Describing a network of live datasets with the SDS vocabulary

Arthur Vercruysse, Sitt Min Oo, and 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 considers this provenance information when selecting a dataset and interface to query the dataset.

Keywords: LDES, Linked Data, Provenance, Dataset selection

1 Introduction

Data portals publish live datasets derived from streams. Streams are often related to other streams by transforming their data. This leads to problems 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 data multiple times.

For example in a GPS application that notifies the user about changes in routes due to construction sites, multiple (query) 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 fulfil both interfaces, but SPARQL carries high operation costs and may leave the users with low availability [1]. Linked Data Event Streams (LDES) alleviate these high costs by publishing static web resources that can be easily cached [2]. 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 well-defined provenance a necessity. The provenance should cover the two steps a query agent takes to query an interface [3, 4]:

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*[5]. Data Catalog Vocabulary (DCAT) is designed to facilitate interoperability between data catalogs published on the web[6]. 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[7].

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. Fragmentations are used to provide semantic meaning to links between HTTP resources. Each HTTP resource can, for example, hold all items that start with a particular letter. A view description is used to define the meaning of the fragments and their links[2].

VoCaLS: Vocabulary of interoperable streams & On a Web of Data Streams (Dell Aglio): extends the ideas of DCAT with more information about streaming data[8]. 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 [9] created to represent the plans that guide the execution of scientific processes. P-Plan describes how plans are composed. This information can be used to keep track of provenance tree[10].

3 The Smart Data Specification for Semantically Describing Streams (SDS)

A stream in the context of the Smart Data Specification for Semantically Describing Streams (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 such as a Kafka stream, a WebSocket stream or even a file where updates are appended. A stream can carry any kind of data: CSV rows, mutable or immutable linked data objects, video stream chunks, etc.

A stream can be derived from a transformation applied to items of a different stream. This transformation is described with `p-plan` and resides in the SDS description 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 also links to metadata about the resulting data with the `sds:dataset` predicate. This makes modifying the datasets' metadata possible after a transformation. This is represented as the green part in Figure 1.

A `sds:Stream` should describe the type of objects it is carrying. First, the stream can use the `sds:carries` predicate that points to the type of the `sds:Records` it is carrying(see below). Next, the stream can describe the shacl shape with `sds:shape` that these members adhere to, which can only be used if the stream carries linked data objects.

Linking specific items to the correct stream is done with `sds:Record` objects. 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` objects 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 as fast as possible whilst ingesting as little data 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[11].

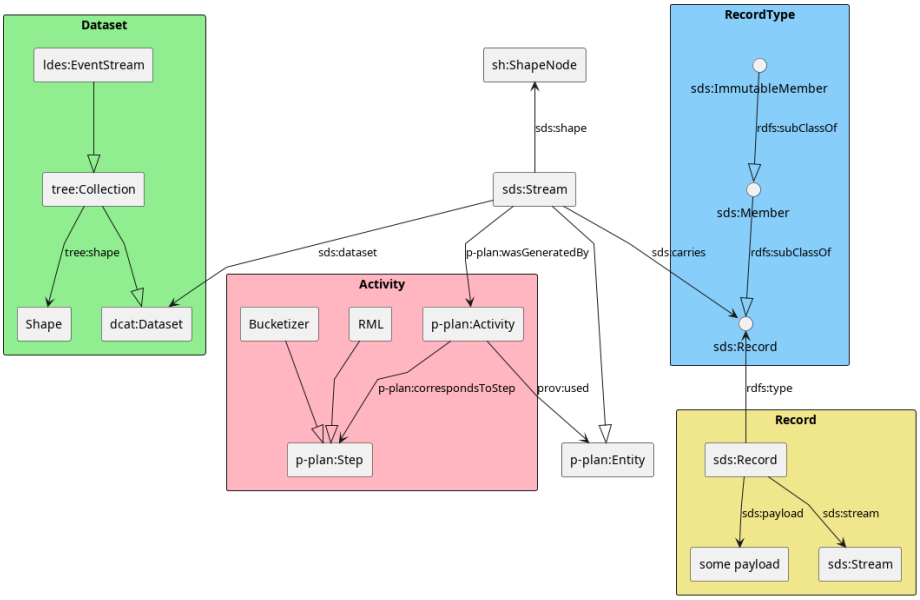

Figure 1: Visual representation of the SDS Ontology. The colored boxes have no semantic meaning and are only used for clarity.

