Describing a network of live datasets with the SDS vocabulary

Arthur Vercruysse
Ghent University

Sitt Min Oo
Ghent University

Wout Slabbinck
Ghent University

Pieter Colpaert
Ghent University

ABSTRACT

Multiple datasets can be derived from the same data, by transforming the objects on the incoming data stream. We want to give query processors transparency in how these datasets are related and what they contain. In this paper, we introduce the Smart Data Specification for Semantically Describing Streams (SDS) to annotate datasets with provenance information, describing the consumed stream and the applied transformations on that stream. We demo the execution of a pipeline that transforms a Linked Data Event Stream and publishes the data in a different structure as described in the SDS description. The SDS vocabulary and application profile bring together DCAT-AP, LDES and P-Plan. In future work, we will create a source selection strategy for federated query processors that take into account this provenance information when selecting a dataset and interface to query the dataset.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

KEYWORDS

pandoc, productivity, awesome

ACM Reference Format:
Arthur Vercruysse, Sitt Min Oo, Wout Slabbinck, and Pieter Colpaert. 2018. Describing a network of live datasets with the SDS vocabulary. In *Proceedings of May 6–8 2018, Portland, US (XCONF'18).* ACM, New York, NY, USA, Article 4, 3 pages. https://doi.org/http://dx.doi.org/xx.xxxx/xxxxxxx.xxxxxxx

1 INTRODUCTION

Data portals publish live datasets derived from streams. Streams are often related with other streams by transforming their data. This leads to a problem: when datasets are not accompanied by provenance information, the data portal has difficulty knowing whether or not this data is already published. Also, during dataset selection query agents are clueless as to what interfaces provide the same underlying data and as a fallback, they query the same underlying dataset multiple times.

For example in a GPS application that notifies the user about changes in routes due to construction sites, multiple interfaces can be used. A time-based interface makes it easy to track the latest changes. On the other hand, a geospatial-based interface makes it easy to calculate whether or not a construction site will be encountered on a route. A SPARQL endpoint can be the solution if low availability and operation cost are of no concern[5]. Another possibility is to publish multiple Linked Data Event Streams (LDES)[13]. Each stream can be fragmented in a different way to accommodate the needs of the users.

Tailoring dataset interfaces to particular needs results in a plethora of interfaces which makes decent provenance a necessity. The provenance should cover the two steps a query agent takes to query an interface [4, 9]:

1. *Dataset discovery*: based on whether a dataset is going to contain statements that contribute to solving the query
2. *Interface discovery*: selecting the right interface that publishes the dataset

We set up a new interface for the Belgian street name registry and demonstrate that a query agent can find the optimal interface to execute particular queries.

2 RELATED WORK

DCTerms, DCAT and VoID: Exposing metadata about datasets is long established. Dublin Core Terms (DCTerms) can be used to provide basic information about resources, providing terms like *title*, *author*, *description* and *subject*[2]. Data Catalog Vocabulary (DCAT) is designed to facilitate interoperability between data catalogs published on the web[3]. DCAT also provides terms like *license*, which makes it possible to define a new license for an interface. The Vocabulary of Interlinked Datasets (VoID) focuses on explicitly linking datasets together on some predicate and defining subset datasets[1].

LDES: Linked Data Event Streams is a way of exposing an evergrowing set of immutable objects. These objects can be divided into fragments as HTTP resources that are linked together with the TREE specification [6]. Fragmentations are used to spread the items over different HTTP resources. Each HTTP resource can, for example, hold all items that start with a particular letter. A view description describes the meaning of the fragments and their links[13].

VoCaLS: Vocabulary of interoperable streams & On a Web of Data Streams (Dell Aglio): extends the ideas of DCAT with more information about streaming data[11]. The work defines a stream slightly differently than in this paper. VoCaLS focuses on streams that generate high throughput updates, this requires processors to use a windowing mechanism. In this paper, a stream is seen more broadly as a growing collection of objects, updates or otherwise.

P-Plan and PROV-O: The Ontology for Provenance and Plans (P-Plan) is an extension of the PROV-O ontology [8] created to represent the plans that guided the execution of scientific processes. P-Plan describes how the plans are composed and their correspondence to provenance records that describe the execution itself [7].

