Re-Construction Impact on Metadata Representation Models

Ana Iglesias-Molina
ana.iglesiasm@upm.es
Ontology Engineering Group,
Universidad Politécnica de Madrid
Boadilla del Monte, Madrid, Spain

Jhon Toledo
ja.toledo@upm.es
Ontology Engineering Group,
Universidad Politécnica de Madrid
Boadilla del Monte, Madrid, Spain

Oscar Corcho
oscar.corcho@upm.es
Ontology Engineering Group,
Universidad Politécnica de Madrid
Boadilla del Monte, Madrid, Spain

David Chaves-Fraga
david.chaves@usc.es
Grupo de Sistemas Intelixentes
Universidade de Santiago de Compostela
Santiago de Compostela, Galicia, Spain

ABSTRACT

Reification in knowledge graphs has been present since the inception of RDF to allow capturing additional information in triples, usually metadata. The need of adopting or changing a metadata representation in a pre-existing graph to enhance the knowledge capture and access can lead to inducing complex structural changes in the graph, according the target representation's schema. In these situations, it is necessary to decide whether to construct the knowledge graph again from its original sources, or to re-construct it using the current version of the graph. In this paper we conduct an empirical study to analyze which re-construction approach is more suitable for switching the representation approach from the created graph ensuring that the additional represented knowledge is preserved. We study four well-known metadata representations, using mapping languages to construct the graph, and SPARQL `CONSTRUCT` queries to re-construct it. With this work we aim to provide insights about the impact of re-construction on metadata representations interoperability and the implications of different approaches.

CCS CONCEPTS

• **Information systems** → Resource Description Framework (RDF); Graph-based database models; *Data exchange*; **Data model extensions**;

KEYWORDS

Knowledge Graphs, Metadata, SPARQL, Declarative Mappings.

ACM Reference Format:
Ana Iglesias-Molina, Jhon Toledo, Oscar Corcho, and David Chaves-Fraga. 2023. Re-Construction Impact on Metadata Representation Models. In *Knowledge Capture Conference 2023 (K-CAP '23), December 5–7, 2023, Pensacola, FL, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3587259.3627554

1 INTRODUCTION

Knowledge graphs (KG) have gained popularity in recent years for integrating and publishing knowledge on the web [22]. KGs are modelled according to schemes that are not immutable, as they are subject to modifications triggered by changes in the domain of knowledge or in the consumption needs of downstream tasks. An example is the need to incorporate additional knowledge about a triple, which is known as statement reification. Reification is usually used to include metadata and provenance to existing triples [15].

Throughout the years, different metadata representations have been proposed for RDF, such as Named Graphs [10], N-Ary Relationships [33] or RDF-star [18]. For instance, Nanopublications [17] adopt a Named Graph-based model to publish minimal scientific statements along with their provenance and associated context. Meanwhile, in ontology engineering, N-Ary Relationships are widely used as an ontology design pattern [16]. It is not uncommon for an existing KG to adopt one of these representations, e.g., the publication of the DisGeNet KG as Nanopublications [35]; or the incorporation of *qualifiers* in the Wikidata model [14].

When such graph re-structuration is needed, the following question arises: Is it more convenient to construct the graph from the original input data sources again, or to re-construct it within the triplestore? Previous studies report on analysis of querying these representations [15, 21, 25, 34], but not on their re-construction. In this paper, we analyze how re-construction approaches behave with metadata representation models, and the parameters that influence their choice. To this end, we perform a comprehensive empirical study of this re-construction in four representations (Standard Reification [30], N-Ary Relationships [33], Named Graphs [10] and RDF-star [18]) using (i) declarative mappings to construct each representation with KG construction systems and (ii) transformation per peers of representations with SPARQL `CONSTRUCT` queries in different triplestores. We aim to unravel the aspects that influence the interchange and interoperability of metadata representations, to help in the decision on how to perform switch of models without information loss. We also raise awareness of recently overlooked aspects of `CONSTRUCT` queries, while contributing to research on the impact of metadata representation sustainability.

The remainder of this paper is structured as follows: Section 2 motivates the work with an example; Section 3 presents the experimental setup, queries and datasets used in the evaluation; Section 4

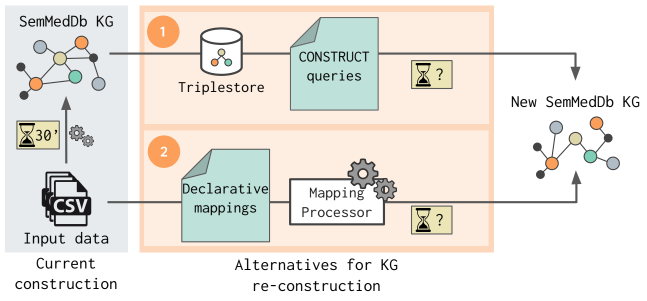

Figure 1: Alternatives for re-constructing a pre-existing KG.

shows the results obtained to test reconstruction with mapping engines and triplestores; Section 5 provides insights and discusses the obtained results; Section 6 presents related works; and Section 7 concludes the paper with final remarks and future steps.

2 MOTIVATING EXAMPLE

Consider the Semantic MEDLINE Database (SemMedDB) [28], that contains entities extracted from biomedical texts, the semantic types of these concepts and a timestamp of when they were extracted. A stable version of a knowledge graph represents this metadata, where the timestamp and semantic type are related to its correspondent entity with an n-ary relationship. However, the maintainers of this knowledge graph want to reduce the size of the graph by reducing the number of nodes, as well as to represent this information with RDF-star to start adopting the RDF 1.2 specification.

The graph maintainers have access to the original source data, so they can change the original KG construction pipeline, that uses declarative mappings to construct the graph. However, since this graph is not going to be updated with new data, it is also possible to change the structure of the graph with queries within the triplestore where it is maintained. They know the time it takes to generate the graph in its original representation. However, they are not certain whether generating the graph in the new representation using this pipeline will take more time, or if with queries the transformation could be performed faster without information loss (Fig. 1); and which representations can be generated with each approach. Hence, the objective of this work consist of assessing the aspects that influence this situation to help in making an informed decision.

