Monitoring water using an RDF-based Water Data Space

Shehabeldeen Abdelfatah[1,2,3][0000−0001−9136−7745],
Julián Andrés Rojas[1][0000−0002−6645−1264],
Janelcy Alferes[2,3][0000−0002−0772−3135], and
Pieter Colpaert[1][0000−0001−6917−2167]

[1] IDLab – Ghent University – imec
[2] Flemish Institute for Technological Research (VITO NV)
[3] Centre for Advanced Process Technology for Urban Resource Recovery (CAPTURE)
Shehab.ayman@vito.be

Abstract. Water locks monitor river stage and discharge for raising and lowering boats between stretches of water of different levels on a canal or river. In the WaterFRAME project, we're building a "water data space" to bring together all relevant data producers and consumers in this ecosystem, and agree on how we want data to be discovered, made interoperable and trusted. In this demonstrator, we show a prototype visualizing all relevant data for a waterlock from disparate sources. We experimented with Linked Data Event Streams (LDES) as a source format and found that combining it with a technique called RDF Time Series Snippets reduces the object count, affecting ingestion efficiency. We ingested the data in a consumption framework and experimented with Virtuoso, Oxigraph, and Postgres. The demo guides you through interactive modules for historical data browsing, plotting, and SPARQL querying.

Keywords: Water Management · Linked Data Time Series · Linked Data Event Streams · RDF Time Series Snippets.

1 Introduction

A water lock is a specialized enclosure with gates at each end used for raising and lowering boats between stretches of water of different levels on a canal or river. Because locks function by manipulating water volume to equalize these levels, monitoring them requires precise synchronization of disparate data streams. In this demonstration, we examine a water lock on the Dessel-Kwaadmechelen canal in Flanders. We utilize linked data published by Waterinfo[4] to gain a multidimensional view essential for operational management.

[4] https://waterinfo.vlaanderen.be/

2 Synchronizing Time Series Data with LDES and TSS

Hydrological readings can be modeled as an incremental feed of sensor readings. Therefore, we are experimenting with Linked Data Event Streams (LDES) specification[5] [4], built on top of the TREE hypermedia specification[6] [1], as a light-weight publishing API. An LDES client can be used by an interested consumer to replicate the history, and stay synchronized with the feed afterwards. In listing 1.1, we provide an example of an LDES fragment with a temporal relation to another fragment.

Listing 1.1: LDES fragment defining an event stream.

```
@prefix ldes: <https://w3id.org/ldes#> .
@prefix tree: <https://w3id.org/tree#> .
@prefix sosa: <http://www.w3.org/ns/sosa/> .

<#ES> a ldes:EventStream ;
    ldes:timestampPath sosa:resultTime ;
    tree:view <> .
<> tree:relation [
    a tree:GreaterThanOrEqualToRelation ;
    tree:node <2025.trig> ;
    tree:path sosa:resultTime ;
    tree:value "2025-01-01T00:00:00+00:00"^^xsd:dateTime
].
```

When publishing raw `sosa:Observations`, each event in the feed will need to be described using ±10 triples, only to add a new sensor reading. Instead, we propose the feed to be updated with a single event for a time series snippet. RDF Time Series Snippets (TSS)[7] is a proposal for a lossless compaction of triples describing a time series into one literal per snippet, as illustrated in listing 1.2, similar to SPARQL CDTs [2] or RDF Tensors [3].

Listing 1.2: TSS optimization for bundling sensor observations.

```
@prefix tss: <https://w3id.org/tss#> .
@prefix sosa: <http://www.w3.org/ns/sosa/> .

<snippet/73617010_20240911000000> a tss:Snippet ;
    tss:from "2024-09-11T00:00:00"^^xsd:dateTime ;
    tss:points "[{\"time\": \"2024-09-11T00:00:00\", \"value
        \": \"29.69\"}, ... ]" ;
    tss:to "2024-09-11T23:45:00"^^xsd:dateTime ;
    tss:about [
        sosa:observedProperty ex:RiverStage .
    ] .
```

[5] https://w3id.org/ldes/specification
[6] https://w3id.org/tree/specification
[7] https://knowledgeonwebscale.github.io/rdf-timeseriessnippets/

3 Demonstrator of a water data space

Our demonstrator as illustrated in fig. 1, visualizes time-series data by automating ingestion into a comparative storage layer featuring Virtuoso, Oxigraph, and Postgres as they are widely used, open-source RDF stores. PostgreSQL was included as a baseline to evaluate the performance trade-offs between structured SQL and graph-based architectures. To ensure interoperability, data from Waterinfo is modeled using SSN/SOSA, QUDT, and GeoSPARQL ontologies. Finally, a Node.js backend executes standardized queries across these engines to serve an interactive React-based frontend.

