Continuous generation of versioned collections' members with RML and LDES

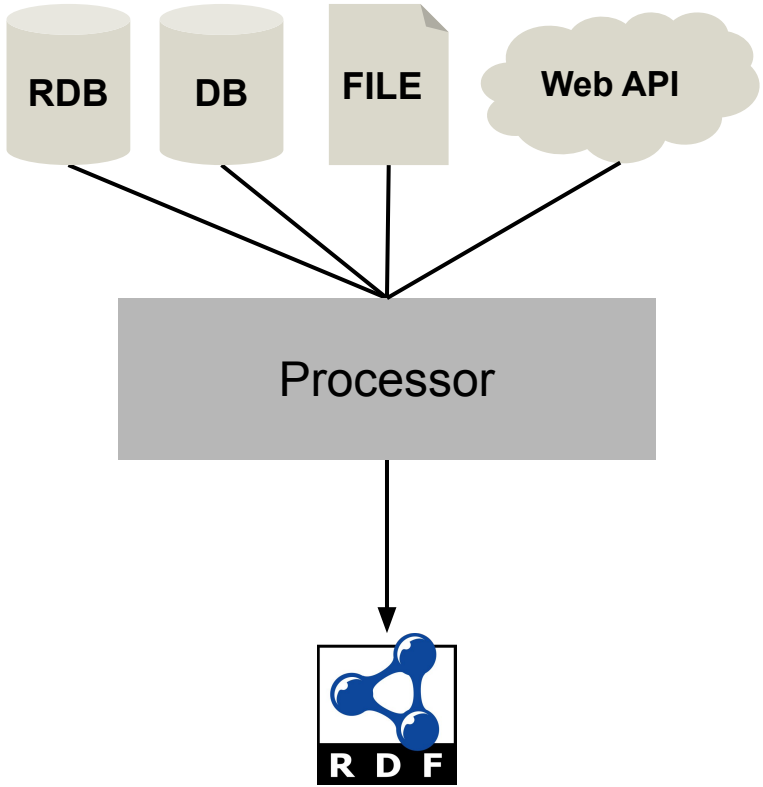
Dylan Van Assche, Sitt Min Oo, Julián Andrés Rojas, Pieter Colpaert

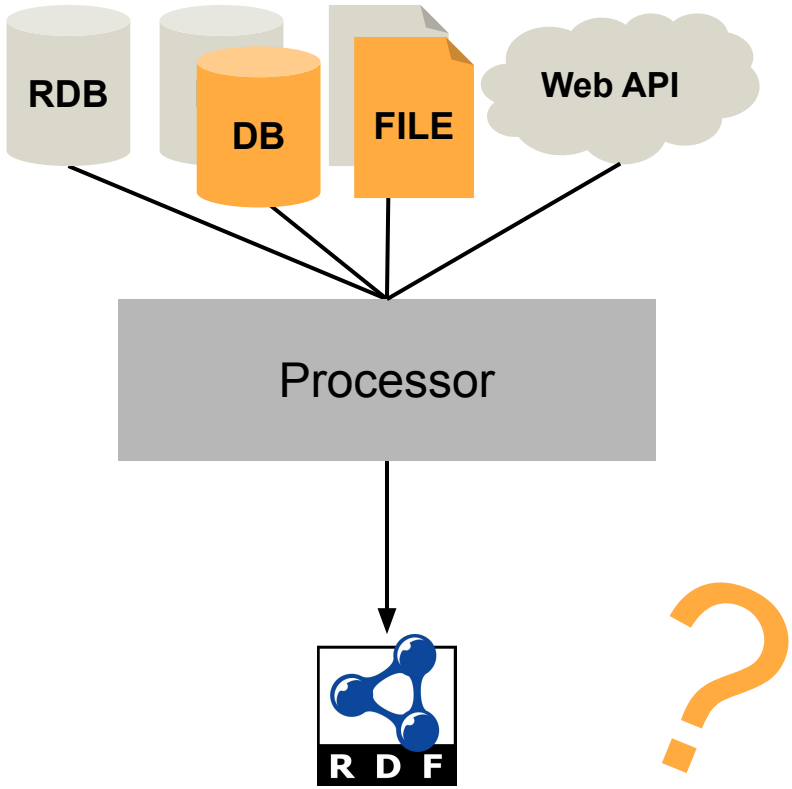
🐦 @DylanVanAssche
📨 dylan.vanassche@ugent.be
🌍 dylanvanassche.be