3 METHODOLOGY

In this section we present the methodology followed to analyze how different approaches for re-constructing knowledge graphs impact the metadata representations interchange. More in detail, we compare two ways of re-constructing a KG to change its representation, (i) with KG construction technologies from scratch (e.g., RML [27]) and (ii) with `CONSTRUCT` queries in a triplestore from the previous version of the graph. We aim to answer the following research questions: **(RQ1)** In which cases is it more efficient in terms of time to either construct a reified knowledge graph from the original data sources or to re-construct it within a triplestore? **(RQ2)** Which of these two approaches is more scalable as the size of the data increases? **(RQ3)** How does the behavior of these approaches change with the metadata representation models? All resources to reproduce the experiments are available online [23].[1]

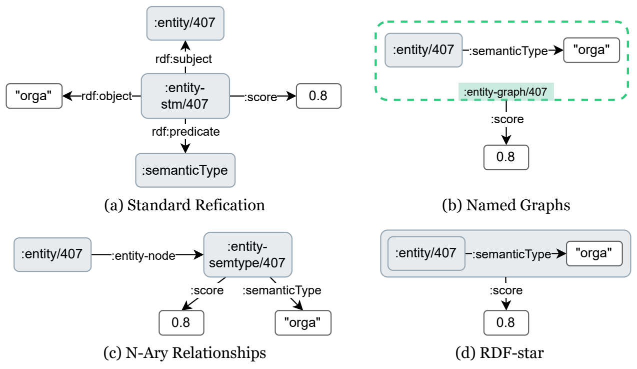

Figure 2: Structure of SemMedDB graph in four metadata representations: (a) Standard Reification, (b) Named Graphs, (c) N-Ary Relationships and (d) RDF-star.

3.1 Representation Models

We first introduce different models that are used for metadata representation. Fig. 2 illustrate the descriptions showing an excerpt of the SemMedDB dataset (used in the evaluations, see Section 3.2), in four representations [10, 19, 30, 33]. This excerpt consists of annotating the semantic type of an entity with a score.

Standard Reification [30] explicitly declares a resource to denote an `rdf:Statement`. This statement has `rdf:subject`, `rdf:predicate`, and `rdf:object` attached to it and can be further annotated with additional statements. The resource is typically a blank node, but an IRI can be used. In Fig. 2a, the resource `:entity-stm/407` is an `rdf:Statement` with four associated triples, where the objects of the triples are the actual values of the triple (i.e. `:entity/407` for the subject, `:semanticType` for the predicate, and `"orga"` for the object). The property `:score` is used with its own value as object.

Named Graphs [10] are a SPARQL 1.1 feature that allows the assignment of an IRI to one or several triples as a graph identifier. Hence, graph IRIs allow the unique identification of triples. These IRIs can be used as subjects to add additional statements. In Fig. 2b, the triple indicating the semantic type of an entity is assigned the named graph `:entity-graph/407`. The graph IRI is subsequently used as subject in a triple that annotates the confidence score of the information within the graph.

N-Ary Relationships [33] converts a relationship into an instance that describes the relation, which can have attached both the main object and additional statements. This representation is widely used in ontology engineering as an ontology design pattern [16]. In Fig. 2c, the entity `:entity/407` points to an intermediate node (`:entity-semtype/ 407`) which holds the triples for both the assignment of the semantic type and the score.

RDF-star [18, 19] extends RDF to introduce a new syntax for compact triple reification. It introduces the notion of triple recursiveness with `Quoted Triples`, which can be used as subjects and/or objects of other triples. This is the only approach that extends the standard

[1]https://github.com/oeg-upm/kg-reconstruction-eval

Table 1: Number of triples of the SemMedDB graph in the selected representations for each scale.

	1K	10K	100K	1M
Standard Reification	25,000	249,997	2,499,966	24,999,607
Named Graphs	10,000	99,994	999,932	9,999,190
N-Ary Relationships	15,000	149,997	1,499,966	14,999,595
RDF-star	8,485	78,655	710,588	6,503,388

RDF features. This representation is currently being incorporated into the RDF 1.2 specification [19], which is currently being developed under the RDF-star W3C Working Group.[2] In Figure 2d we observe the example as an RDF-star graph, and it is represented in RDF as `«:entity/407 :semanticType "orga"» :score 0.8`.

3.2 Dataset

We use the Semantic MEDLINE Database (SemMedDB) [28] in the experimental evaluation, which is used in previous evaluations of metadata representation [32, 34]. This database consists of a repository with biomedical entities and relationships (subject-predicate-object) extracted from biomedical texts, and is available as a relational database and CSV files.[3] It is licensed under the UMLS - Metathesaurus License Agreement,[4] which may be accessed with an account with the UMLS license.[5]

We use the CSV files for (i) *entity* predictions (from ENTITY.csv), and (ii) *predication* predictions (from PREDICATION.csv and PREDICATION_AUX.csv). *Subjects* and *objects* (from *predications*), and *entities* are assigned a *semantic type* with a confidence score. These *semantic types* categorize the extracted concept in the biomedical domain.[6] In addition, the extraction of *subjects* and *objects* is assigned a timestamp. Thus, the score and timestamp represent metadata for other statements. We model this tabular dataset as five annotated statements: Three assign *semantic types* to *subjects*, *objects*, and *entities* with a confidence score; and two provide the timestamp for the extraction of *subjects* and *objects* from text.

To test the scalability of the evaluated approaches, we subset this dataset into four sizes taking as input from the aforementioned CSV files with (i) 1K rows, (ii) 10K rows, (iii) 100K rows and (iv) 1M rows. The number of triples produced for each representation version on each scale is shown in Table 1.

3.3 Mappings and Queries

The following resources are used: (i) a set of declarative mappings that are used for constructing the knowledge graph from the SemMedDB tabular files in the four selected representations; and (ii) a set of SPARQL queries that are used for re-constructing the graph within different triplestores.

