Publishing cultural heritage collections of Ghent with Linked Data Event Streams

Brecht Van de Vyvere[1][0000−0002−7671−6203], Olivier Van D'Huynslager[2], Achraf Atauil[3], Maarten Segers[3], Leen Van Campe[3], Niels Vandekeybus[3], Sofie Teugels[2], Alina Saenko[4], Pieter-Jan Pauwels[3], and Pieter Colpaert[1][0000−0001−6917−2167]

[1] IDLab, Department of Electronics and Information Systems, Ghent University – imec
{brecht.vandevyvere,pieter.colpaert}@ugent.be
https://idlab.technology/
[2] Design Museum Gent, Belgium
https://www.designmuseumgent.be/
{olivier.vandhuynslager,sofie.teugels}@stad.gent
[3] District09, Ghent, Belgium
https://district09.gent/
{achraf.atauil,maarten.segers,leen.vancampe,
niels.vandekeybus,pieter-jan.pauwels}@district09.gent
[4] Meemoo, Belgium
https://www.meemoo.be/
alina.saenko@meemoo.be

Abstract. Cultural heritage institutions maintain digital artefacts of their collections using Collection Management Software (CMS). In order to attract new audiences, these data should be interoperable with and reusable within other Web APIs. In this article, we explain how we applied Flemish Linked Data Standards (OSLO) to make the data within the Axiell Collections CMS interoperable, and how we applied the method of Linked Data Event Streams (LDES) for making the data reusable. The LDES has been successfully adopted by third parties to then host subject pages, a SPARQL endpoint, a substring fragmentation for autocompletion purposes, and a IIIF enriched LDES. To this end, we see LDES as the core Web API of a CMS, allowing third parties to take up other querying and processing tasks on their own machines.

Keywords: Collection Management Software · Linked Data Event Streams · Cultural Information Publication.

1 Introduction

Digital cultural heritage can play an important role in tackling urban societal issues, such as social inclusion. To this end, Collections of Ghent[5] (CoGhent)

[5] Collections of Ghent is co-financed by the European Regional Development Fund through the Urban Innovative Actions (UIA) initiative. More info: https://www.collectie.gent/

explores the role and capacity of the collections from four museums and one archive in the City of Ghent[6]. Together with stories coming from the communities themselves, these will be displayed on a multi-voice platform to spark new conversations and relations between cultural heritage organizations (CHOs) and their communities, thus fostering a more inclusive reading of our cultural heritage. The collections of the CHOs and the communities' digitized stories need to be integrated into this platform by replicating and synchronizing the data. Currently, these CHOs use the Collections Management Software (CMS) Axiell Collections (AC)[7], which exposes their collection objects' metadata with an Application Programming Interface (API). This API offers different functionalities, such as an advanced search language statement and autocompletion. However, the platform of Coghent needs to integrate data from heterogeneous sources and therefore requires the usage of persistent identifiers and Linked Data at the source, which are not supported by the CHO's version (Adlib Xplus 4.3), and the Linked Data must be exposed with a replication and synchronization API to efficiently create an up-to-date copy of the data. In this article, we will explain how such a Linked Data API can be created on top of AC, and how, among others, a Digital Asset Management System (DAMS) can use this API to enrich the images of objects with metadata stored in AC.

Since 1999, replicating and synchronizing the data repository of a CHO has been performed with the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) protocol [1]. By executing specific HTTP calls in a linear fashion, the whole repository can be retrieved in XML format. One major drawback of this approach is that Representational State Transfer (REST) API design [2] is not implemented decoupling client and server in a stateless fashion. With OAI-PMH, clients can ask sets of records within any time interval and when the response is too big to send over one response, the first part of the result is returned with a resumption token. With this token, the client can request the next part and so forth. As a consequence, the server must maintain the state of every client separately. Also, this approach is not suited for the replication and synchronization of specific resources, such as updates from paintings. In 2012, ResourceSync solved these gaps by specifying different methods (Resource List, Resource Dump, Change List, Change Dump) to retrieve updates from resources [3]. These methods are described with a sitemap allowing clients to discover changes in a hypermedia-driven approach. However, sitemaps are originally created for search engines and do not use the Resource Description Framework (RDF) [4], which is a requirement for Linked Data publishing[8]. Starting from publishing Linked Data with data dumps, subject pages or SPARQL endpoints, it became clear that Linked Data can be published in several ways and in 2014, the term 'Linked Data Fragments' (LDF) was introduced to define how a dataset is fragmented and which hypermedia controls and metadata are added [5]. With this LDF vision, data publishers need to decide on trade-offs whether to put

