Supporting Relational Database Joins for Generating Literals in R2RML

Christophe Debruyne[1]

[1]*Montefiore Institute, University of Liege, 4000 Liège, Belgium*

Abstract

Since its publication, R2RML has provided us with a powerful tool for generating RDF from relational databases. R2RML has its limitations, which are being recognized by W3C's Knowledge Graph Construction Community Group. That same group is currently developing a specification that supersedes R2RML in terms of its functionalities and the types of resources it can transform into RDF–primarily hierarchical documents. The community has a good understanding of problems of relational data and documents, even if they might need to be approached differently because of their different formalisms. This paper presents a challenge that has not been addressed yet for relational databases–generating literals based on (outer-)joins. We propose a simple extension of the R2RML vocabulary and extend the reference algorithm to support the generation of literals based on (outer-)joins. Furthermore, we implemented a proof-of-concept and demonstrated it using a dataset built for benchmarking joins. While it is not (yet) an extension of RML, this contribution informs us on how to include such support and how it allows us to create self-contained mappings rather than relying on less elegant solutions.

Keywords

R2RML, Knowledge Graph Generation, Outer-joins, Joins

1. Introduction

R2RML [1] is a powerful technique for transforming data from relational databases into RDF and was published almost a decade ago. Since then, it HAS inspired many initiatives to generalize this approach to other types of data sources, such as RML [2] and xR2RML [3]. Others looked at extending aspects of (R2)RML not pertaining to the sources being transformed but to tackle unaddressed challenges and requirements such as RDF Collections [3, 4] and functions [5, 6].

The R2RML Recommendation specified a reference algorithm in which relational joins (natural joins or equi-joins, to be specific) can be used to relate resources. The implementation can be broken into two parts: (1) the generation of triples based on a triples map tm_1 related to a logical source, and (2) the generation of triples relating subjects from tm_1 with those of another triples map tm_2. While (2) does not use an outer-join, the combination of both (1) and (2) ensures that the data being transformed "behaves" as the result of an outer-join. The problem, however, is that support for such outer-joins is only limited to resources; there is no *convenient* way to do something similar for literals.

Third International Workshop On Knowledge Graph Construction Co-located with the ESWC 2022, 30th May 2022, Crete, Greece

c.debruyne@uliege.be (C. Debruyne)
0000-0003-4734-3847 (C. Debruyne)

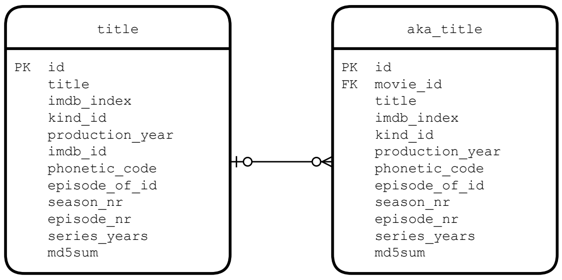

Figure 1: The tables `title` and `aka_title` of the database. A title may be related to one or more `aka_titles`, and an `aka_title` may be related to one `title`.

This paper proposes a simple extension of R2RML for supporting joins for the generation of literals and proposes how the reference algorithm could be extended. We demonstrate this extension using a fairly big relational database developed for benchmarking joins [7]. This benchmark provides us also with a realistic case, motivating the need for such an extension. Furthermore, this paper positions this contribution with other initiatives developed by the Knowledge Graph Generation community to open a discussion.

2. The Problem

We framed the problem in the previous section. This section will rephrase the problem and discuss several approaches to achieve the desired result that one can observe in practice. To this end, we will be using a running example based on the database developed by [7].

To benchmark the performance of joins, [7] developed a database based on the Internet Movie Database[1] (IMDb). In short, their motivation was that existing synthetic benchmarks might be biased, whereas real and "messy" provided better grounds for comparison. While that aspect is not important for this paper, the database they developed did contain two big tables: `title` containing information about movies and their titles, and `aka_title` containing variations in titles (either an alternative title or titles in different languages). Figure 1 depicts the relation between the two tables and their attributes.[2]

How can one use R2RML to relate instances of movies stored in the table `title` with their titles from `title` and their alternative titles from `aka_title`, which requires one to apply a left outer join between `title` and from `aka_title`? There are two approaches to achieving this with R2RML:

Sol1 The first is the creation of two triples maps with one dedicated to the generation of triples for the outer-join.

