UNIVERSITEIT GENT

Faculty of Sciences

Stream join processing in RDF mapping engines

by

Sitt Min Oo
Student number: 01503244

Supervisors: Prof. Dr. Ruben Verborgh and Dr. Anastasia Dimou
Counsellor: Gerald Haesendonck

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Computer Science

Academic Year 2020–2021

Preface

Acknowledgement

Sitt Min Oo, December 2020

Stream join processing in RDF mapping engines

by

Sitt Min Oo

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Computer Science

Academic year 2020–2021

Supervisor: Prof. Dr. Ir. Bjorn De Sutter
Counsellor: Dr. Tim Besard

Faculty of Sciences
University Ghent

Abstract

Here comes abstract.

Keywords

RDF, RMLStreamer, RML, Adaptive windows, Stream joins.

Contents

Chapter 1

Introduction

A large volume of data is generated daily on the Web in a variety of domains. These data are often structured according to an organization's specific needs or formats: Leading to a difficulty in integrating the data across the different applications. These generated data might have to be associated with archival data, also of heterogeneous formats, to provide a coherent view required by analysis tasks. Heterogeneous Web data formats, such as CSV or HTML, are not explicitly defined to enable linking entities in one document to other related entities in external documents.

Based on W3C standard, semantic data formats such as RDF triples [1], are a solution to this particular problem by enriching the data with knowledge and association across different domains, through the use of common ontologies. RDF triples also form the basic building blocks of knowledge graphs. Knowledge graphs are extensively used in social networks like Facebook[2], IoT devices[3] and especially with Google's search engine[4], it enables machines to understand the data and perform complex automated processing on the data. Considering the aforementioned scenarios, there is a need to transform these non-RDF data to RDF compliant formats on the fly while new data are being generated. Furthermore, we would also like to apply stream operators on the input before transforming, to enhance the enrichment of the data by applying operators on the incoming tuples.

There exists state-of-the-art techniques to solve the task of consolidating heterogeneous data and transforming them to an RDF compliant format. In this thesis, we will focus on one such format called TURTLE [5]. These RDF transformation engines can be categorized into two major categories based on the type of input which they consume; bounded and unbounded data input. Since we are focussing on the generation of RDF data in a streaming environment, the class of

RDF transformation engines on unbounded data will be of interest to our study.

Some engines support traditional stream operators like joins and aggregations. However, they do not consider the characteristics of the streaming sources such as velocity and time-correlations between the different input streams. This leads to a decline in the quality of the generated RDF triples. Moreover, due to the nature of the infinite, continuous and real-time changing data of the streaming environment, these operators have to be applied in the context of windows over a subset of the incoming data. Clearly, with these restrictions and characteristics of the streaming sources, we need an adaptive approach to applying these operators in windows.

Chapter 2

Semantic Web Technologies

Multiple framework exists to deal with c This section will define and describe the different notations and terms used throughput the rest of this work. We will elaborate more on RDF, RML and stream windowing for more intuition behind the approach of this work.

2.1 RDF

Resource Definition Framework [6] is a framework for representing data on the Web. It portrays the data as a directed graph with the resources as nodes in the graph and the edges as the relationship between the different resources. Figure 2.1 shows an example of an RDF triple statement describing the information "John has an apple". The triple statement consists of the subject *John*, the predicate *has* and the object *apple*.

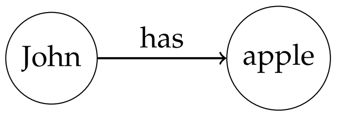

Figure 2.1: An RDF triple representing the information "John has an apple".

By composing these simple triple statements into a set of RDF triples, it yields us an RDF graph. In Figure 2.2, 4 triple statements are composed together to form a simple RDF graph describing *John* and *Mary* having the same *apple*. It might not be evident from the simple figures, about the advantages of RDF graphs. Data representation in a graph model allows machines to follow the *links* between the resources, and discover more unknown data in the linked knowledge graph. Link following is possible due to the nodes in the triples being classified as one of the 3 different term types.