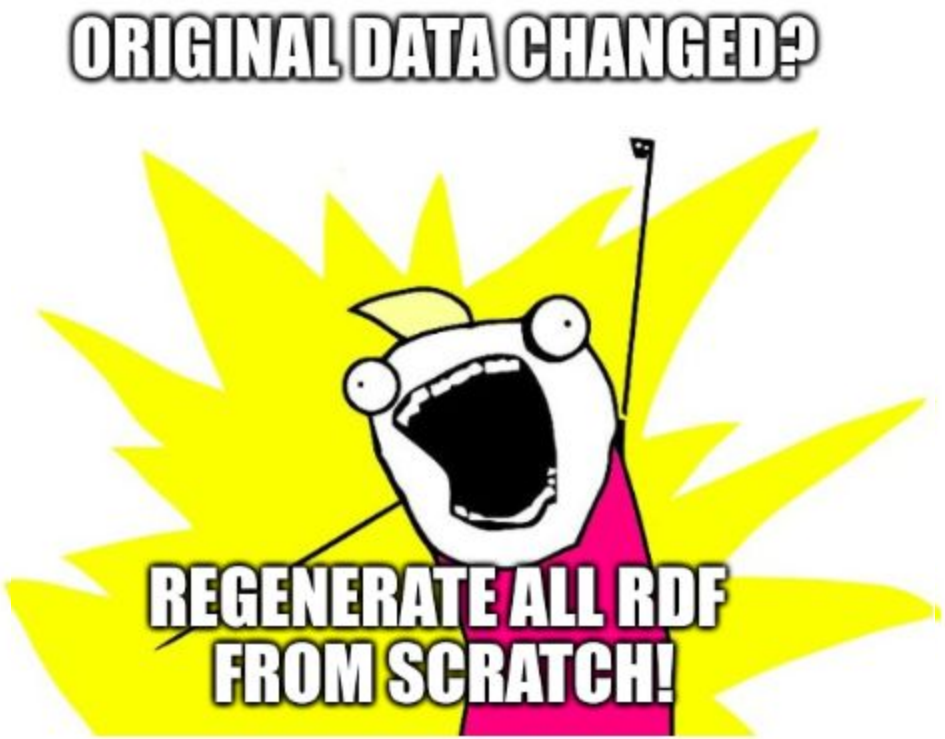
IDLab
INTERNET & DATA LAB

UNIVERSITEIT
GENT

imec


RDB
DB
FILE
Web API
Processor
R D F

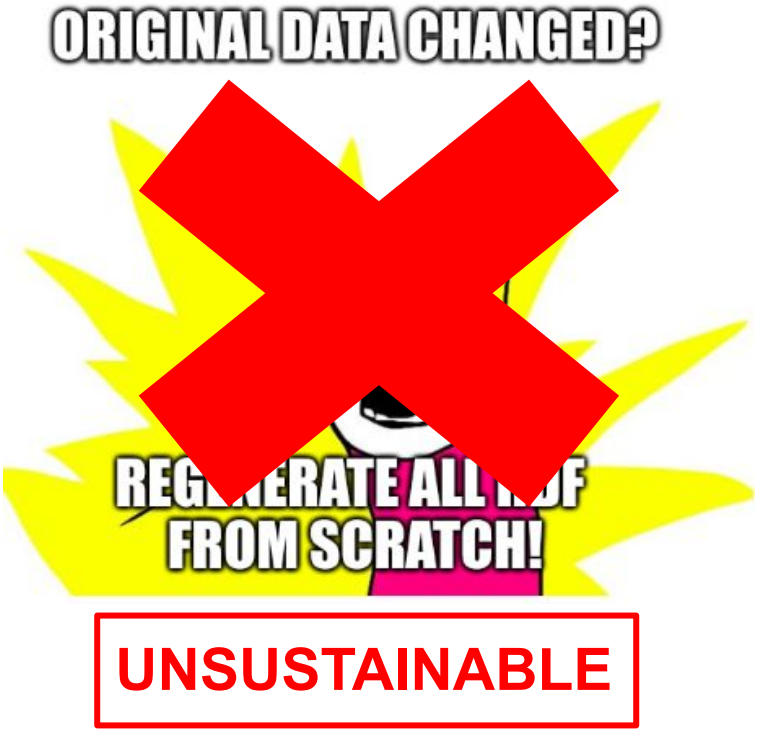

RDB
DB
FILE
Web API
Processor
R D F
?