[6] Design Museum Gent, STAM, Industriemuseum, Huis van Alijn, and Archief Gent
[7] `https://www.axiell.com/nl/oplossingen/product/axiell-collections/`
[8] `https://5stardata.info/en/`

all the data processing effort on the server-side (SPARQL endpoint) or apply a more fragmented data strategy (subject pages, Triple Pattern Fragments [6]) shifting part of the querying cost to clients. In 2019, the Linked Data Event Streams (LDES) specification was created for the replication and synchronization of Linked Data [7]. Although a one-dimensional pagination strategy is put forward as best practice, similar to Hydra [8], Activity Streams[9] and International Image Interoperability Framework (IIIF) Change Discovery[10] [9], other fragmentation strategies can be researched using the TREE hypermedia specification[11]. As a result, LDES does not only allow third parties to integrate the data into their own systems, it also gives third parties the opportunity to take part of the publication effort with derived LDESs, reusable indexes and query services.

Given this background, the main issues addressed in this paper are a) how cultural heritage institutions can extend their AC CMS with LDES and b) how integrating LDES by a third party leads to integrated advanced services. This article begins by giving an overview of the TREE and LDES specifications. It will then explain how CoGghent's CHOs created an LDES on top of their AC instance and give an overview of derived query services. Finally, we conclude how LDES tackles the challenges related to the management of cultural information.

2 Background

TREE specification The hypermedia specification TREE[12] allows to publish objects of a collection, called members. A member is part of a collection that describes the data graph of its member using shapes[13], and can broadly be defined as prov:Entity being a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary[14]. This way, TREE can be used cross-domain for slow (cultural heritage objects) and fast (sensor observations) moving data. The collection's members are published over multiple pages on the Web where each page can be seen as a node of a traversable tree and contains qualified relations to other nodes. A TREE relation describes which property and value of members can be expected when a client follows the relation to another node. Specific TREE relations are available, such as *GreaterThanRelation* and *GeospatiallyContainsRelation*, to compare strings, geospatial features and time literals giving clients the ability to prune the search tree. Another aspect of TREE is that the collection can have multiple views where the view refers to the root node from where all members can be retrieved thus multiple

[9] `https://www.w3.org/TR/activitystreams-core/`
[10] `https://iiif.io/api/discovery/1.0/`
[11] `https://w3id.org/tree/specification`
[12] `https://w3id.org/tree/specification`
[13] Shapes can be described with Shapes Constraint Language (SHACL) or Shape Expressions (ShEx)
[14] `https://www.w3.org/TR/prov-o/#Entity`

fragmentation strategies can be applied to one collection, such as geospatial clusters to search efficiently (speed, bandwidth) over certain regions [10], and prefix trees to solve text search queries on the client-side [11].

Linked Data Event Streams An LDES is a specific type of TREE collection where its members are immutable. If objects need life cycle management, such as cultural heritage objects, immutability is achieved through versioned objects. Each time an object changes, a new immutable version is created. As a consequence, a client only needs to fetch the objects once and can maintain a cache of processed members. It is the responsibility of the data publisher to maintain LDESs for their collections of objects from where other parties can replicate and synchronize. LDES is positioned in a three-layered architecture (LDES, indexes and query services) by the Semantic Interoperability Community (SEMIC)[15] where third parties can offer indexes and query services using an LDES as base data source. For example, LDES is implemented by the Flemish government for the publication of base registry data, such as address related information [12].

3 Extending Axiell Collections API with Linked Data Event Streams

In this section, we will describe how we implemented an LDES on top of an AC API. Our method is composed of two steps. First, an Extract Mapping and Loading (ETL) pipeline is used to generate the versioned Linked Data objects. Then, another LDES building block bundles these objects inside hypermedia-driven LDFs. Fig. 1 illustrates the various stages of the process, which we will discuss in depth hereafter. This section also describes four query services that use an LDES with cultural heritage data as input.

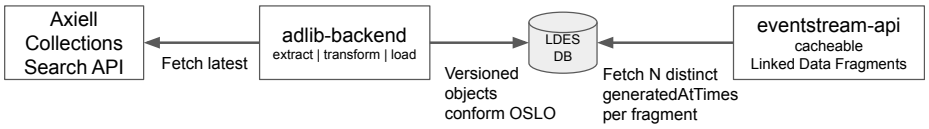

Fig. 1. Creating an LDES from Axiell Collections uses two building blocks and an LDES DB.