[1]https://www.imdb.com/

[2]The files were loaded into a MySQL database but required some minor pre-processing: a handful of encoding issues in the files and NULL values in `aka_title` were represented with the number 0. We also introduced a foreign key constraint that was not present in the SQL schema provided by [7]. The reason is that the foreign key constraint optimizes joins on these two tables. The tables contain 2528312 and 361472 records, respectively. There are 93 records from `aka_title` not referring to a record in `title` and 2322682 records in `title` have no alternative titles.

Sol2 The second is using one triples map with a (outer-)join in its logical table.

The problem with Sol1 is that the mapping is not self-contained and that there are two distinct triples maps that need to be maintained.[3] One also must document that this construct was necessary to facilitate this outer-join. The advantage is that there are two distinct processes for querying the underlying database and thus less overhead. And while Sol2 makes the triples map self-contained, it may require the processor to process many logical rows that generate the same triples when the first table has multiple matches with the second. There may be many NULL values in the result set when there are few matches.

Practitioners familiar with RDF may lean towards the first solution as they may think in terms of the generated RDF; they know that the triples generated via these ”disjointed” triples maps will be ”merged” in the output. Practitioners familiar with SQL may naturally lean towards the second solution.

In the next section, we propose a small extension of R2RML to provide support for joins on literal values.

3. Proposed solution

In Listing 1, we demonstrate the extension. It introduces the predicate `rrf:parentLogicalTable`.[4] The domain of that predicate is `rr:RefObjectMap` and the range is `rr:LogicalTable`. Our extension requires that a `rr:RefObjectMap` must have either a `rrf:parentLogicalTable` or `rr:parentTriplesMap`. A referencing object map may now also generate literals. Where necessary, we will refer to object maps with a parent-triples map as “regular” referencing object maps.

```
<#title>
    rr:logicalTable [ rr:tableName "title" ] ;
    rr:subjectMap [ rr:template "http://data.example.com/movie/{id}" ; rr:class ex:Movie; ] ;
    rr:predicateObjectMap [ rr:predicate ex:title ; rr:objectMap [ rr:column "title" ] ; ] ;
    rr:predicateObjectMap [
        rr:predicate ex:title ;
        rr:objectMap [
          rr:column "title" ;
          rrf:parentLogicalTable [ rr:tableName "aka_title" ] ;
          rr:joinCondition [ rr:child "id" ; rr:parent "movie_id" ] ;
        ] ;
    ] ;
```

Listing 1: Using parent-logical tables for managing joins

The reference algorithm[5] is extended as follows: step 6 will now iterate over all referencing object maps with a `rr:parentTriplesMap`, and we add a 7th step for each referencing object

[3]We may observe cases of the first also being conducted for referencing object maps, especially when the processor used uses the reference algorithm. The problems with respect to the ”self-containedness” of triples maps still hold. An R2RML processor may internally “rewrite” referencing object maps as triples maps to optimize the process. [8], for instance, analyzes the attributes that are referenced to apply a projection to the source data.

[4]The namespace `rrf` refers to the namespace used in [6].

[5]https://www.w3.org/TR/r2rml/#generated-rdf

map that uses a parent-logical table. The steps for generating are mostly the same. The differences are:

1. the column names used for the parent SQL query are the union of the attributes mentioned in the join conditions and the columns referred to in the term map[6];
2. it may generate a term of any term type; and
3. the column names referred to by the object map are those of the parent.

In other words, if both logical tables share a column X, then a reference to X would be to that of the parent. This behavior is consistent with that of regular referencing object maps.

An implementation of this algorithm is made available.[7] We have chosen to extend R2RML-F as it implements R2RML's reference algorithm.