ORIGINAL DATA CHANGED?
REGENERATE ALL RDF
FROM SCRATCH!

Regenerating all RDF for each update is unsustainable

- Waste of computing resources
- Hard to keep if the data changes rapidly
- History is lost

Data is not static, but we assume so when generating RDF from data sources.



How can we efficiently model and process source data updates for Knowledge Graph publishing?

Overview

Linked Data Event Streams for publishing updates

Data collection types

Generating unique & reproducible IRIs for updates

Applying to 3 use cases

Overview

Linked Data Event Streams for publishing updates

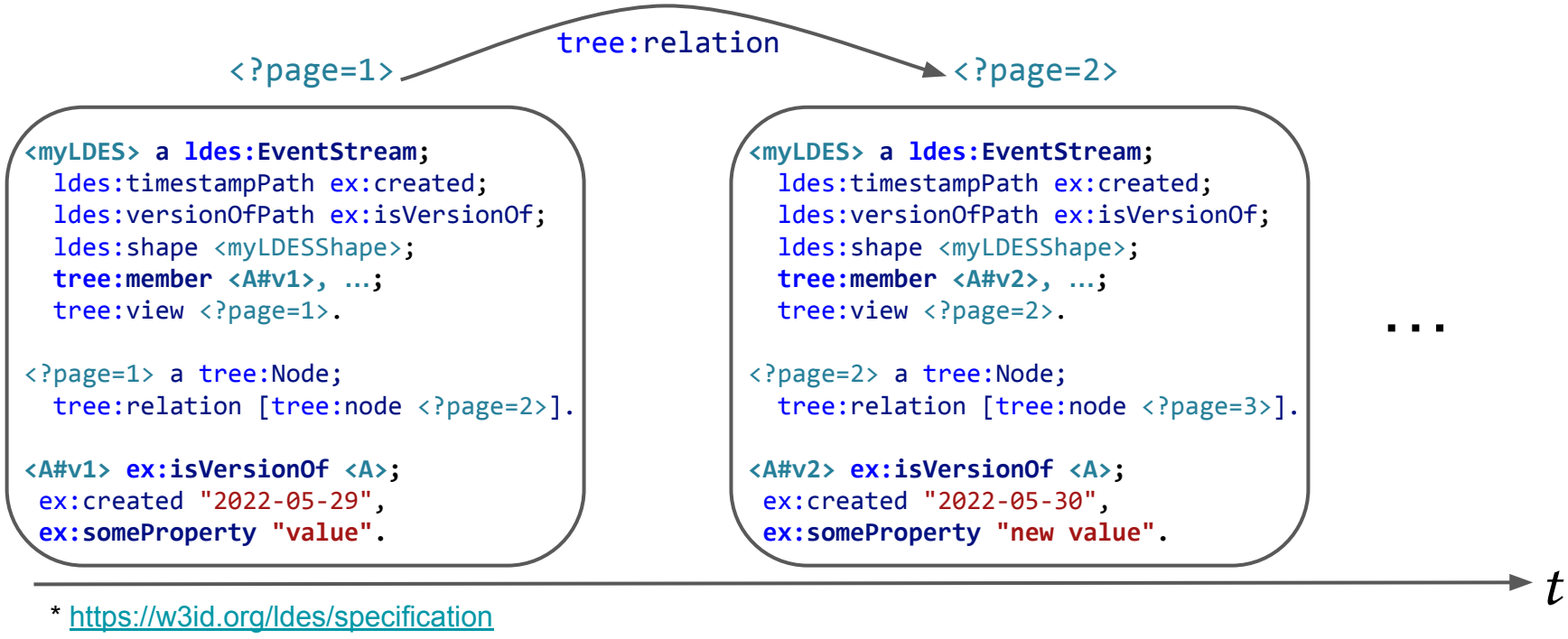
Data collection types

Generating unique & reproducible IRIs for updates

Applying to 3 use cases

Linked Data Event Streams (LDES)