3.1 Extraction, Transformation and Loading

The LDES generation component "eventstream-api" (Section 3.2) requires querying over the list of immutable objects of the LDES collection to bundle these objects in fragments. An ETL pipeline called "adlib-backend" is created that i)

[15] `https://w3id.org/ldes/specification`

periodically fetches the latest objects from an AC API, ii) maps the objects to a Linked Data model and iii) loads them in a relational database, which we will call the LDES DB.

Extraction The ETL pipeline needs to periodically run to fetch the latest objects from the AC CMS. The AC Search API is used and allows, among others, to configure an output format (JSON or XML), a database, pagination size and search parameters. To simplify explaining how the latest objects can be fetched, we will use in the rest of the examples for the query parameters as shown in Listing 1.1: JSON as format (format=json), "collection objects" as database (database=objecten), and a page size of 10 objects (start=0&limit=10). For the search parameters, we will set the institution's name, e.g. "Het Huis van Alijn (Gent)", and web publication flag (webpublication=EUROPEANA). The latter is used by the institutions of CoGhent to indicate that the object metadata may be used as Open Data.

Listing 1.1. Base query parameters used.

```
?output=json&database=objecten&startFrom=0&limit=10
&search=webpublication=EUROPEANA
AND institution.name='Het Huis van Alijn (Gent)' AND ...
```

On the first run, no objects from the institution's database have been retrieved thus running the query in Listing 1.2 can be used in a loop by increasing the startFrom parameter. When objects are already harvested in the LDES DB, the generatedAtTime of the latest versioned object is retrieved and must be used in the Modified search parameter:

Listing 1.2. Search parameters to retrieve the latest objects.

```
Modified > 'timestamp-last-retrieved-object'
```

However, in AC, records are sorted by their internal identifier (priref record number) and not by their modified time. As a result, when a run has not been completed, records will be missed in the next run. Therefore, before running the query from Listing 1.3, an extra query needs to retrieve all records that were modified before the last retrieved object and that has a priref identifier that is greater than the last retrieved object's priref (Listing 1.3):

Listing 1.3. Search parameters to complete the previous run.

```
Modified <= 'timestamp-last-retrieved-object'
    AND priref > 'last-retrieved-priref'
```

Transformation The extracted Adlib records are returned in JSON format and need to be transformed to standardized Linked Data information models to achieve semantic interoperability with other related Linked datasets. In the next paragraphs, we will describe how this is done i) by creating URIs and ii) mapping to Linked Data models.

The first step in creating Linked Data is the usage of HTTP URIs as identifiers for entities instead of the local identifier (priref). For LDES, two URIs need to be created: the version object URI and the persistent object URI.

A field is reserved in AC where the institution can fill in the persistent URI when this is available. For example, when institutions host their own URIs using a resolver. If a persistent URI is available, the versioned object URI will extend this URI with a timestamp. When this is empty, the following URI template is used based on the Flemish URI standard[16]: `https://{domain}/{type}/{concept}(/{reference-basic})+(/{reference-version})`. CoGhent makes use of the https:// protocol and uses the domain of the City of Ghent's website (stad.gent). All entities are given 'id' as *type* to clarify that the URIs are identifiers. The *concept* depends on the type of entity. For example, human-made objects have the concept 'mensgemaaktobject' (Dutch for human-made object). Two *reference-basics* are used. First, a namespace is given per collection, which aligns with the database name in AC. Second, an MD5 hash will be generated based on the local identifier (*priref*) and the record's creation time. In the current version, the second reference basic is set to the local identifier only. *Reference-version* is used for versioned objects and contains the record's modification time. For example, an object of Design Museum Gent (dmg) that has been modified at 2021-08-15T01:56:04.152Z has a URI `https://stad.gent/id/mensgemaaktobject/dmg/530027447/2021-08-15T01:56:04.152Z`.