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 and substring fragmentation. The published `tree:View`'s can then be associated with their respective `viewDescription` as described in .

```
@prefix ex:     <http://example.org/ns#>.
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ldes:   <https://w3id.org/ldes#>.
@prefix p-plan: <http://purl.org/net/p-plan#> .
@prefix prov:   <http://www.w3.org/ns/prov#> .
@prefix sds:    <https://w3id.org/sds#> .

ex:MyLDES a ldes:EventStream;
  dcat:title "An example LDES".

# One fragmentation is based on an existing LDES
ex:BasicFragmentation a tree:ViewDescription ;
  dcat:endpointURL </basic> ; # A rootnote from which you can access all members
  dcat:servesDataset ex:MyLDES ; # the LDES
  # This viewDescription was created by importing this stream
```

```
  ldes:managedBy sds:LDESStream.

# The other fragmentation bucketizes the members before publishing
ex:SubstringFragmentation a tree:ViewDescription ;
  dcat:endpointURL </substring> ; # A rootnote from which you can access all members
  dcat:servesDataset ex:MyLDES ; # the LDES
  # This viewDescription was created by importing this stream
  ldes:managedBy sds:BucketizedStream.


ex:ImportLDES a p-plan:Activity;
  rdfs:comment "Reads csv file and converts to rdf members";
  prov:used <https://smartdata.dev-vlaanderen.be/base/gemeente>.

ex:LDESStream a sds:Stream;
  p-plan:wasGeneratedBy ex:ImportLDES;
  sds:carries sds:Member.


ex:BucketizeStrategy a ldes:BucketizeStrategy;
    ldes:bucketType ldes:SubstringFragmentation; # zegt aan de client dat ik een timestampfi
    tree:path rdfs:label;
    ldes:pageSize 50.

ex:BucketizeStream a p-plan:Activiy;
  rdfs:comment "Execute a substring bucketization on the incoming stream";
  prov:used ex:LDESStream, ex:BucketizeStrategy.

ex:BucketizedStream a sds:Stream;
  p-plan:wasGeneratedBy ex:BucketizeStream;
  sds:carries sds:Member.
```

RDF sample creating two LDES Views from different Streams

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 and another request gets the latest updates due to the timestamp-based fragmentation.

To execute this pipeline we use a proof of concept pipeline runner called Nautirust[12]. 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, this adds provenance information. The provenance links streams together and transformations applied to the streams. The SDS ontology aligns well with long-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. Buil-Aranda, C., Hogan, A., Umbrich, J., Vandenbussche, P.-Y.: SPARQL web-querying infrastructure: Ready for action? In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., and Janowicz, K. (eds.) The semantic web – iswc 2013. pp. 277–293. Springer Berlin Heidelberg, Berlin, Heidelberg (2013).
2. Van Lancker, D., Colpaert, P., Delva, H., Van de Vyvere, B., Meléndez, J.R., Dedecker, R., Michiels, P., Buyle, R., De Craene, A., Verborgh, R.: Publishing base registries as linked data event streams. In: Brambilla, M., Chbeir, R., Frasincar, F., and Manolescu, I. (eds.) Web engineering. pp. 28–36. Springer International Publishing, Cham (2021).
3. Ben Ellefi, M., Bellahsene, Z., Breslin, J.G., Demidova, E., Dietze, S., Szymański, J., Todorov, K.: RDF dataset profiling – a survey of features, methods, vocabularies and applications. Semantic Web. 9, 677–705 (2018).
4. Michel, F., Faron-Zucker, C., Corby, O., Gandon, F.: Enabling automatic discovery and querying of web apis at web scale using linked data standards. Companion proceedings of the 2019 world wide web conference. pp. 883–892. Association for Computing Machinery, New York, NY, USA (2019).
5. Baker, T.: Libraries, languages of description, and linked data: A dublin core perspective. Library Hi Tech. 30, 116–133 (2012).

6. Beltran, A.G., Cox, S., Browning, D., Perego, A., Albertoni, R., Winstanley, P.: Data catalog vocabulary (DCAT) - version 2. W3C (2020).
7. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets. LDOW (2009).
8. Tommasini, R., Sedira, Y.A., Dell'Aglio, D., Balduini, M., Ali, M.I., Le Phuoc, D., Della Valle, E., Calbimonte, J.-P.: VoCaLS: Vocabulary and catalog of linked streams. In: Vrandečić, D., Bontcheva, K., Suárez-Figueroa, M.C., Presutti, V., Celino, I., Sabou, M., Kaffee, L.-A., and Simperl, E. (eds.) The semantic web – iswc 2018. pp. 256–272. Springer International Publishing, Cham (2018).
9. Lebo, T., Sahoo, S., McGuinness, D.: PROV-o: The PROV ontology. W3C (2013).
10. Garijo, D., Gil, Y.: The P-Plan ontology. (2014).
11. Van de Vyvere, B., D'Huynslager, O.V., Atauil, A., Segers, M., Van Campe, L., Vandekeybus, N., Teugels, S., Saenko, A., Pauwels, P.-J., Colpaert, P.: Publishing cultural heritage collections of ghent with linked data event streams. In: Garoufallou, E., Ovalle-Perandones, M.-A., and Vlachidis, A. (eds.) Metadata and semantic research. pp. 357–369. Springer International Publishing, Cham (2022).
12. Vercruysse, A., Oo, S.M.: Nautirust connector architecture orchestrator, https://github.com/ajuvercr/nautirust.