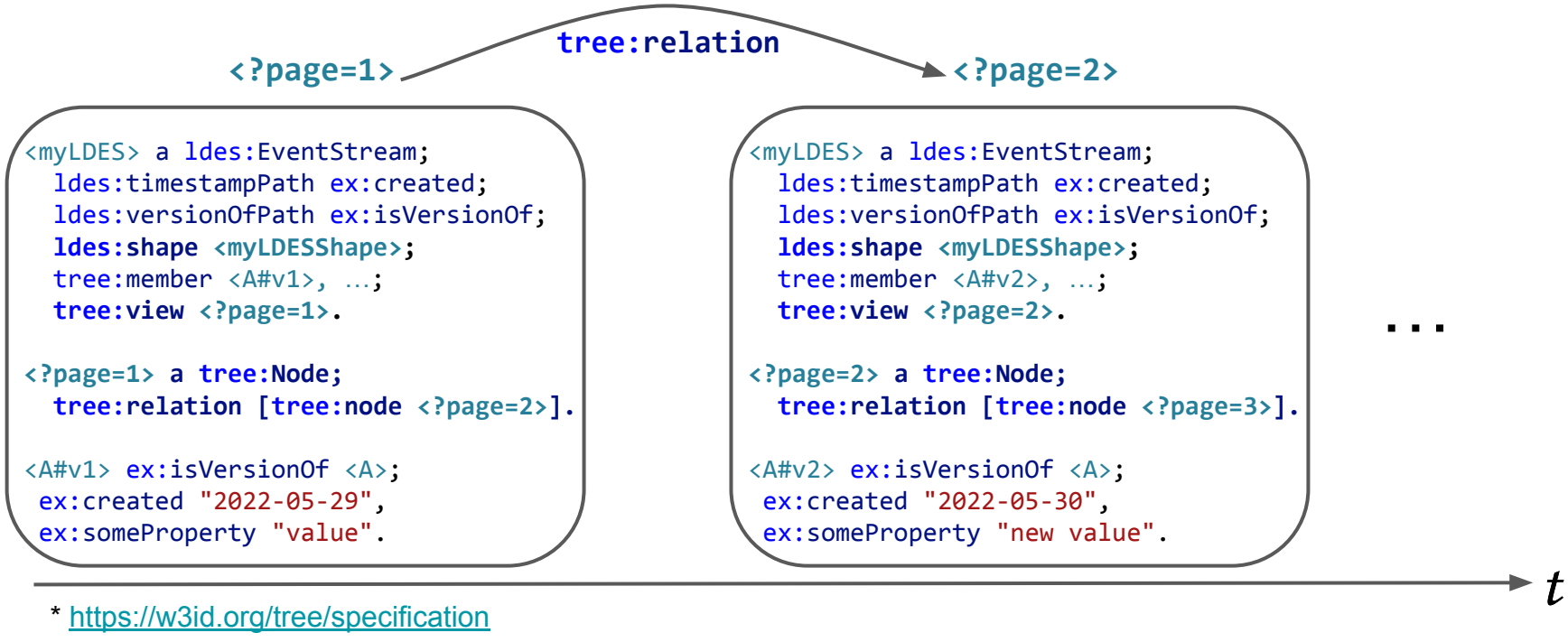
An LDES* describes the **stream of changes** of topic-based data sources, as a **collection** of **immutable** data objects (a.k.a. collection **members**).