[2]https://www.w3.org/groups/wg/rdf-star/
[3]https://lhncbc.nlm.nih.gov/ii/tools/SemRep_SemMedDB_SKR/SemMedDB_download.html
[4]https://www.nlm.nih.gov/research/umls/knowledge_sources/metathesaurus/release/license_agreement.html
[5]An account with the UMLS license can be requested at https://www.nlm.nih.gov/databases/umls.html.
[6]https://www.nlm.nih.gov/research/umls/new_users/online_learning/SEM_003.html

Table 2: Characteristics of mappings in RML and SPARQL-Anything. #TM stands for number of Triples Map, #POM for Predicate Object Map, and #TP for Triple Patterns. The shown operators appear usually in the `WHERE` clause; the ones marked with ^{c} appear in the `CONSTRUCT` clause.

	RML		SPARQL-Anything	
	#TM	**#POM**	**#TP^c**	**Additional operators**
Standard Reification	9	20	25	UNION, BIND
Named Graphs	10	10	10	UNION, BIND, GRAPH^c
N-Ary Relationships	13	15	15	UNION, BIND
RDF-star	10	10	10	UNION, BIND

We use two sets of mappings, one set written in the RML mapping language [13, 27], and the other in SPARQL-Anything [7]. These languages allow describing transformation of heterogeneous data sources into RDF following the schema provided by an ontology or vocabulary, and are processed by different engines (see Section 3.4). Each set of mappings is comprised of four mappings, to construct the KGs in the four representations selected (i.e. Standard Reification, N-Ary Relationships, Named Graphs and RDF-star).

RML [27] extends the R2RML Recommendation [11] to describe more data sources besides Relational Databases. RML rules are grouped within *Triples Maps*, which contain one *Logical Source*, one *Subject Map* and zero to multiple *Predicate Object Maps*. *Logical Sources* describe the input data to be transformed. *Subject Maps* indicates how the subjects of the triples are created, while *Predicate Object Maps* specify how the predicates and objects of the triples are created. Listing 1 shows an example of a mapping that generates in RDF-star the annotation of the semantic types of *entities* with a score. This mapping uses the RML-star module [12, 27] to generate RDF-star graphs, which allows RML to quote *Triples Map* with the `rml:quotedTriplesMap` property. For these mappings, we show the number of sets of rules (i.e. *Triples Map*) and *Predicate Object Maps* specified in Table 2.

```
<#Entity> a rml:TriplesMap ;
 rml:logicalSource [ rml:source "ENTITY.csv" ];
 rml:subjectMap [ rml:template ":{ENTITY_ID}" ];
 rml:predicateObjectMap [
  rml:predicate :semanticType;
  rml:objectMap [ rml:reference "SEMTYPE"  ] ] .
<#EntityScore> a rml:AssertedTriplesMap ;
 rml:logicalSource [ rml:source "ENTITY.csv" ];
 rml:subjectMap [ rml:quotedTriplesMap <#Entity>;];
 rml:predicateObjectMap [
  rml:predicate :score ;
  rml:objectMap [ rml:reference "SCORE" ] ] .
```

Listing 1: RML-star mapping snippet to create the RDF-star graph for *entity* from data in ENTITY.csv.

SPARQL-Anything [7] heavily relies on the SPARQL syntax, overriding the `SERVICE` operator while leveraging the rest of its features. This operator is then always present in the queries and hence is not shown in the table. The graphs are built with `CONSTRUCT` clauses.

Table 3: Characteristics of the SPARQL queries used for the evaluation to transform the graph from a *source* representation to a *target* representation. #TP stands for the number of triple patterns. Fields marked with ^{c} appear in the CONSTRUCT clause, while ^{w} indicates the WHERE clause.

	Source	Target	#TP	#TP^w	#TP^c	UNION^w	BIND^w	FILTER^w	VALUES^w	GRAPH
Q1	**N-AryRel.**	**Std. Reif.**	16	6	10	✓			✓	
Q2		**RDF-Star**	10	6	4	✓			✓	
Q3		**Named Graphs**	10	6	4	✓			✓	✓^{c}
Q4	**Std. Reif.**	**RDF-star**	12	8	4	✓			✓	
Q5		**N-Ary Rel.**	35	20	15	✓		✓		
Q6		**Named Graphs**	12	8	4	✓			✓	✓^{c}
Q7	**RDF-star**	**Std. Reif.**	30	5	25	✓	✓	✓		
Q8		**N-Ary Rel.**	20	5	15	✓	✓	✓		
Q9		**Named Graphs**	15	5	10	✓	✓	✓		✓^{c}
Q10	**Named Graphs**	**Std. Reif.**	14	4	10	✓			✓	✓^{w}
Q11		**RDF-Star**	8	4	4	✓			✓	✓^{w}
Q12		**N-Ary Rel.**	19	4	15	✓			✓	✓^{w}

Listing 2 shows a mapping example that generates in RDF-star the annotation of the semantic types of *entities* with a score. For each mapping, we show the number of triple patterns in this clause and additional SPARQL clauses (Table 2). All mappings contain the same number of triple patterns in the WHERE clause.

```
CONSTRUCT {
  ?entity_id_iri :semanticType ?entity_semtype .
  << ?entity_id_iri :semanticType ?entity_semtype >>
                 :score ?entity_score . }
WHERE {
  {  SERVICE <x-sparql-anything:location=./data/entity.csv>
   { [] xyz:ENTITY_ID ?entity_id;
     xyz:SEMTYPE ?entity_semtype;
     xyz:SCORE ?entity_score;
    BIND(uri(concat(str("http://semmeddb.com/entity/"),
         encode_for_uri(?entity_id))) as ?entity_id_iri) }}}
```

Listing 2: SPARQL-Anything mapping snippet to create the RDF-star graph for *entity* from data in ENTITY.csv.

```
CONSTRUCT {
  ?entity_id_iri :semanticType ?entity_semtype .
  << ?entity_id_iri :semanticType ?entity_semtype >>
                 :score ?entity_score . }
WHERE {
  GRAPH ?entity_graph_iri {
    ?entity_id_iri :semanticType ?entity_semtype }
  ?entity_graph_iri :score ?entity_score . }
```

Listing 3: SPARQL query snippet to create the RDF-star graph from the Named Graphs representation for *entity*.