The second step towards Linked Data is mapping the extracted Adlib records to, among others, the CIDOC-Conceptual Reference Model (CRM) information model [13]. CIDOC-CRM provides the concepts and relations to describe entities in the cultural heritage domain and has been an official ISO standard since 2006. In Flanders (Belgium), its usage is promoted with the use of two Application Profiles (APs) created by the Open Standards for Linked Organizations (OSLO) standardization programme [14]. While both OSLO and Linked Art[17] use CIDOC-CRM as a semantic layer, OSLO is an inter-domain effort using the same approach for non-cultural heritage domains. An AP operates in the context of use cases and specifies which concepts and relations of vocabularies need to be reused. Also, a closed-world assumption is created by adding cardinality constraints. The first AP[18] focuses on static concepts, such as a human-made object, information object, work, expression and collection, while the second AP[19] focuses on cultural heritage related activities, such as the creation and acquisition of an object. JavaScript Object Notation Linked Data (JSON-LD) is used as the format and uses a JSON-LD context file for each AP. Listing 1.4 demonstrates the mapping of a human-made object, which refers to the two context files and indicates when the versioned object has been created (prov:generatedAtTime) and the persistent, non-versioned URI of the object (dcterms:isVersionOf). How OSLO maps JSON-LD terms to URIs is also exemplified by repeating the class "MensgemaaktObject" in the context to

[16] `https://data.vlaanderen.be/cms/VlaamseURI-StandaardVoorData_V1.0.pdf`
[17] `https://linked.art/`
[18] `https://data.vlaanderen.be/doc/applicatieprofiel/cultureel-erfgoed-object`
[19] `https://data.vlaanderen.be/doc/applicatieprofiel/cultureel-erfgoed-event`

its CIDOC-CRM URI. Furthermore, the institution that maintains the object ("MaterieelDing.beheerder") is demonstrated. To allow semantic reconciliation, contextualisation and alignment across institutions, URIs referring to external vocabularies, such as Wikidata and Getty's AAT, ULAN and TGN, were added to AC. Mapping relations between source and target models are established by creating a JSON object in the Javascript programming language to which JSON-LD mappings can be added. A Javascript function is created for each feature (descriptions, associations, iconography) and appends JSON-LD objects that are aligned with the OSLO context files to the main JSON object. With this mapped JSON-LD object in mind, the last step of the ETL pipeline can be performed.

Listing 1.4. Snippet of a versioned human-made object.

```
{
"@context": [
  "https://apidg.gent.be/opendata/adlib2eventstream/v1/
     context/cultureel-erfgoed-object-ap.jsonld",
  "https://apidg.gent.be/opendata/adlib2eventstream/v1/
     context/cultureel-erfgoed-event-ap.jsonld",
  {
    "dcterms": "http://purl.org/dc/terms/",
    "prov": "http://www.w3.org/ns/prov#",
    "MensgemaaktObject": "http://www.cidoc-crm.org/
       cidoc-crm/E22_Man-Made_Object"
  }
],
"@id": "https://stad.gent/data/mensgemaaktobject/dmg/53
   0027447/2021-09-11T01:56:30.635Z",
"@type": "MensgemaaktObject",
"dcterms:isVersionOf": "https://stad.gent/id/
   mensgemaaktobject/dmg/530026077",
"prov:generatedAtTime": "2021-09-11T01:56:30.635Z",
"MaterieelDing.beheerder": "http://www.wikidata.org/
   entity/Q1809071"
}
```

Load Mapped JSON-LD objects are stored in the payload column of the "Members" table in a relational database (Table 1). Next to inserting the URI, institution, generation time and AC database name, also the software version number of adlib-backend at the time of running the ETL pipeline is inserted. Institutions may request new features to the mapping algorithm, such as adding links to other external thesauri. When this happens, the ETL pipeline needs to extract every object again and append the mapped objects to the LDES. This way, older version objects are preserved and the life cycle of an object can be retained. Detecting whether the mapping algorithm has changed requires checking if the software version of "adlib-backend" has changed since the latest

software version in the database (version column). In the next section, we will describe how an LDES can be constructed on top of the Members table.

Table 1. Columns of the Members table.

Column Name	Definition
URI	Web identifier of versioned object
version	Software version of adlib-backend
institution	Institution name
adlibDatabase	Database name of Adlib
generatedAtTime	Timestamp when versioned object was created
payload	JSON-LD mapping of object

3.2 Linked Data Event Stream

The LDES generation component[20] called "eventstream-api" needs to construct an LDES by applying a fragmentation strategy over the version objects in the Members table. We will explain in this section how we implemented this strategy following the three parts of a Linked Data Fragment [5]: i) data: which objects will be selected per fragment, ii) controls: the relationships added between fragments, and iii) metadata: data about the knowledge graph.