* https://w3id.org/ldes/specification

Linked Data Event Streams (LDES)

LDES leverages the **TREE specification*** for describing how to traverse the collection and how each member is structured through a SHACL shape

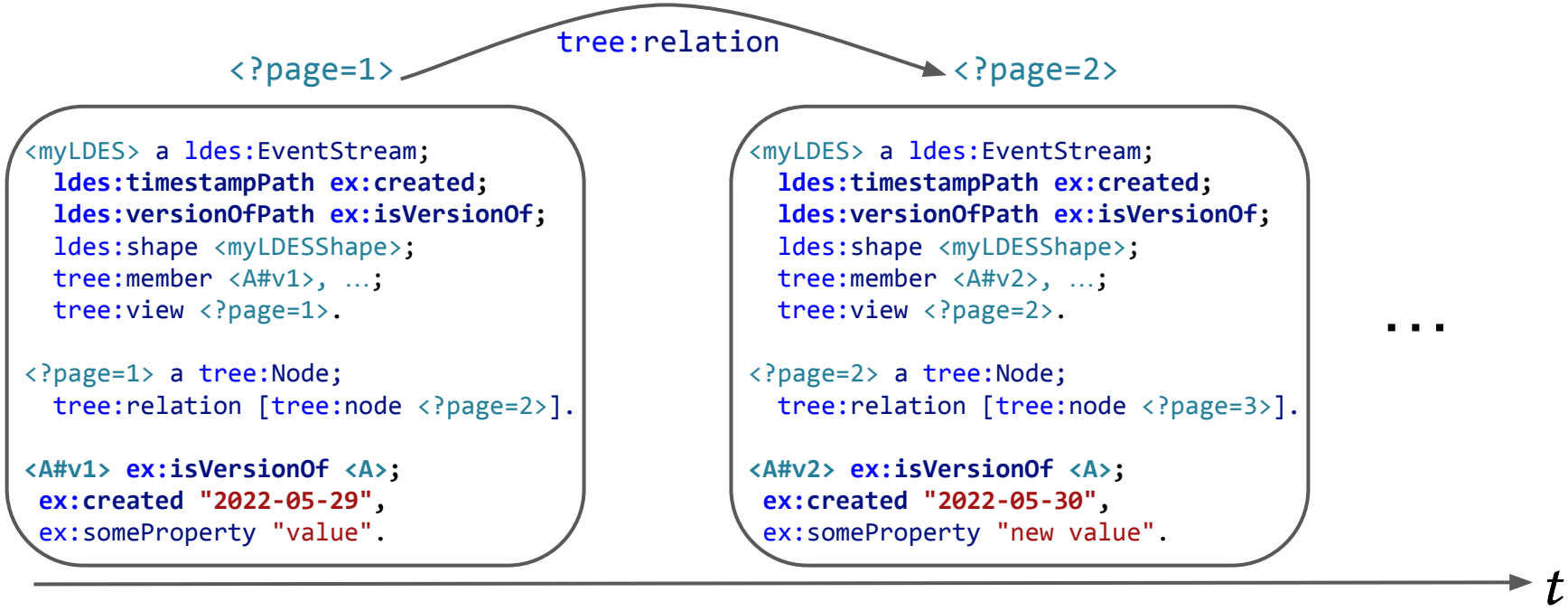


* https://w3id.org/tree/specification

Linked Data Event Streams (LDES)

Multiple path descriptions are possible to traverse the history e.g.

`lds:timestampPath` or `lds:versionOfPath`



Overview

Linked Data Event Streams for publishing updates

Data collection types

Generating unique & reproducible IRIs for updates

Applying to 3 use cases

Data collection types regarding immutability and history

Immutability

- Immutable collection
- Mutable collection

A collection is **immutable** if **unique IDs** are available for **each version** of all collection's members

History

- Latest state
- Latest changes
- Full history

Collections provide a **dump of the latest version** (latest state), only the **latest updates** (latest changes), or **several versions of all members** (full history)

Data collection types regarding immutability and historic provision

History	Immutable	Description	Example
Latest state	Yes	Complete dump, unique IDs	Bike-sharing data
Latest state	No	Complete dump, non-unique IDs	GTFS timetables
Latest changes	Yes	Delta updates, unique IDs	OpenStreetMap diffs
Latest changes	No	Delta updates, non-unique IDs	GTFS-RealTime
Full history	Yes	Complete dump, versions history	KMI meteorological data

Overview

Linked Data Event Streams for publishing updates

Data collection types

Generating unique & reproducible IRIs for updates

Applying to 3 use cases

Unique & reproducible IRIs

Immutable collections

Unique & reproducible IRIs can be generated from **unique IDs** across **versions** such as hash codes or timestamps.

```
{
   "name": "someName",
   "property1": "value"
   "modified": "2022-05-29T18:00:00.000Z"
}
// new version
{
   "name": "someName",
   "property1": "new value"
   "modified": "2022-05-29T20:00:00.000Z"
}
```

```
<http://ex.org/someName#2022-05-29T18:00:00.000Z>

# new version

<http://ex.org/someName#2022-05-29T20:00:00.000Z>
```