SPARQL queries use the CONSTRUCT clause to re-construct a given graph with one of the representations into the other three representations. We use twelve queries to transform all pairs of representations. We show in Table 3 for each query the number of triple patterns in the WHERE and CONSTRUCT clauses, and the additional operators used. Except for GRAPH, the rest of operators only appear within the WHERE clause. An example of a query is shown in Listing 3, which generates RDF-star from Named Graphs.

3.4 Engines

We choose two mapping engines and three triplestores to be representative in our evaluation, while focusing on open-source tools. The selected mapping engines, Morph-KGC (v2.5.0) [4] and SPARQL-Anything (v0.8.1) [7] are the engines with a stable version capable of producing RDF-star at the time of writing this paper. The SDM-RDFizer [24] is developing this feature, but no stable version has been released yet. Regarding triplestores, we use the free version of GraphDB (v10.2.1), and the open-source Jena Fuseki (v4.8.0) and Oxigraph (v0.3.16). While GraphDB and Jena Fuseki store the graph in physical memory, Oxigraph performs the queries in memory. All engines perform duplicate removal in the results by default, except for Oxigraph. For this triplestore, we add and measure a second step of duplicate removal with BASH commands.

We did attempts to include Virtuoso in the evaluation; however, several issues were raised. The number of queries this triplestore can perform is very limited in this evaluation (only 4 from 12): This triplestore does not yet implement RDF-star and cannot produce named graphs with CONSTRUCT. In addition, Virtuoso limits the number of triples that can be produced with this clause to 1M.[7] By removing this limit, an error appears that impedes file writing with large results, which has been unsolved for years.[8]

3.5 Experimental setup and Metrics

We perform the evaluation in two steps. First, the KG construction (KGC) system evaluation is run, taking as input the SemMedDB dataset in CSV format and the mappings, and producing the corresponding RDF datasets in the four selected representations. Then,

[7]https://lig-membres.imag.fr/rousset/publis/tess.pdf
[8]https://github.com/openlink/virtuoso-opensource/issues/11

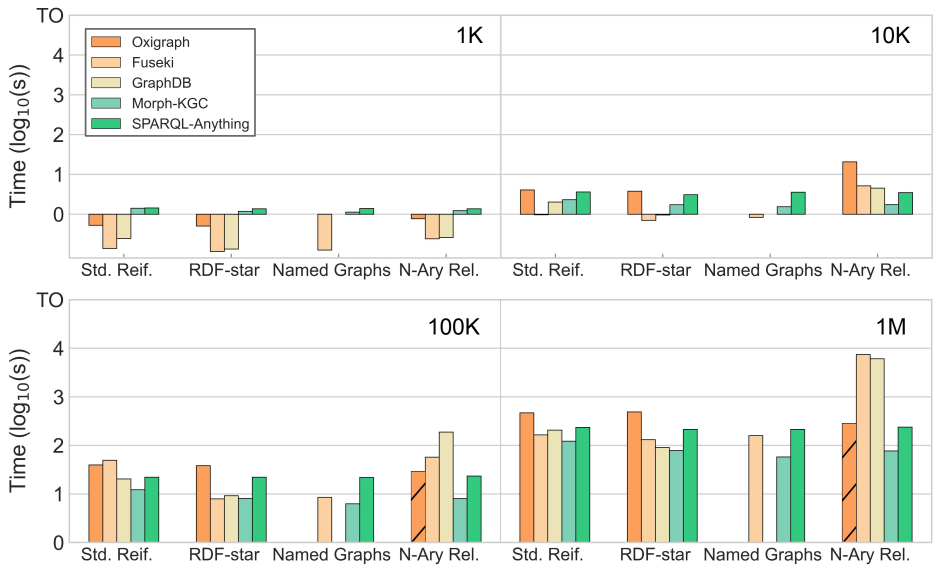

Figure 3: Execution time for KG construction engines with declarative mappings, and triplestores with SPARQL queries. The results of the triplestores are grouped by the *target* representation representing their geometric mean. Bars with diagonal pattern do not include some target representation times, as they report out-of-memory errors (details in Fig. 4).

the triplestore evaluation is carried out, taking as input the produced graphs in the KGC system evaluation, and producing RDF datasets in another representation. We perform a validation step afterwards to verify that the output graphs do not lack information.

We measure the *materialization time* to construct the RDF graph from the input sources in KGC systems. For triplestores, we measure *query execution time* as the total time from query execution until the complete answer is generated. We also report the geometric mean of all queries that generate graphs in the same metadata representation, similar to that previously used by [31, 38]. The *geometric mean* reports the central tendency of all execution times for a set of queries, reducing the effect of outliers. With this metric, we provide a general measurement of the performance of each triplestore when generating graphs in each representation. This allows an easier comparison with the KGC systems. We run every experiment 5 times with a timeout of 24h, measure the time and calculate the median. We run all experiments over an Ubuntu 20.04 server with 32 cores Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz 102400 Mb RAM RDIMM, 3200 MT/s and 100 Gb HD SSD 6 Gb/s.

4 RESULTS

The evaluation results are presented in this section. Fig. 3 reports the comparison between (KGC) systems using declarative mappings, and triplestores with SPARQL queries. Fig. 4 reports a fine-grained comparison of each triplestore performance on the proposed queries performing translations between each pair of representations.

Focusing on the comparison between triplestores and KGC systems (Fig. 3), we generally observe that for small data sizes, triplestores obtain better results, while KGC systems scale better as data size increases. However, except for producing N-Ary Relationships, Fuseki and GraphDB are competitive w.r.t. SPARQL-Anything or Morph-KGC. These triplestores obtain better results for the scales 1K and 10K, and similar ones for 100K and 1M. Additionally, despite the variety in mapping characteristics, neither Morph-KGC nor SPARQL-Anything seem to be highly affected by the different representations, as opposed to the triplestores. The differences are not remarkable, but in general Morph-KGC generates the fastest Named Graphs, in contrast with SPARQL-Anything, which performs equally well with Named Graphs and RDF-star. For triplestores, producing Standard Reification and specially N-Ary Relationships is more costly than the other representations. The apparent good performance of Oxigraph in this case is due to out-of-memory errors in scales 100K and 1M. In addition, SPARQL 1.1 does not allow `GRAPH` clause within the `CONSTRUCT` operator. Hence, only Fuseki, which implements this extension natively, can generate the Named Graphs datasets.