Within the proposed system, we identify three primary actor roles involved in the data life cycle:

- **Data Producers:** Government bodies publishing sensor observations.
- **Data Space Providers (WaterFrame[8]):** Manage the ingestion pipeline, apply Time Series Storage (TSS) optimizations, and maintain the stores.
- **Data Consumers:** End-users, such as water scientists, utilize the dashboard to execute queries and visualize trends.

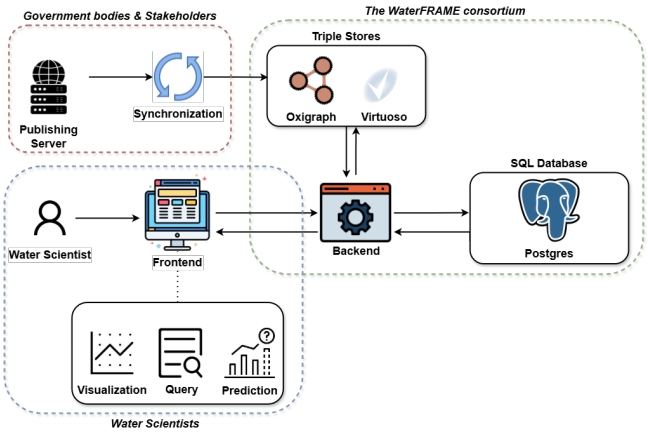
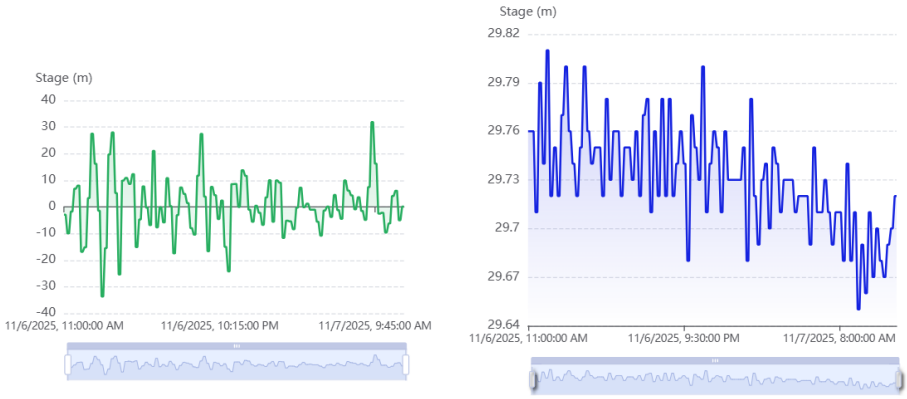


Fig. 1: High-level architecture of the demonstration framework.

The React dashboard allows interactive engagement with time series data through several modules:

- **Data Browsing & Station Info:** Raw tabular data and geographical station information via an interactive map.
- **Interactive Visualization:** Graphing for River Stage and Discharge (Fig. 2).
- **SPARQL Querying:** A sandbox for executing custom queries directly against the triple stores.

[8] `https://capture-resources.be/projects/waterframe`



(a) River Discharge (b) River Stage

Fig. 2: The figure illustrates the synchronized relationship between river stage and discharge, where periodic discharge fluctuations correspond to the changes in water levels maintained during lock operations.

4 Conclusion

Integrating LDES provides a foundation for replication and synchronization with hydrological data. Implementing TSS atop this stream achieves efficiency in storage and querying. These specifications provide a method for storage and querying intended for decision support systems.

The repository for the demo implementation is available at: `https://github.com/ShehabEldeenAyman/ESWC-2026-Demo.git`. The video demo is available at `https://github.com/ShehabEldeenAyman/ESWC-2026-Demo/blob/main/ESWC%20Demo%202026.mp4`

Acknowledgments This work has been carried out within the FWO-SBO WaterFRAME project (S001124N). The data used in this project is publicly available through `https://waterinfo.be`.

References

1. Colpaert, P.: Building materializable querying interfaces with the tree hypermedia specification. In: MEPDaW@ ISWC. pp. 8–18 (2022)
2. Hartig, O., Williams, G.T., Schmidt, M., Lassila, O., Enriquez, C.M.L., Thompson, B.: Datatypes for lists and maps in rdf literals. In: European Semantic Web Conference. pp. 234–238. Springer (2024)
3. Marciniak, P., Sowiński, P., Ganzha, M.: Representing and querying data tensors in rdf and sparql. In: European Semantic Web Conference. pp. 62–66. Springer (2025)
4. 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: International Conference on Web Engineering. pp. 28–36. Springer (2021)