4. Demonstration

We now present a limited experiment comparing the performance of Sol1, Sol2, and our proposal using the relational database introduced in Section 2. The mappings for Sol1 and Sol2 are in Appendix A. In this experiment, we join using the tables as a whole. As R2RML requires result sets to have unique names for each column, we created a third table `aka_title2`, where each column received the suffix '2'. We also created a foreign key from `aka_title2` to `title`. We wanted to avoid using subqueries to rename the columns, and these may become materialized and thus have an unfairly negative impact on the outcome.

We ran the experiment on a MacBook Pro with a 2.3 GHz Dual-Core Intel Core i5 processor and 16 GB 2133 MHz LPDDR3 RAM. The database was stored in a MySQL 8.0 database using MyISAM tables in a Docker container. We compiled both R2RML-F and the script for our experiment with Java 17.0.2. When invoked, the Java Virtual Machine only loads the classes from libraries (JARs). I.e., Java adopts lazy loading, which negatively impacts the first execution of the mapping. We, therefore, execute a mapping once to ensure all classes are loaded and then run each mapping 30 times in a loop. We explicitly call the garbage collector between executions.

The code calls upon the extension of R2RML-F and registered timestamps before and after executing the mapping. We have not registered the time for writing the graph onto the hard disk.

Table 1 provides the descriptive statistics of the time it took to process each mapping that corresponds with Sol1, Sol2, and our proposed solution. The data we have collected is listed in Appendix B.

From Figure 2, which shows the average run times in seconds, it is clear that the approach of using two different triples maps (Sol1) is faster than the two other approaches, which comes as no surprise. The problem, however, is that we have two distinct triples maps and their relationship is not explicit. Placing the outer-join in the logical table (Sol2) has the worst

[6]One column name in case of a `rr:column`, a set of column names in case of a `rr:template` and an empty set in case of a `rr:constant`. In the case of a `rr:constant`, the algorithm will generate constants based on the matches of the join query.

[7]https://github.com/chrdebru/r2rml/tree/r2rml-join

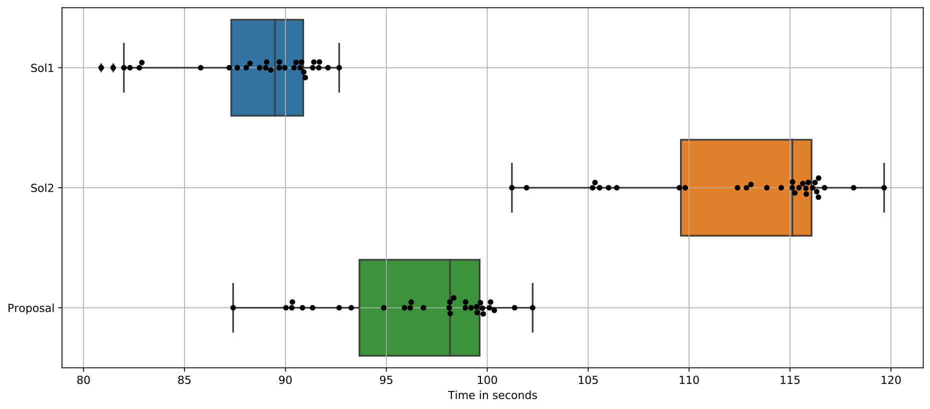

Figure 2: Time taken to process three mappings: Sol1–2 triples maps for the outer-join, Sol2–one triples map with the outer-join in the logical table, and our proposed solution.

	Sol1	Sol2	Proposal
count	30.000000	30.000000	30.000000
mean	88.332720	112.595483	96.616708
std	3.548997	5.047198	4.004154
min	80.873969	101.222406	87.411147
25%	87.317585	109.592302	93.666924
50%	89.485543	115.118579	98.156296
75%	90.884436	116.067260	99.623490
max	92.665459	119.661447	102.249416

Table 1
Descriptive statistics about the time taken to process the three approaches and their mappings: Sol1–2 triples maps for the outer-join, Sol2–one triples map with the outer-join in the logical table, and our proposed solution.

performance. The outer join yields a result set with 155749 more records than the referred table and contains twice the number of attributes. The overhead can be significantly reduced by only selecting the columns of interest[8], but the three mappings refer to the logical tables as a whole. Unsurprisingly, our solution is less efficient than Sol1 but considerably more efficient than Sol2.