3 THE SMART DATA SPECIFICATION FOR SEMANTICALLY DESCRIBING STREAMS (SDS)

A stream in the context of SDS is a *physical* live channel that carries updates or items. A dataset can be derived from a stream as the collection of all updates or items. A *physical* channel can be any medium like a Kafka stream, WebSocket stream or even a file where updates are appended. A stream can carry any data: CSV rows, mutable or immutable linked data objects, video stream bytes, etc.

A stream can be derived from a transformation applied to items of a different stream. This transformation is described with `p-plan` in the SDS description that is part of the resulting stream. The stream and the transformation correspond with `p:plan:Entity` and `p-plan:Activity` respectively. This is shown as the pink part of Figure 1. The transformation links to the previous stream with the `prov:used` predicate. With the power of the `p-plan`, query agents can understand how datasets are linked and what interface fits a specific query the best.

The SDS description can be expanded with metadata about the dataset collected from the stream with the `sds:dataset` predicate. This way parts of the datasets' metadata can be changed after a transformation. This is represented as the green part in Figure 1.

Linking specific items to the correct stream is done with `sds:Record`. An `sds:Record` points to the data (`sds:payload`) and the corresponding stream (`sds:stream`). These small objects make it possible for multiple streams to use the same channel. Each transformation can thus push `sds:Record`'s and leave the original stream intact. A stream of immutable objects can still be transformed, for example, to calculate a hash or add a fragment id to the `sds:Record` object. The yellow part of Figure 1 gives a visual overview of `sds:Record`.

4 DEMO

Data published with Linked Data Event Streams can be partitioned or fragmented in a multitude of ways. This helps query agents resolve their queries with as few web requests