Comparing the behavior of engines that perform the same task, GraphDB overcomes Fuseki and Oxigraph for SPARQL queries, while Morph-KGC reports better results than SPARQL-Anything for KG construction from heterogeneous data sources, as was reported in previous works [5]. For small data sizes, GraphDB and Fuseki perform similar, while Oxigraph reports higher query execution time and out-of-memory errors. This is because Oxigraph loads the complete RDF graph in memory and the physical data structures from Fuseki and GraphDB speed up query execution time. In larger datasets, GraphDB generally scales better than Fuseki, which for example, reports a timeout for generating N-Ary Rel. in scale 1M.

Answering the research questions, we can say that triplestores perform faster for small sizes (**RQ1**), while KGC systems exhibit a more robust behavior with increasing data size (**RQ2**). In addition, triplestores are more influenced by changes in representations, whereas KGC systems are mostly unaffected (**RQ3**).

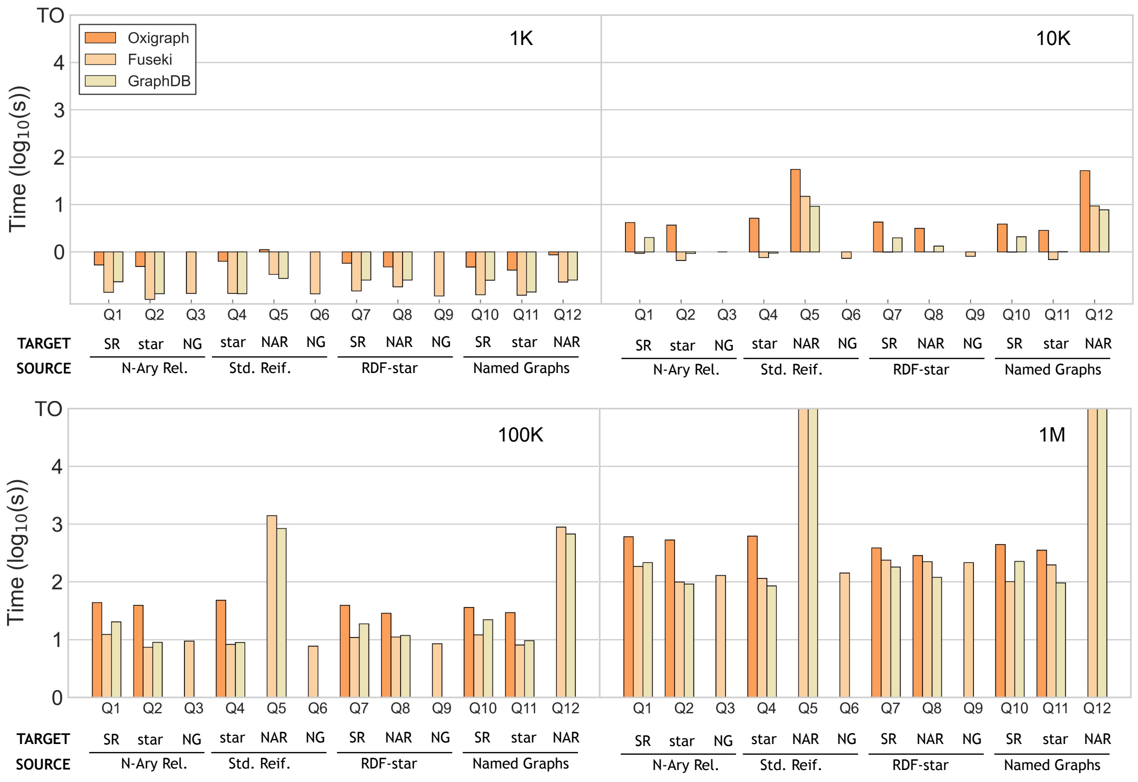

Figure 4: Execution time of triplestores with the SPARQL queries that perform translations for pairs of representations.

Looking into the particular differences of the pairs of translation performed with triplestores, we can observe how the different representations affect the performance of the graph re-construction with `CONSTRUCT` queries. Fig. 4 present the results grouping the queries in the legend by the *source* representation (i.e. from which representation the dataset is transformed). The behavior reported when producing N-Ary Relationships stands out. It is more costly to produce than the other representation, except when RDF-star is the *source* representation (Q8). Queries Q5 and Q12 report an out-of-memory error in Oxigraph for scales 100K and 1M, while reaching a timeout in Fuseki and GraphDB in scale 1M. In contrast, Q8 requires a more complex `WHERE` clause, that avoids introducing a join that is needed in Q5 and Q12.

Table 4 shows a summary of the results from the triplestores grouped by *source* and *target* representation. In general, Named Graphs are the fastest representation to construct. N-Ary Relationships perform the fastest as the *source* representation, but require performing joins in the `CONSTRUCT` that make them the least suitable as a target representation. Standard Reification supposes the least suitable *source* representation, probably because of its larger size. Meanwhile, RDF-star performs consistently well acting as both *source* and *target* representation. Therefore, we can conclude for **RQ3** that the combination of representations highly influences the behavior of triplestores, both in the `WHERE` and `CONSTRUCT` clauses.

Table 4: Geometric mean of the query result times (s) from all triplestores grouped by *source* representation and *target* representation. The lowest times are highlighted in bold, while the highest are <u>underlined</u>.