Data The term 'data' is used here for the objects in an LDES. To emphasize the autonomy and decentralized nature of an institution, we decided in the CoGhent project to create an LDES of collection objects per institution. This selection is performed by filtering on the "institution" and "adlibDatabase" columns. Filtering also needs to be performed to limit the number of objects per fragment. A one-dimensional fragmentation based on the generation time of the version object has been implemented, because clients will only need to subscribe to the latest page to retrieve updates. New versions are generated during the working hours of an institution, thus fragmenting with static time periods, e.g. every 10 minutes, would lead to a large number of empty fragments. To evenly spread objects over fragments, we group N distinct generatedAtTime timestamps T per fragment ($N > 0$) from the sorted list of distinct Ts. As a result, every fragment corresponds with N Ts and contains the version objects that have a T greater or equal than the fragment's lowest T and less than the next fragment's lowest T.

Controls Hypermedia controls are added between fragments following the TREE specification. Every fragment has two TREE relations: to go forward and backward in time. These relations are qualified links indicating to clients which objects of the LDES collection can be expected when the link is followed. Listing 1.5 demonstrates a LessThanRelation between two fragments,

[20] Available as Open Source: `https://github.com/StadGent/node_service_eventstream-api`

which are instances of *tree:Node*, to retrieve objects that contain T before '2021-08-15T01:52:51.561Z'. Also, the number of remaining items is added to each relation to indicate how many objects can be retrieved through that link.

Listing 1.5. Example of a TREE relation to retrieve version objects with a generatedAtTime value less than 2021-08-15T01:52:51.561Z.

```
{
  @id: "https://apidg.gent.be/opendata/adlib2
     eventstream/v1/dmg/objecten?generatedAtTime=2021-
     08-15T01:52:51.561Z",
  @type: "tree:Node",
  tree:relation: {
    @type: "tree:LessThanRelation",
    tree:node: "https://apidg.gent.be/opendata/adlib2
       eventstream/v1/dmg/objecten?generatedAtTime=202
       1-08-15T01:51:51.569Z",
    tree:path: "prov:generatedAtTime",
    tree:value: "2021-08-15T01:52:51.561Z",
    tree:remainingItems: 1325
}
```

Metadata Every object in the LDES is linked with the LDES collection through a tree:member relation. By dereferencing the collection URI, a Data Catalog Application Profile (DCAT-AP) description is returned where data about the LDES can be retrieved, such as title, license, and maintainer. Figure 2 shows how the three layers (metadata, controls, and data) are linked with each other: an LDES corresponds with a DCAT dataset and all objects can be retrieved by following the accessURL on a distribution or by choosing a TREE view directly. Version objects of the information model are embedded inside fragments thus do not require an explicit link with its *tree:Node*.

3.3 External query services

In the CoGhent project, the two LDES generation components from previous sections are deployed on behalf of the institutions by the ICT partner of Ghent (District09) in a production environment with the use of Docker containers[21]. By replicating and synchronizing with these LDESs, which can be done with the LDES client[22], other actors in the ecosystem can create derived query services. In this section, we describe how query services are created by three different organizations (the City of Ghent, Inuits[23], and IDLab) in the context of CoGhent.

Subject pages The transformation step in Section 3.1 creates HTTP URIs using the cities' domain, because the city is responsible for resolving the URIs with relevant information (Linked Data principle 3). To make this possible, the

[21] https://github.com/StadGent/docker_adlib2eventstream
[22] Available as library: https://github.com/TREEcg/event-stream-client
[23] https://inuits.eu/

