reviewers for their constructive feedback that helped improve the paper.

References

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

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

[3] M. Lefrançois, A. Zimmermann, N. Bakerally, A SPARQL extension for generating RDF from heterogeneous formats, in: The Semantic Web 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 – June 1, 2017, Proceedings, Portoroz, Slovenia, 2017, pp. 35–50. doi:10.1007/978-3-319-58068-5_3.

[4] H. García-González, I. Boneva, S. Staworko, J. E. Labra-Gayo, J. M. C. Lovelle, ShExML: improving the usability of heterogeneous data mapping languages for first-time users, PeerJ Computer Science 6 (2020) e318.

[5] E. Iglesias, S. Jozashoori, D. Chaves-Fraga, D. Collarana, M.-E. Vidal, 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, 2020. doi:10.1145/3340531.3412881.

[6] E. Iglesias, S. Jozashoori, M.-E. Vidal, Scaling Up Knowledge Graph Creation to Large and Heterogeneous Data Sources, arXiv:2201.09694 [cs] (2022). ArXiv: 2201.09694.

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

[8] 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.

[9] T. Delva, B. D. Smedt, S. M. Oo, D. V. Assche, S. Lieber, A. Dimou, RML2shacl: RDF generation taking shape, in: Proceedings of the 11th on Knowledge Capture Conference, 2021, pp. 153–160. doi:10.1145/3460210.3493562.

[10] W. McKnight, Data quality, in: Information Management, Elsevier, 2014, pp. 32–43. doi:10.1016/B978-0-12-408056-0.00004-7.