	Representation	1K	10K	100K	1M
Source	Std. Reif.	<u>0.283</u>	<u>4.126</u>	<u>56.612</u>	<u>1085.228</u>
	RDF-Star	0.248	1.612	**15.951**	223.626
	Named Graphs	0.260	3.395	43.756	773.737
	N-Ary Rel.	**0.204**	**1.505**	16.040	**205.407**
Target	Std. Reif.	0.261	2.005	34.212	252.490
	RDF-Star	0.199	1.366	14.066	180.384
	Named Graphs	**0.126**	**0.836**	**8.524**	**158.678**
	N-Ary Rel.	<u>0.365</u>	<u>7.896</u>	<u>67.886</u>	<u>2351.176</u>

5 DISCUSSION

In this paper, we study how two graph re-construction approaches behave with metadata representation models, and the aspects that influence their performance ensuring no information loss. We evaluate four representations with (i) KG construction systems, that construct the KG from heterogeneous data with declarative mappings; and (ii) using `CONSTRUCT` queries from KG stored in triplestores.

Our evaluation shows that KGC systems are more robust for increasing data size and interchange of metadata representation models. These systems are able to produce the four representations without major performance changes among them. Performing the re-construction in triplestores is suitable for small data sizes, but it is more dependant on the target representation and query optimizations to offer competitive performance. In addition, this approach presents limitations as to which representations can be produced: not all existing triplestores process RDF-star, and only Fuseki can produce Named Graphs with CONSTRUCT. Thus, we can affirm that performing the re-construction with mappings is in general more reliable, as it is less affected by data size and target representation.

Regarding the representations, RDF-star and Named Graphs are a safe option to adopt in terms of system performance for both re-construction approaches. However, they are the ones that are currently less supported by both KGC systems (only for RDF-star) and triplestores. The other two representations present complete support by a broader set of engines, not included in this study too (e.g., RMLMapper, Virtuoso), but they present worse performance. Independently on the representation and re-construction approach, it is possible to re-construct a graph with no information loss.

The setup of the evaluation shows us the importance of optimizing SPARQL queries, which gains relevance as the triples to construct increase in number. For instance, the introduction of UNION clauses was needed to avoid costly cartesian products that prevented queries from finishing before the established timeout, or before reaching an out-of-memory error. This is particularly important when RDF-star is the *source* representation, as otherwise queries would often incur in erroneous output graphs. While this is well known by proficient SPARQL practitioners, it does not come as easily for non-expert users. The performance of SPARQL-Anything, relying almost entirely on SPARQL and Jena processing, is also affected by these different manners of writing the mapping. In contrast, RML mappings are unaffected in this aspect, since how the user writes the mapping does not influence the performance of the compliant systems. SPARQL possess a flexibility and rich expressiveness that languages such as RML lack yet. Nevertheless, it poses the risk for non-expert users to hamper the result retrieval, incurring in suboptimal and erroneous queries. This opens up a challenge for improving the query processing, or even rewriting the queries automatically to be optimized, so as to reduce this accessibility gap of SPARQL for non-expert users.

In addition, this evaluation brings to light that the behavior of the CONSTRUCT clause is not as studied as other SPARQL operators. SPARQL benchmarks often overlook this clause, as it is considered as an extension of SELECT [38]. Hence, it is assumed that their performance is comparable and not affected by the change of clause. However, we encountered SELECT queries that returned results in miliseconds, while taking several minutes with CONSTRUCT. In addition, all triplestores struggle when the CONSTRUCT clause includes several joins. This fact affects not only the metadata representations studied in this paper, but also all potential transformations for evolving knowledge graphs. Regarding KGC systems, the main challenge still consists of their adoption, as the learning curve for mapping languages is still steep despite efforts to lower it [26].

6 RELATED WORK

There are diverse studies on the impact of different metadata representations during query execution [15, 20, 21, 32, 34]. Nguyen et al. [32] propose Singleton Properties and evaluate query performance with their proposal against other representations. Hernández et al. [20] test Wikidata represented with different representations (i.e., Standard Reification, N-Ary Relationships, Singleton Properties and Named Graphs) over multiple triplestores, and in a subsequent study [21] they extend the evaluation with a relational database (PostgreSQL) and a graph database (Neo4J). Frey et al. [15] evaluates DBpedia with different representations, including their proposal, Companion Property, and Blazegraph's Reification Done Right[9] which is currently known as RDF-star) also with different triplestores. More recently, the REF-benchmark [34] evaluates different reification approaches with the inclusion of RDF-star, providing a version of the Biomedical Knowledge Repository (BKR) dataset [36] and three series of queries for each representation that can be applied to different triplestores. To the best of our knowledge, these studies focus on comparing different representations regarding query performance, but none provides a systematic evaluation of the impact of transformation between them.

The analysis of SPARQL triplestores has been widely studied in the literature [1, 37]. The Berlin SPARQL Benchmark (BSBM) [8], focused on the e-commerce domain, is one of the most well-known benchmarks used to evaluate the performance and scalability of triplestores and virtual knowledge graph construction systems. This benchmark provides a set of 12 SPARQL queries, together with a generator that scales-up the input data source for both RDF and SQL. Only one of the 12 proposed queries evaluates the performance of the CONSTRUCT operator. Under the assumption that triplestores apply the same procedures to execute and optimize CONSTRUCT and SELECT queries, other SPARQL benchmarks (e.g. WatDiv [2], SP2Bench [38], DBPedia Benchmark [31], WDBench [3]) do not include any SPARQL CONSTRUCT queries. Our results demonstrate the need to apply new optimization techniques in the construction of knowledge graphs using the CONSTRUCT operator to make their execution more scalable.

Knowledge graph construction engines have also been evaluated w.r.t. performance and scalability. BSBM [8] or NPD [29] are examples of benchmarks used to perform these evaluations, although GTFS-Madrid-Bench [9] has gained exponential popularity and adoption since its release as a dedicated benchmark for KG construction [39]. It provides a set of 18 queries, together with a synthetic data generator that scales-up and distributes the input tabular data source in multiple formats. In addition to evaluating the capabilities of virtual KG construction systems [9], the benchmark is also used to test the performance and scalability of materialized knowledge graph systems [6]. All these proposals have helped to improve these engines, yet none of them evaluate KG construction with metadata representation. Additionally, we are not aware of any study that compares KG construction from the original data sources with mapping languages against its re-construction using already existing graphs.