The three groups do not follow a normal distribution.[9] To test whether there is a statistically significant difference between the medians of the three groups, we apply the Kruskal-Wallis Test. This test can be applied to groups that do not fit a normal distribution. Given that the p-value $1.4500079774576953 \times 10^{-16} \leq 0.05$, we reject the test's null hypothesis that the median is equal across all groups.

[8] [8] presented an approach to do this automatically.
[9] All three groups are skewed and fit an Asymmetric Laplace distribution the best.

We may conclude from these initial results that the proposed solution is not only a viable solution in terms of performance. More importantly, our proposal ensures that the mappings remain self-contained. While performance is crucial in knowledge graph generation, we argue that our proposed vocabulary extension is crucial. An R2RML processor may rewrite referencing object maps (both types) into distinct triples maps to optimize the process.

5. Discussion

In this paper, we extended the concept of `rr:RefObjectMap` to support joins for literal values. The reference algorithm for R2RML processes these in a separate loop for the generation of relations between subjects of two triples maps. Our approach added a similar step to the generation of literals based on a join. One may ask whether this approach may be adopted for term maps in general. The generation of subjects, predicates, and graphs for relational databases is based on a logical row. Generalizing this approach for such term maps may require a join per row, which is not efficient and is thus best done in the logical table of a triples map.

As we can generate resources with our approach, one can question whether the notion of parent-triples maps is still necessary. The reference algorithm uses both logical tables, even though a processor can only select those used by the subject maps. The question arises: do we refer to (data in) sources, or do we refer to triples maps?

Related to this work is the approach proposed by [9], where they proposed ”fields” to manipulate and even combine the source before generating RDF. Their work, demonstrated with hierarchical data, aimed to address the problem of references that may yield multiple results and that sources may contain data of mixed formats. They also introduced an abstraction allowing one to retrieve information via a reference that does not depend on the underlying reference formulation. To the best of our knowledge, support for relational databases and the addition of fields from different tables has not yet been published. However, as they declare fields on the logical source, such an approach may boil down to a situation similar to Sol2 mentioned in Section 2.

This study assumed that we did not store our values in SQL arrays or JSON. These may provide another solution, but processing those data types is not within R2RML’s capabilities. And if the data is not stored in such columns, one has to write an SQL query to create those for the logical table. There are additional challenges with that approach: one is how to combine different representations (for which the work presented in [9] may be useful), and the other is the processing of multi-valued term maps. The latter is being discussed in the Knowledge Graph Construction Community Group.

5.1. Limitations

With this paper, we aimed to propose an idea for extending R2RML, and we conducted a small experiment to demonstrate the viability of our approach. There are some limitations concerning the experiment.

First, we have chosen to adopt a benchmark for SQL joins for this paper. We could have considered adopting other benchmarks. [10] developed an approach to assess the variables that affect RML implementations using synthetic data stored as CSV files. Their framework may have provided insights as R2RML-F as implementation as a whole (even though it is not an RML engine). However, their work may offer us guidelines for choosing datasets with particular characteristics for future experiments (join duplicates, dataset size, etc.).

Secondly, we have chosen to declare a foreign key between the two tables. It may be interesting to what the impact of omitting that foreign key may be. We believe, however, that this would provide us with information about the underlying database instead of our proposed solution.

Finally, we ran each mapping 30 times and used one particular implementation of R2RML. The extended implementation follows R2RML's reference implementation, even though an R2RML engine may process it differently (and even more efficiently). However, this paper aims to start a discussion on the viability of our approach and, more importantly, the advantages of extending the vocabulary for SQL joins for literals.