Member IRI template:

```
http://ex.org/{name}#{modified}
```

Unique & reproducible IRIs

Mutable collections

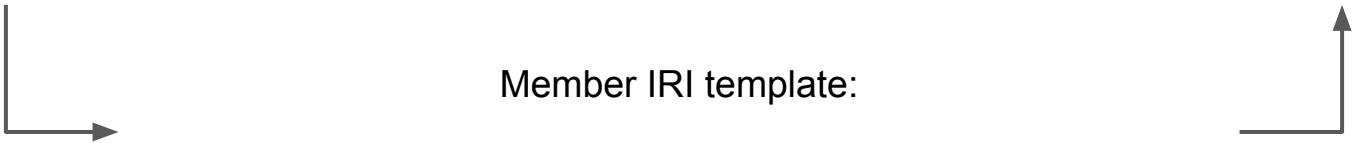
Unique & reproducible IRIs require **externally generated input**. E.g., current system timestamp, hashcode based on member properties

```
{
   "name": "someName",
   "property1": "value"
}
// new version
{
   "name": "someName",
   "property1": "new value"
}
```

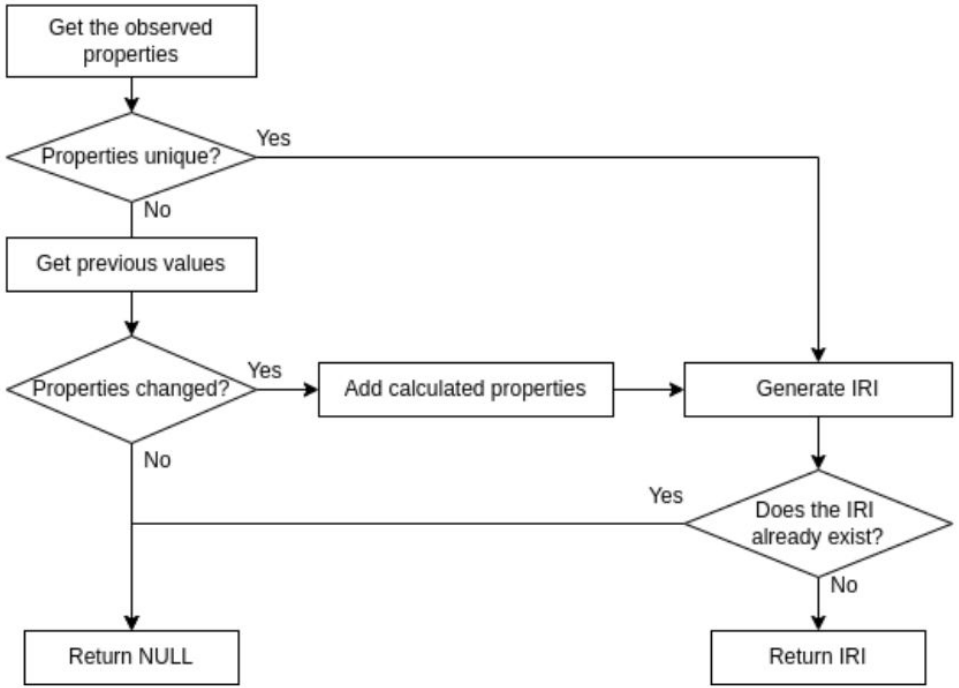
```
<http://ex.org/someName#134235500>

# new version

<http://ex.org/someName#134235700>
```



Unique & reproducible IRI generation flow



Extending RML Logical Target with LDES properties

Introduce a LDES Logical Target which extends RML Logical Target with LDES properties such as:

- SHACL shape IRI descdribing each collection member (`tree:shape`)
- Path descriptions (`ldes:timestampPath`, `ldes:versionOfPath`) describing how to traverse the generated LDES
- Retention policy for history (`ldes:retentionPolicy`) specifying how long history will be kept e.g. none, last 20 versions, last hour, etc.