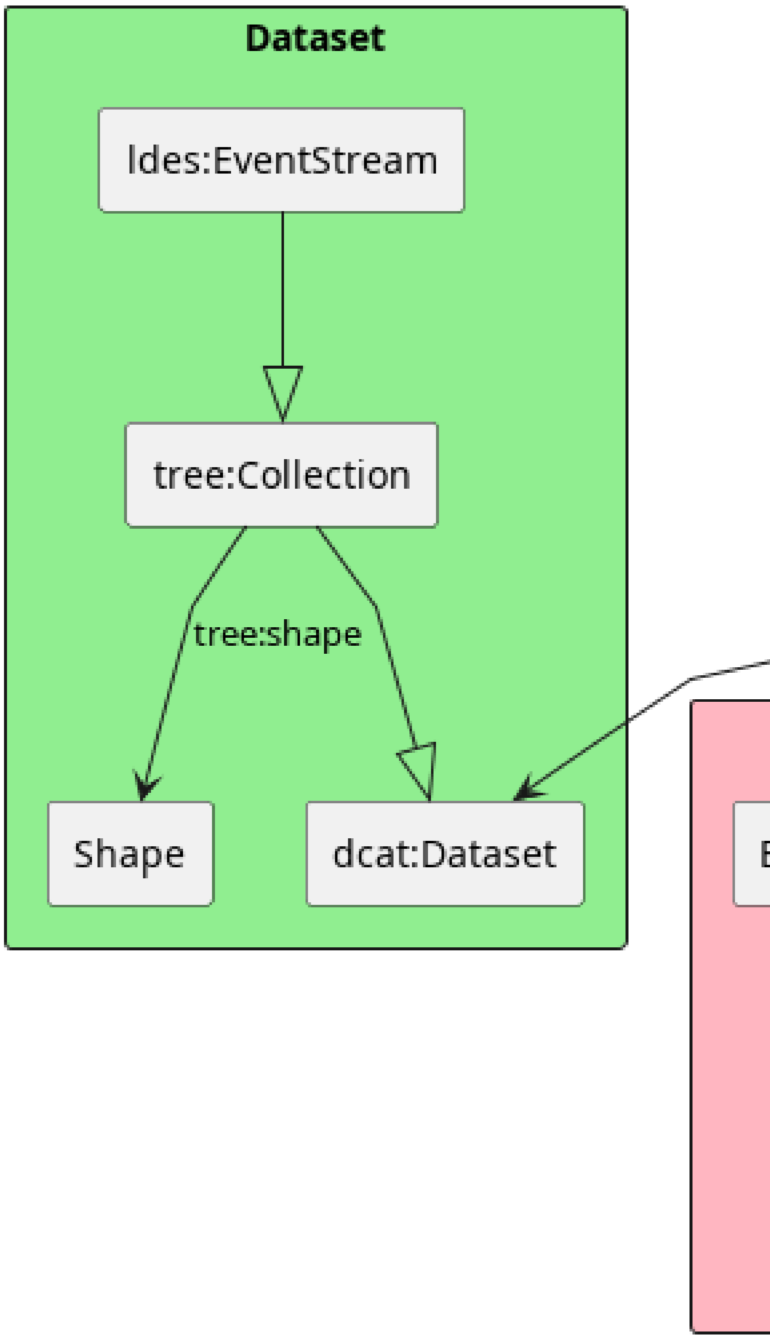

Figure 1: SDS Ontology

as possible. A default fragmentation constitutes a timestamp fragmentation, this allows clients to replicate and synchronize the dataset efficiently. A substring fragmentation, on the other hand, makes autocompletion more efficient[12].

In this demo, we set up a pipeline starting from an existing LDES that exposes the registry of street names with a timestamp fragmentation. The pipeline calculates a substring fragmentation based on the name of the street and exposes a new LDES with the corresponding SDS Description.

When asking a query agent "What are the 10 latest updated street names?" starting from the newly created LDES, the query agent can derive from the SDS description that the current LDES is not suitable for this query. This query would require the query agent to request the entire LDES tree and manually find the 10 latest updates, whereas following the links from the SDS description back to the original LDES, this query would only require a few HTTP requests. One HTTP request gets the SDS description. The other requests get the latest updates due to the timestamp-based fragmentation.

Note that the original LDES does not expose an SDS description, so this has to be bootstrapped in the pipeline.

To execute this pipeline we use a proof of concept pipeline runner called Nautirust[10]. This makes it easy to start the three required processes with the correct arguments. The three required steps are: read the original LDES with an LDES client, add buckets to the SDS Records and ingest the new SDS records in an LDES server.

5 CONCLUSION

With the introduction of the SDS ontology, it is possible to add a description to a stream and the resulting dataset, that provides provenance. The provenance links streams together and transformations applied to the stream. The SDS ontology aligns well with well-established ontologies like DCAT and P-Plan to maximize interoperability.

With the SDS description, the query agent can now automatically select the right dataset and interface based on a given query.

Federated query processors, that utilize source selection based on this provenance information when selecting a dataset and interface to query the dataset, are still future work.

6 ACKNOWLEDGMENTS

Funded by the Flemish government's recovery fund VSDS project: the "Vlaamse Smart Data Space".

REFERENCES

[1] Alexander, K. et al. 2009. Describing linked datasets. *LDOW* (2009).

[2] Baker, T. 2012. Libraries, languages of description, and linked data: A dublin core perspective. *Library Hi Tech*. 30, 1 (Jan. 2012), 116–133. DOI:https://doi.org/10.1108/07378831211213256.

[3] Beltran, A.G. et al. 2020. *Data catalog vocabulary (DCAT) - version 2*. W3C.

[4] Ben Ellefi, M. et al. 2018. RDF dataset profiling – a survey of features, methods, vocabularies and applications. *Semantic Web*. 9, (2018), 677–705. DOI:https://doi.org/10.3233/SW-180294.

[5] Buil-Aranda, C. et al. 2013. SPARQL web-querying infrastructure: Ready for action? *The semantic web – iswc 2013* (Berlin, Heidelberg, 2013), 277–293.

[6] Colpaert, P. 2022. *The tree hypermedia specification.*

[7] Garijo, D. and Gil, Y. 2014. *The P-Plan ontology.*

[8] Lebo, T. et al. 2013. *PROV-o: The PROV ontology.* W3C.

[9] Michel, F. et al. 2019. Enabling automatic discovery and querying of web apis at web scale using linked data standards. *Companion proceedings of the 2019 world wide web conference* (New York, NY, USA, 2019), 883–892.

[10] Nautirust connector architecture orchestrator: 2022. *https://github.com/ajuvercr/nautirust*. Accessed: 2022-08-24.

[11] Tommasini, R. et al. 2018. VoCaLS: Vocabulary and catalog of linked streams. *The semantic web – iswc 2018* (Cham, 2018), 256–272.

[12] Van de Vyvere, B. et al. 2022. Publishing cultural heritage collections of ghent with linked data event streams. *Metadata and semantic research* (Cham, 2022), 357–369.

[13] Van Lancker, D. et al. 2021. Publishing base registries as linked data event streams. *Web engineering* (Cham, 2021), 28–36.