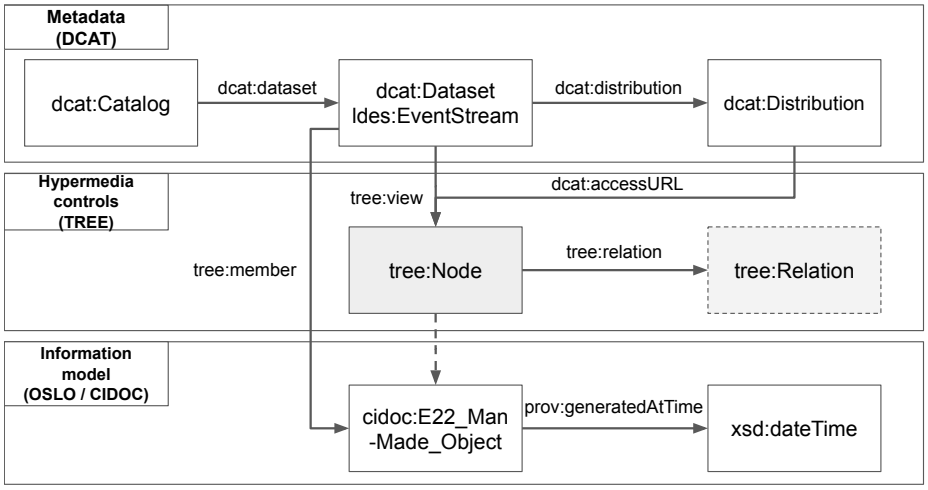

Fig. 2. Overview how metadata, hypermedia controls and information objects are linked.

City of Ghent hosts a Virtuoso triple store for Linked Open Data publication and has created a Linked pipes configuration to import an LDES[24]. From this triple store, a service returns subject pages with relevant triples when a URI containing `https://stad.gent/` is dereferenced.

SPARQL endpoint The city of Ghent also exposes a SPARQL endpoint[25] on top of their triple store to publish cross-domain semantic data related to the city, such as public services, news articles, and parking availability. All version objects of the object from Listing 1.4 can now be retrieved with a SPARQL query having following triple pattern: $?version < http://purl.org/dc/terms/isVersionOf >< https://stad.gent/id/mensgemaaktobject/dmg/530027447 >$. This query can be used when the version URI from Listing 1.4 does not work anymore as the city decides which versions to preserve. If a third party wants to set up their own SPARQL endpoint: an open source Software Development Kit (SDK) is being developed to deploy multiple services that replicate and synchronize with LDESs[26]. One of these services is to set up a GraphDB triple store[27]. From there, a SPARQL endpoint can be deployed.

IIIF enriched LDES AC is used for the management of collection metadata. However, high-resolution images of objects still need to be maintained in a dedicated Digital Asset Management System (DAMS). The DAMS used by the CoGhent institutions have the LDESs integrated to link images with their corresponding object metadata using the object identifier[28]. As a result, the DAMS

[24] `https://github.com/district09/ldes-to-linkedpipes/`
[25] `https://stad.gent/sparql`
[26] `https://github.com/Informatievlaanderen/ldes2service`
[27] `https://github.com/osoc21/ldes2service/tree/ldes-graphdb-connector`
[28] `https://gitlab.com/inuits/dams`

will publish derived LDESes where the versioned objects contain links to the IIIF image API (3.0) and presentation API (3.0). These enriched LDESes will serve for the creation of innovative applications (multi-voice platform, cocreation funding, hackathons) in the context of Coghent.

Substring index One of the benefits of LDES is the use of the allround hypermedia specification TREE. Instead of describing TREE relations to traverse in time, also other fragmentations, such as geospatial and text-based fragmentations, can be expressed. A substring index has been created to demonstrate how (thesauri) objects of an LDES can be fragmented by substring[29]. For example, a fragment B contains a limited set of objects having a label starting with "b". With a Substring relation having a value "bo", objects having a label starting with "bo" can be retrieved. These relations are especially useful for autocompletion applications[30].

4 Conclusion and future work

In CoGhent, semantic and technical interoperability is achieved using interdomain efforts: the OSLO APs are used to semantically align the cultural information within the broader semantic ecosystem of Flanders, and LDES is used for the replication and synchronization of the data towards other parties. This article gave some insights how a CMS of Axiell can be extended with an LDES using two components. This publication strategy can be generalized to other JSON APIs with search functionality. Also, the versioned objects allow knowledge to be inferred that is not contained in a single source and addresses topics, such as historiography. With the LDES publication paradigm, others can more easily integrate the data within their advanced query services. For example, the City of Ghent already had a subject page and SPARQL endpoint service running and now also supports an integrated view for cultural heritage data. Compared to sensor data, cultural heritage objects update slowly. However, a retention policy on the side of the LDES will still be necessary in the long-term. To this end, collaboration with archives may be needed.