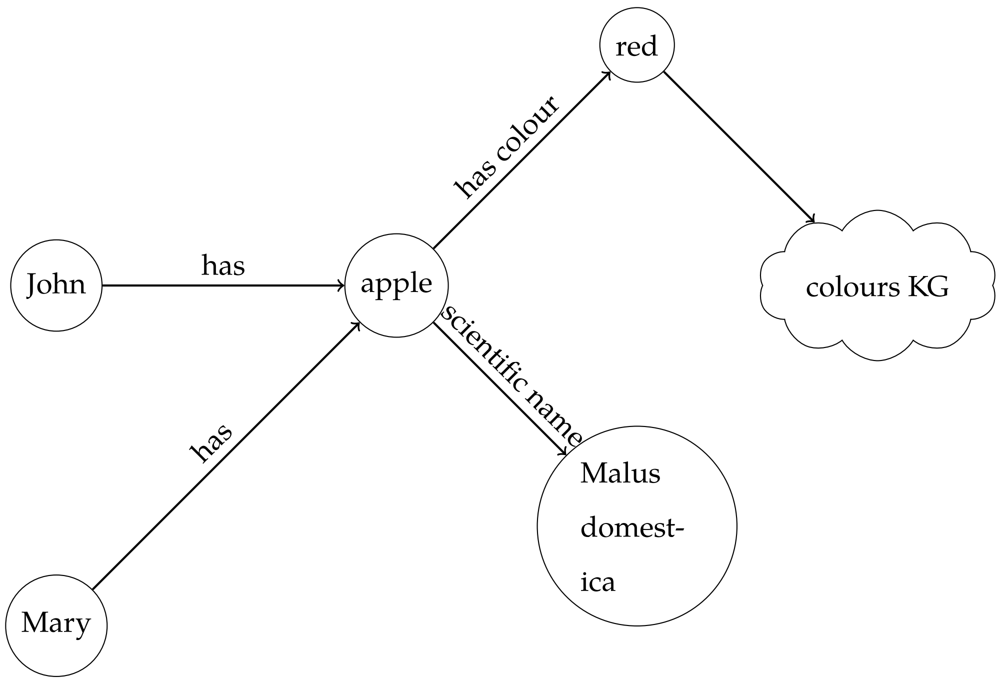

Figure 2.2: A simple RDF graph where the same "apple" is shared by both John and Mary.

2.1.1 Term types

Resources are classified into 3 different term types; IRI (Internationalized Resource Identifier), literals and blank nodes. IRI is a string identifier unique in the global scope to represent a resource. It is usually in the form of a web address, however, it can also be in other forms so long as it conforms to the syntax defined in RFC 3987 [1]. IRI can represent a relationship, a concept or an object. Therefore, it could be used in the *subject*, the *predicate* and the *object* components of an RDF triple.

Literals term type is used to represent a value such as strings, numbers, boolean and dates. To ensure that the machines know the type of the data being read, we could explicitly specify the type of the data with a datatype IRI. Moreover, the language in which the data is written, could also be explicitly stated with a language tag.

Lastly, we have the term type blank nodes. Blank nodes identify resources in the local scope (i.e. to a local file or an RDF store). Since it is used to identify resources in nodes, blank nodes are only applicable as the *subject* and the *object* components of an RDF triple.

[1]RFC 3987: https://www.ietf.org/rfc/rfc3987.txt

2.1.2 TURTLE syntax

The aforementioned term types are defined for abstract RDF syntax. For a concrete syntax to write RDF triples, we would focus on the TURTLE (Terse RDF Triple Language) [5] syntax in this work. A simple triple statement is a sequence of *subject, predicate, object* terms, ending in a '.'. To reduce the repetition of writing the same subject and predicate combination with different objects, TURTLE allows the use of ',' to separate the different objects. Additionally, one could also use ';' to separate the different predicates and objects sharing the same subject.