6. Conclusions

We addressed the problem of generating literals from an outer-join, which R2RML does not support. While interesting initiatives are proposed for mostly hierarchical documents, we wanted to address this problem for relational databases by extending R2RML. We proposed a small extension with few implications regarding the R2RML vocabulary. We also extended the reference algorithm and provided an implementation that we have analyzed in an experiment.

From this paper, we can conclude that, for relational databases, our approach is a viable solution. While not as efficient as disjoint triples maps, it may be worth considering not as an approach. In our approach, the mappings are self-contained, and the relationship between the two logical tables is thus explicit. It is essential not to consider this vocabulary extension as syntactic sugar, as that would imply it is shorthand for something semantically equivalent.

We have addressed this problem for relational databases and R2RML. We could envisage that such an approach could be part of RML, which has the ambition to supersede R2RML. How this approach would work for non-relational data is to be studied.

Acknowledgments

We thank Pano Maria, Jhon Toledo Barreto, Hannes Voigt, and Herminio García-González for their valuable and constructive reviews.

References

[1] S. Das, R. Cyganiak, S. Sundara, R2RML: RDB to RDF Mapping Language, 2012. URL: https://www.w3.org/TR/r2rml/.

[2] A. Dimou, M. V. Sande, P. Colpaert, R. Verborgh, E. Mannens, R. V. de Walle, RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data, in: C. Bizer, T. Heath, S. Auer, T. Berners-Lee (Eds.), Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014),

Seoul, Korea, April 8, 2014., volume 1184 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2014. URL: http://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf.

[3] F. Michel, L. Djimenou, C. Faron-Zucker, J. Montagnat, Translation of relational and non-relational databases into RDF with xr2rml, in: V. Monfort, K. Krempels, T. A. Majchrzak, Z. Turk (Eds.), WEBIST 2015 - Proceedings of the 11th International Conference on Web Information Systems and Technologies, Lisbon, Portugal, 20-22 May, 2015, SciTePress, 2015, pp. 443–454. doi:10.5220/0005448304430454.

[4] C. Debruyne, L. McKenna, D. O'Sullivan, Extending R2RML with support for RDF collections and containers to generate MADS-RDF datasets, in: J. Kamps, G. Tsakonas, Y. Manolopoulos, L. S. Iliadis, I. Karydis (Eds.), Research and Advanced Technology for Digital Libraries - 21st International Conference on Theory and Practice of Digital Libraries, TPDL 2017, Thessaloniki, Greece, September 18-21, 2017, Proceedings, volume 10450 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 531–536. doi:10.1007/978-3-319-67008-9_42.

[5] B. De Meester, W. Maroy, A. Dimou, R. Verborgh, E. Mannens, Declarative data transformations for linked data generation: The case of dbpedia, in: E. Blomqvist, D. Maynard, A. Gangemi, R. Hoekstra, P. Hitzler, O. Hartig (Eds.), The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part II, volume 10250 of *Lecture Notes in Computer Science*, 2017, pp. 33–48. doi:10.1007/978-3-319-58451-5_3.

[6] C. Debruyne, D. O'Sullivan, R2RML-F: towards sharing and executing domain logic in R2RML mappings, in: S. Auer, T. Berners-Lee, C. Bizer, T. Heath (Eds.), Proceedings of the Workshop on Linked Data on the Web, LDOW 2016, co-located with 25th International World Wide Web Conference (WWW 2016), volume 1593 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2016. URL: http://ceur-ws.org/Vol-1593/article-13.pdf.

[7] V. Leis, A. Gubichev, A. Mirchev, P. A. Boncz, A. Kemper, T. Neumann, How good are query optimizers, really?, Proc. VLDB Endow. 9 (2015) 204–215. doi:10.14778/2850583.2850594.

[8] S. Jozashoori, M. Vidal, Mapsdi: A scaled-up semantic data integration framework for knowledge graph creation, in: H. Panetto, C. Debruyne, M. Hepp, D. Lewis, C. A. Ardagna, R. Meersman (Eds.), On the Move to Meaningful Internet Systems: OTM 2019 Conferences - Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21-25, 2019, Proceedings, volume 11877 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 58–75. doi:10.1007/978-3-030-33246-4_4.