[9]https://github.com/blazegraph/database/wiki/Reification_Done_Right

7 CONCLUSIONS

This paper aims to study the re-construction of knowledge graphs using metadata representation models with two different approaches. We perform an empirical analysis using two KG construction systems and three triplestores, performing the re-construction with four data sizes in four different representations: RDF-star, Standard Reification, N-Ary Relationships and Named Graphs. Our results show that KG construction engines in general are more scalable, almost independently on the representation. Triplestores perform best for small data sizes. However, their performance is more dependent on the source representation (in the `WHERE` clause) and the target representation (in the `CONSTRUCT` clause). Both RDF-star and Named Graphs demonstrate that they are suitable as target representation. They present a good performance in terms of time, but have still limited technical support by both re-construction approaches. Meanwhile, Standard Reification and N-Ary Relationships, with poorer performance, are supported by all known re-construction engines. Independently on the re-construction approach and representation, it is possible to interchange data among representations without information loss.

With this study, we aim to provide a better understanding of re-constructing knowledge graphs with metadata representations, along with the relevant aspects to consider. We discuss the importance of broadening research on the `CONSTRUCT` clause regarding query optimization and implementation in different triplestores.

In the future, we plan on widening the study of KG modifications and their impact in re-construction. We limit the presented work to changes from one metadata representation to another representation, motivated by different current use cases. Moreover, it is unsure if there is a differential impact coming from the number of reified triples in the graph, or from the nature of their value. Additional changes usually present in KG evolution force their re-construction, with potential additional nuances not discovered so far, which opens the door to future analysis.

ACKNOWLEDGMENTS

This work is partially supported by the project *Knowledge Spaces* (Grant PID2020-118274RB-I00 funded by MCIN/AEI/10.13039/50110 0011033), and partially supported by the Galician Ministry of Education, University and Professional Training and the European Regional Development Fund (ERDF/FEDER program) through grants ED431C2018/29 and ED431G2019/04).

REFERENCES

[1] Waqas Ali, Muhammad Saleem, Bin Yao, Aidan Hogan, and Axel-Cyrille Ngonga Ngomo. 2022. A survey of RDF stores & SPARQL engines for querying knowledge graphs. *The VLDB Journal* (2022), 1–26.

[2] Güneş Aluç, Olaf Hartig, M Tamer Özsu, and Khuzaima Daudjee. 2014. Diversified stress testing of RDF data management systems. In *The Semantic Web–ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I 13*. Springer, 197–212.

[3] Renzo Angles, Carlos Buil Aranda, Aidan Hogan, Carlos Rojas, and Domagoj Vrgoč. 2022. WDBench: A Wikidata Graph Query Benchmark. In *The Semantic Web–ISWC 2022: 21st International Semantic Web Conference, Virtual Event, October 23–27, 2022, Proceedings*. Springer, 714–731.

[4] Julián Arenas-Guerrero, David Chaves-Fraga, Jhon Toledo, María S Pérez, and Oscar Corcho. 2022. Morph-kgc: Scalable knowledge graph materialization with mapping partitions. *Semantic Web* (2022).

[5] Julián Arenas-Guerrero, Ana Iglesias-Molina, David Chaves-Fraga, Daniel Garijo, Oscar Corcho, and Anastasia Dimou. 2023. Declarative generation of RDF-star graphs from heterogeneous data. *Semantic Web (Under Review)* 24 (2023).

[6] Julián Arenas-Guerrero, Mario Scrocca, Ana Iglesias Molina, Jhon Toledo, Luis Pozo-Gilo, Daniel Dona, Oscar Corcho, and David Chaves-Fraga. 2021. Knowledge graph construction with R2RML and RML: an ETL system-based overview. In *CEUR workshop proceedings.*, Vol. 2873. CEUR Workshop Proceedings.

[7] Luigi Asprino, Enrico Daga, Aldo Gangemi, and Paul Mulholland. 2023. Knowledge Graph Construction with a Façade: A Unified Method to Access Heterogeneous Data Sources on the Web. *ACM Transactions on Internet Technology* 23, 1, Article 6 (2023), 31 pages. https://doi.org/10.1145/3555312

[8] Christian Bizer and Andreas Schultz. 2009. The Berlin SPARQL Benchmark. *International Journal on Semantic Web and Information Systems (IJSWIS)* 5, 2 (2009), 1–24.

[9] David Chaves-Fraga, Freddy Priyatna, Andrea Cimmino, Jhon Toledo, Edna Ruckhaus, and Oscar Corcho. 2020. GTFS-Madrid-Bench: A benchmark for virtual knowledge graph access in the transport domain. *Journal of Web Semantics* 65 (2020), 100596.

[10] Richard Cyganiak, David Wood, Markus Lanthaler, Graham Klyne, Jeremy J. Carroll, and Brian McBride. 2014. *RDF 1.1 Concepts and Abstract Syntax.* W3C Recommendation. World Wide Web Consortium. http://www.w3.org/TR/rdf11-concepts/

[11] Souripriya Das, Seema Sundara, and Richard Cyganiak. 2012. *R2RML: RDB to RDF Mapping Language.* W3C Recommendation. World Wide Web Consortium. http://www.w3.org/TR/r2rml/

[12] Thomas Delva, Julián Arenas-Guerrero, Ana Iglesias-Molina, Oscar Corcho, David Chaves-Fraga, and Anastasia Dimou. 2021. RML-star: A Declarative Mapping Language for RDF-star Generation. In *International Semantic Web Conference, ISWC, P&D*, Vol. 2980. CEUR Workshop Proceedings. http://ceur-ws.org/Vol-2980/paper374.pdf

[13] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. 2014. 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. http://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf

[14] Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandečić. 2014. Introducing Wikidata to the linked data web. In *International Semantic Web Conference.* Springer, 50–65.

[15] Johannes Frey, Kay Müller, Sebastian Hellmann, Erhard Rahm, and Maria-Esther Vidal. 2019. Evaluation of metadata representations in RDF stores. *Semantic Web* 10, 2 (2019), 205–229.

[16] Aldo Gangemi and Valentina Presutti. 2013. A multi-dimensional comparison of ontology design patterns for representing n-ary relations. In *International Conference on Current Trends in Theory and Practice of Computer Science.* Springer, 86–105.