Listing 2.1: Usage of ';' where triples share the same subject.

```
<http://example.org/apple>  <http://example.org/hasColor> "red";
                            <http://example.org/scientificName> "Malus domestica".
```

Listing 2.2: Usage of ',' where triples differs only in the objects.

```
<http://example.org/apple>  <http://example.org/scientificName> "Malus pumila",
                                                                 "Malus domestica".
```

IRIs are written between the angle brackets like <http://example.org/#John>. Since blank nodes are locally scoped version of IRIs, the same syntax to write IRIs is also used. TURTLE syntax allows us to define *prefixes* at the head of the TURTLE file. Users could then use prefixes, to write RDF triples in a more compact form. For example, <http://example.org/#John> could be shortened to <#John> using the relative *@base* path.

Listing 2.3: Prefixes in TURTLE syntax.

```
    @base <http://example.org/> . # default base IRI
    @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
    @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
    @prefix foaf: <http://xmlns.com/foaf/0.1/> .
    @prefix rel: <http://www.perceive.net/schemas/relationship/> .
    ...
```

Literals are written between the double quote '"'. It has a default datatype of a string. One could also *cast* the literal value to a specific datatype by appending '^^ [IRI of the datatype]'. For example, "12" is cast to an integer in "12"^^xsd:integer. Language of the literal value could also be specified using @ similarly to datatype casting.

2.2 RML

RDF Mapping Language[7] is a superset of the W3C's R2RML[8] which maps relational databases to RDF datasets. RML improves upon R2RML by expressing mapping rules from heterogeneous data sources and transforming them to RDF datasets whereas R2RML could only consume data from relational databases. RML mapping file is composed of one or more *triples maps*, which in turn consist of *subject, predicate* and *object* term maps. As the names imply, the term maps are used to map elements of the data sources to their respective terms in an RDF triple. The definitions of these maps are similar to the specifications in R2RML[9].

Logical sources could be defined by specifying the *source, logical iterator* and zero or one *reference formulation* property. The logical sources in the default RML mapping file are bounded data, where the data already exists and has a predetermined size. RMLStreamer extends the vocabulary of RML to also handle unbounded data in a streaming context.

RML supports defining relationships amongst the different logical sources through the use of *rr:parentTriplesMap, rr:joinCondition, rr:child and rr:parent* properties. The relationship definitions are

Listing 2.4: An example of a RML mapping file.

Bibliography

[1] E. Miller, 'An introduction to the resource description framework,' *Bulletin of the American Society for Information Science and Technology*, vol. 25, no. 1, pp. 15–19, 1998. DOI: https://doi.org/10.1002/bult.105. eprint: https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/bult.105. [Online]. Available: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/bult.105.

[2] J. Weaver and P. Tarjan, 'Facebook linked data via the graph api,' *Semantic Web*, vol. 4, pp. 245–250, 2013.

[3] D. Phuoc, H. Nguyen Mau Quoc, H. Ngo, T. Nhat and M. Hauswirth, 'The graph of things: A step towards the live knowledge graph of connected things,' *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 37, pp. 25–35, Mar. 2016. DOI: 10.1016/j.websem.2016.02.003.

[4] A. Singhal. (2012). 'Introducing the knowledge graph: Things, not strings,' [Online]. Available: https://blog.google/products/search/introducing-knowledge-graph-things-not/ (visited on 25/12/2020).

[5] E. Prud'hommeaux and G. Carothers, 'RDF 1.1 turtle,' W3C, W3C Recommendation, Feb. 2014, https://www.w3.org/TR/2014/REC-turtle-20140225/.

[6] M. Lanthaler, D. Wood and R. Cyganiak, 'RDF 1.1 concepts and abstract syntax,' W3C, W3C Recommendation, Feb. 2014, https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[7] A. Dimou, M. V. Sande, P. Colpaert, R. Verborgh, E. Mannens and R. Walle, 'Rml: A generic language for integrated rdf mappings of heterogeneous data,' in *LDOW*, 2014.

[8] R. Cyganiak, S. Das and S. Sundara, 'R2RML: RDB to RDF mapping language,' W3C, W3C Recommendation, Sep. 2012, https://www.w3.org/TR/2012/REC-r2rml-20120927/.

[9] A. Dimou, M. V. Sande, B. D. Meester, P. Heyvaert and T. Delva, 'Rdf mapping language (rml),' IDLab - imec - Ghent University, Specification document, Oct. 2020, https://rml.io/specs/rml