[9] T. Delva, D. Van Assche, P. Heyvaert, B. De Meester, A. Dimou, Integrating nested data into knowledge graphs with RML fields, in: D. Chaves-Fraga, A. Dimou, P. Heyvaert, F. Priyatna, J. F. Sequeda (Eds.), Proceedings of the 2nd International Workshop on Knowledge Graph Construction co-located with 18th Extended Semantic Web Conference (ESWC 2021), Online, June 6, 2021, volume 2873 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2873/paper9.pdf.

[10] D. Chaves-Fraga, K. M. Endris, E. Iglesias, Ó. Corcho, M. Vidal, What are the parameters that affect the construction of a knowledge graph?, in: H. Panetto, C. Debruyne, M. Hepp, D. Lewis, C. A. Ardagna, R. Meersman (Eds.), On the Move to Meaningful Internet Systems: OTM 2019 Conferences - Confederated International Conferences: CoopIS, ODBASE,

C&TC 2019, Rhodes, Greece, October 21-25, 2019, Proceedings, volume 11877 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 695–713. doi:10.1007/978-3-030-33246-4_43.

A. Mappings Used in the Experiment

```
### MAPPING USED FOR SOL1 IN THE EXPERIMENT
<#title_tm>
    rr:logicalTable [ rr:tableName "title" ] ;
    rr:subjectMap [ rr:template "http://data.example.com/movie/{id}" ; rr:class ex:Movie; ] ;
    rr:predicateObjectMap [ rr:predicate ex:title ; rr:objectMap [ rr:column "title" ] ; ] .
<#aka_title_tm>
    rr:logicalTable [ rr:tableName "aka_title" ] ;
    rr:subjectMap [ rr:template "http://data.example.com/movie/{movie_id}" ; rr:class ex:Movie; ] ;
    rr:predicateObjectMap [ rr:predicate ex:title ; rr:objectMap [ rr:column "title" ] ; ] .

### MAPPING USED FOR SOL2 IN THE EXPERIMENT
<#title_tm>
    rr:logicalTable [
      rr:sqlQuery "SELECT * FROM title t LEFT OUTER JOIN aka_title2 a ON t.id = a.movie_ID2" ] ;
    rr:subjectMap [ rr:template "http://data.example.com/movie/{id}" ; rr:class ex:Movie; ] ;
    rr:predicateObjectMap [ rr:predicate ex:title ; rr:objectMap [ rr:column "title" ] ;
        rr:objectMap [ rr:column "title2" ] ;
```

B. Data

This table presents the data we collected. It represents the time (seconds) it took to transform the tables into RDF. The time for reading the R2RML mapping, preprocessing the R2RML mapping, and writing the files onto the disk is not considered.

Sol1	Sol2	Proposal	Sol1	Sol2	Proposal
82.767074	105.217382	91.348131	81.999299	106.416826	90.347002
90.427724	109.809143	93.264663	87.217652	112.399161	95.900177
88.725155	109.520021	96.184195	88.061773	113.065328	96.838843
89.692226	112.841730	96.235368	91.653757	115.125541	99.763734
90.985728	116.415721	98.115005	90.911754	118.151595	100.106114
90.802484	115.789062	99.497830	87.617385	115.111617	98.930498
89.970738	113.851943	102.249416	91.687230	115.240347	99.203136
89.278861	116.240366	99.665376	91.411960	119.661447	99.472763
92.111467	116.408101	100.164253	89.072998	116.122525	98.916101
89.702658	114.564878	99.801730	90.528770	115.901463	98.147563
90.727826	116.322650	98.165029	89.027501	115.633416	101.355876
91.352994	115.445335	94.873705	88.246109	115.808912	98.338438
92.665459	116.712422	100.349824	82.890711	105.337932	87.411147
81.469006	101.222406	90.316647	85.803827	101.949639	92.664208
82.297501	106.010150	90.027944	80.873969	105.567430	90.846512