[17] Paul Groth, Andrew Gibson, and Jan Velterop. 2010. The anatomy of a nanopublication. *Information services & use* 30, 1-2 (2010), 51–56.

[18] Olaf Hartig. 2017. 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. http://ceur-ws.org/Vol-1912/paper12.pdf

[19] Olaf Hartig, Pierre-Antoine Champin, and Gregg Kellog. 2023. *RDF 1.2 Concepts and Abstract Syntax.* W3C Working Draft. World Wide Web Consortium. https://www.w3.org/TR/rdf12-concepts/

[20] Daniel Hernández, Aidan Hogan, and Markus Krötzsch. 2015. Reifying RDF: What works well with wikidata? *11th International Workshop on Scalable Semantic Web Knowledge Base Systems co-located with 14th International Semantic Web Conference* 1457, 32–47.

[21] Daniel Hernández, Aidan Hogan, Cristian Riveros, Carlos Rojas, and Enzo Zerega. 2016. Querying wikidata: Comparing sparql, relational and graph databases. In *International Semantic Web Conference.* Springer, 88–103.

[22] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. 2021. Knowledge graphs. *ACM Computing Surveys (CSUR)* 54, 4 (2021), 1–37.

[23] Ana Iglesias, Jhon Toledo, and David Chaves. 2023. *oeg-upm/kg-reconstruction-eval: v1.0.0.* https://doi.org/10.5281/zenodo.8309880

[24] Enrique Iglesias, Samaneh Jozashoori, David Chaves-Fraga, Diego Collarana, and Maria-Esther Vidal. 2020. 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.* 3039–3046.

[25] Ana Iglesias-Molina, Kian Ahrabian, Filip Ilievski, Jay Pujara, and Oscar Corcho. 2023. Comparison of Knowledge Graph Representations for Consumer Scenarios. In *Proceedings of the 22nd International Semantic Web Conference, Athens, Greece, November 6–10, 2023.* Springer.

[26] Ana Iglesias-Molina, David Chaves-Fraga, Ioannis Dasoulas, and Anastasia Dimou. 2023. Human-Friendly RDF Graph Construction: Which One Do You Chose?. In *International Conference on Web Engineering.* Springer, 262–277.

[27] Ana Iglesias-Molina, Dylan Van Assche, Julián Arenas-Guerrero, Ben De Meester, Christophe Debruyne, Samaneh Jozashoori, Pano Maria, Franck Michel, David

Chaves-Fraga, and Anastasia Dimou. 2023. The RML Ontology: A Community-Driven Modular Redesign After a Decade of Experience in Mapping Heterogeneous Data to RDF. In *Proceedings of the 22nd International Semantic Web Conference, October 6-10, Athens, Greece*. Springer.

[28] Halil Kilicoglu, Dongwook Shin, Marcelo Fiszman, Graciela Rosemblat, and Thomas C. Rindflesch. 2012. SemMedDB: a PubMed-scale repository of biomedical semantic predications. *Bioinformatics* 28, 23 (10 2012), 3158–3160.

[29] Davide Lanti, Martin Ignacio Rezk, Guohui Xiao, and Diego Calvanese. 2015. The NPD benchmark: Reality check for OBDA systems. In *Advances in database technology-EDBT 2015: 18th International Conference on Extending Database Technology, Brussels, Belgium, March 23-27, 2015, proceedings*. University of Konstanz, University Library, 617–628.

[30] Ora Lassila and Ralph R. Swick. 1999. *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

[31] Mohamed Morsey, Jens Lehmann, Sören Auer, and Axel-Cyrille Ngonga Ngomo. 2011. DBpedia SPARQL benchmark–performance assessment with real queries on real data. In *The Semantic Web–ISWC 2011: 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I 10*. Springer, 454–469.

[32] Vinh Nguyen, Olivier Bodenreider, and Amit Sheth. 2014. Don't like RDF Reification? Making Statements about Statements Using Singleton Property. In *Proceedings of the 23rd International Conference on World Wide Web*. Association for Computing Machinery, 759—-770. https://doi.org/10.1145/2566486.2567973

[33] Natasha Noy and Alan Rector. 2006. *Defining N-ary Relations on the Semantic Web: Use With Individuals*. Technical Report. W3C. https://www.w3.org/TR/swbp-n-aryRelations/

[34] Fabrizio Orlandi, Damien Graux, and Declan O'Sullivan. 2021. Benchmarking RDF metadata representations: Reification, singleton property and RDF. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*. IEEE, 233–240.

[35] Núria Queralt-Rosinach, Tobias Kuhn, Christine Chichester, Michel Dumontier, Ferran Sanz, and Laura I Furlong. 2016. Publishing DisGeNET as nanopublications. *Semantic Web* 7, 5 (2016), 519–528.

[36] Satya S Sahoo, Olivier Bodenreider, Pascal Hitzler, Amit Sheth, and Krishnaprasad Thirunarayan. 2010. Provenance Context Entity (PaCE): Scalable provenance tracking for scientific RDF data. In *International Conference on Scientific and Statistical Database Management*. Springer, 461–470.

[37] Muhammad Saleem, Gábor Szárnyas, Felix Conrads, Syed Ahmad Chan Bukhari, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. 2019. How Representative is a SPARQL benchmark? An Analysis of RDF Triplestore Benchmarks. In *The World Wide Web Conference*. 1623–1633.

[38] Michael Schmidt, Thomas Hornung, Georg Lausen, and Christoph Pinkel. 2009. SPˆ 2Bench: a SPARQL performance benchmark. In *2009 IEEE 25th International Conference on Data Engineering*. IEEE, 222–233.

[39] Dylan Van Assche, Thomas Delva, Gerald Haesendonck, Pieter Heyvaert, Ben De Meester, and Anastasia Dimou. 2023. Declarative RDF graph generation from heterogeneous (semi-)structured data: A systematic literature review. *Journal of Web Semantics* 75 (2023), 100753. https://doi.org/10.1016/j.websem.2022.100753