LDES Logical Target also generates a `tree:member` property to list all IRIs of the members in the LDES as required by the LDES and TREE specifications.

Overview

Linked Data Event Streams for publishing updates

Data collection types

Generating unique & reproducible IRIs for updates

Applying to 3 use cases

Use case 1: BlueBike bike-sharing data

Immutable latest state data collection

- Update timestamp for each station (`last_seen`)
- Dumps the latest state of all stations
- Minutely updates

Generating IRIs with a template
Leverage unique timestamps:

`http://ex.org/station/{id}#{last_seen}`


```
[
   {
       "id": 48,
       "name": "Station Antwerpen-Centraal",
       "bikes_available": 30,
       "bikes_in_use": 17,
       "last_seen": "2022-02-24T14:16:43"
   },
   {
       "id": 49,
       "name": "Brussel Centraal",
       "bikes_available": 8,
       "bikes_in_use": 4,
       "last_seen": "2022-02-24T14:18:46"
   },
   ...
]
```

Use case 2: KMI meteorological data

Full historical data collection

- Immutable, unique IDs (`FID`) for each version
- Historical versions are dumped as well
- Updates every 10 minutes

Generating IRIs by a template
Leverage unique version IDs:

`http://ex.org/station/{code}#{FID}`

FID	code	temp	humidity
aws_10min.6418.2022-02-01 10:00:00.0	6418	1017	7
aws_10min.6472.2022-02-01 10:00:00.0	6472	985	2
aws_10min.6418.2022-02-01 10:10:00.0	6418	1017	8

Use case 3: NMBS GTFS timetables

Mutable latest state data collection

- No unique properties for each version
- Dumps the latest state of all stations
- Daily updates

Generating IRIs by observing properties

Observing **`parent_station`** and **`stop_id`**
If they are updated, generate IRI with template:

`http://ex.org/station/{`**`parent_station`**`}`
`/stop/{`**`stop_id`**`}#{`***`current_system_time`***`}`

Generation timestamp appended to make IRI unique

stop_name	parent_station	stop_id
Bruxelles-Central	**S8813003**	**8813003_5**
Bruxelles-Central	**S8813003**	**8813003_1**
Bruxelles-Central	**S8813003**	**8813003_2**

Preliminary tests

Use case	Type	Average dump size	Average number of triples reduction	Average generation time reduction
BlueBike bike-sharing data	immutable latest state	0.029 Mb	4.6x	1.1x
KMI meteorological data	immutable historical	1.9 Mb	17.0x	1.2x
NMBS GTFS timetables	mutable latest state	55 Mb	33.7x	24.7x

https://github.com/RMLio/RML-LDES-mapping-rules/blob/main/PRELIMINARY_RESULTS.md

Preliminary tests

Use case	Type	Average dump size	Average number of triples reduction	Average generation time reduction
BlueBike bike-sharing data	immutable latest state	0.029 Mb	4.6x	1.1x
KMI meteorological data	immutable historical	1.9 Mb	17.0x	1.2x

Generation time is **not always reduced** since the **processor's start up time** is bigger than the RDF generation time for small datasets

Preliminary tests

Use case	Type	Average dump size	Average number of triples reduction	Average generation time reduction
Small change in a large data collections, results in a **higher reduction** for number of triples and generation time				
NMBS GTFS timetables	mutable latest state	55 Mb	33.7x	24.7x

https://github.com/RMLio/RML-LDES-mapping-rules/blob/main/PRELIMINARY_RESULTS.md

Recap

Linked Data Event Streams for publishing updates

Data collection types

Generating unique & reproducible IRIs for updates

Applying to 3 use cases

Conclusion & future work

Source **data updates** should be **considered** when generating RDF

Promising preliminary results by only **generating RDF for updated data members e.g. number of triples and generation time reduced**.

In the future, we will extend our approach to **handle deletions** and perform an extensive **performance evaluation**

Dylan Van Assche

PhD Student
IDLab - Ghent University - imec

🐦 @DylanVanAssche
📨 dylan.vanassche@ugent.be
🌍 dylanvanassche.be

IDLab
INTERNET & DATA LAB

UNIVERSITEIT
GENT

imec