We also showed that an LDES collection can be fragmented using TREE for text search applications. Not only objects and thesauri from the CoGhent institutions can be retrieved with a substring query, also public sector information (Flemish registry for street names) or external thesauri (Netwerk Digitaal Erfgoed) can be queried ad hoc by integrating the data on the client-side. By adding links to external thesauri, such as AAT, ULAN, TGN, we want to generate translations of descriptive texts automatically in future work. Finally, an enriched LDES with IIIF promises to be an interesting approach to replicate and synchronize CMSs and DAMs in both ways. To conclude, we advocate for the support of LDES as a standard Linked Data service at the level of CMSs.

[29] `https://github.com/TREEcg/substring_fragmenter`

[30] An autocompletion demo can be found here: `https://tree.linkeddatafragments.org/demo/autocompletion/?datasets[]=https://treecg.github.io/demo_data/vtmk.ttl,https://oliviervd.github.io/substrings_coghent/agents/root.ttl`

References

1. Herbert Van de Sompel, Michael L Nelson, Carl Lagoze, and Simeon Warner. Resource harvesting within the oai-pmh framework. *D-lib magazine*, 10(12), 2004.
2. Roy T Fielding and Richard N Taylor. Principled design of the modern web architecture. *ACM Transactions on Internet Technology (TOIT)*, 2(2):115–150, 2002.
3. Bernhard Haslhofer, Simeon Warner, Carl Lagoze, Martin Klein, Robert Sanderson, Michael L. Nelson, and Herbert Van de Sompel. Resourcesync: Leveraging sitemaps for resource synchronization. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13 Companion, page 11–14, New York, NY, USA, 2013. Association for Computing Machinery.
4. Ora Lassila, Ralph R. Swick, World Wide, and Web Consortium. Resource description framework (rdf) model and syntax specification, 1998.
5. Ruben Verborgh, Miel Vander Sande, Olaf Hartig, Joachim Van Herwegen, Laurens De Vocht, Ben De Meester, Gerald Haesendonck, and Pieter Colpaert. Triple Pattern Fragments: a low-cost knowledge graph interface for the Web. *Journal of Web Semantics*, 37–38:184–206, March 2016.
6. Ruben Verborgh, Miel Vander Sande, Olaf Hartig, Joachim Van Herwegen, Laurens De Vocht, Ben De Meester, Gerald Haesendonck, and Pieter Colpaert. Triple pattern fragments: a low-cost knowledge graph interface for the web. *Journal of Web Semantics*, 37:184–206, 2016.
7. Dwight Van Lancker, Pieter Colpaert, Harm Delva, Brecht Van de Vyvere, Julián Rojas Meléndez, Ruben Dedecker, Philippe Michiels, Raf Buyle, Annelies De Craene, and Ruben Verborgh. Publishing base registries as linked data event streams. In *Proceedings of the 21th International Conference on Web Engineering*, May 2021.
8. Markus Lanthaler and Christian Gütl. Hydra: A vocabulary for hypermedia-driven web apis. In *LDOW*, 2013.
9. Stuart Snydman, Robert Sanderson, and Tom Cramer. The international image interoperability framework (iiif): A community & technology approach for web-based images. In *Archiving conference*, volume 2015, pages 16–21. Society for Imaging Science and Technology, 2015.
10. Harm Delva, Julián Andrés Rojas, Pieter-Jan Vandenberghe, Pieter Colpaert, and Ruben Verborgh. Geospatial partitioning of open transit data. In *International Conference on Web Engineering*, pages 305–320. Springer, 2020.
11. Ruben Dedecker, Harm Delva, Pieter Colpaert, and Ruben Verborgh. A file-based linked data fragments approach to prefix search. In *International Conference on Web Engineering*, pages 53–67. Springer, 2021.
12. Dwight Van Lancker, Pieter Colpaert, Harm Delva, Brecht Van de Vyvere, Julián Rojas Meléndez, Ruben Dedecker, Philippe Michiels, Raf Buyle, Annelies De Craene, and Ruben Verborgh. Publishing base registries as linked data event streams. In *International Conference on Web Engineering*, pages 28–36. Springer, 2021.
13. Crm Cidoc. The cidoc conceptual reference model, 2003.
14. Raf Buyle, Laurens De Vocht, Mathias Van Compernolle, Dieter De Paepe, Ruben Verborgh, Ziggy Vanlishout, Bjórn De Vidts, Peter Mechant, and Erik Mannens. Oslo: Open standards for linked organizations. In *Proceedings of the international conference on electronic governance and open society: Challenges in Eurasia*, pages 126–134, 2016.