# Linked Data Enrichment with Self-Unfolding URIs

Barnabás Szász *, Rita Fleiner**, András Micsik***
* University of Debrecen, Debrecen, Hungary
** Óbuda University, Budapest, Hungary
*** MTA, SZTAKI, Budapest, Hungary
bszasz@gmail.com, fleiner.rita@nik.uni-obuda.hu, micsik@sztaki.mta.hu

***Abstract*— Linked Data resources are identified by Uniform Resource Identifiers. It is an important step in any Linked Data project to define the conventions for URI assignments. In some cases resources already have their natural identifiers, or they can be inherited from previous databases. However, there are cases when frequent insertions of triple sets occur without any convenient way for identification and grouping of them. In this paper we elaborate on a mechanism that makes handling complex and frequent insertions easier, and also provides the benefits of simple authoring together with rich querying and reasoning on the data. We show how to eliminate some of the time consuming and error prone aspects of Linked Data authoring by introducing the self-unfolding URI concept. This solution generates RDF description to entities based on information encoded in their URIs. For the generation of these new RDF triples we propose templates that can be implemented by SPARQL Insert queries.**

## I. INTRODUCTION

The principles of Linked Open Data are related to publishing and interlinking structured data on the Web so that computers can read it automatically. This method enables data from different sources to be connected and queried. The Linked Data concept - invented by Tim Berners-Lee in 2006 – is based on the following four principles [1]:

1. Use URIs as names for things
2. Use HTTP URIs, so that people can look up those names
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
4. Include links to other URIs, so that they can discover more things

The 4th recommendation ensures the links within the different datasets. The standard data model for Linked Open Data is the Resource Description Framework (RDF). In RDF data is structured in triples in the form of subject, predicate and object, which is called a statement. The predicate specifies how the subject and object are related. The subject and the predicate are both URIs and the object is a URI or a string literal. SPARQL is an RDF query language, designed to retrieve and manipulate data stored in RDF format. Linked Data builds links between arbitrary things described in RDF. In RDF, URIs identify any kind of object or concept.

Publishing Linked Data on Linked Data Platform [2] demands certain best practices, e.g. to provide meaningful URIs to identify entities. For human usage – especially for manual entry – it is important to encode some semantics in the URI structure. This can lead to redundancy in the data, as the same information might be represented as RDF triplets, describing the entity and in the URI as well. If this redundancy cannot be avoided, some automated mechanism should take care of the maintenance or the consistency verification.

We propose the Self-unfolding Semantic URI concept: these are URIs following a specific pattern and a template, which describes the structure the entity should have. The pattern of the appearing new resource URI identifies the template that is used to generate a set of triples providing basic semantic description of the resource, and thus enabling better querying and reasoning for the new resource. Let's suppose a LOD dataset is given and there is a mechanism (e.g. a trigger) monitoring the data. In case of a special type of entity appears, the system generates new RDF triples that are semantically derived from the original ones.

Automatic Linked Data expansion can be categorized by (i) the characteristics of the data that triggers the automatic data generation, (ii) the method of the generation, (iii) the structure of the new triples and (iv) the mechanism supporting the automatism. The process is indispensable for efficient data management of certain types of LOD datasets. For example, a proper, re-usable OWL-Time Interval description requires at least 7-8 triples, which is quite tedious and error-prone for manual input.

The focus of this paper is the automatic expansion of Linked Data sets, which is a special case of Linked Data enrichment. We show how to eliminate the time consuming steps of Linked Data authoring with introducing the self-unfolding URIs, by generating RDF description to entities based on information encoded in their URI. In Section 2 we describe research fields that are related to the topic. In Section 3 application and data specific use cases are described where automatic unfolding of URIs might be useful, while in Section 4 scenarios are presented that are independent from the application area and the type of the data. Section 5 shows examples for the implementation details of the enrichment. Section 6 provides some conclusive remarks.

## II. RELATED WORK

Enriching Linked Data is the process leveraging implicit or hidden semantics built into a dataset and made explicit by RDF tools. Previous studies in this field

focused mainly on link generation to semantically related resources, which in most cases means the automatic insertion of owl:sameAs statements between instances of different resources, (e.g. to DBpedia or to Geonames datasets) [3] or between the corresponding concepts (e.g. ontology classes or properties) [4]. The latter is the main objective of ontology alignment [5] and schema alignment [6]. Answering complex queries often requires accessing and combining information from multiple datasets, which can be achieved by federated query processing. Different approaches to federated query processing over Linked Data are analyzed in [7]. The authors study how different design alternatives affect the performance and practicality of query processing and define a benchmark for federated query processing, comprising a selection of data sources in various domains and representative queries.

Semantic Sensor Web is the combination of sensor and Semantic Web technologies, where the encoding of sensor descriptions and observation data with Semantic Web languages enables more expressive representation, advanced access, and formal analysis of sensor information. Dynamic enrichment is a current research topic in the field of linked sensor data streams. In [8] authors propose the use of a Complex Event Processing engine with a dynamic enrichment component that expands the sensor information items before evaluating them. The authors suggest a prototype to realize situation awareness over large-scale and open web sensor networks. For the core processing model of the enrichment a relational query model is used.

Transforming sensor-based data into RDF and making it available using HTTP requires the use of sensor data related URIs. In [9] authors propose a URI-based mechanism to identify and access Sensor Data coming from sensor networks and they propose several URI design to represent Time, Space and sensor identity information in URIs. In [10] the most relevant challenges of the Semantic Sensor Web are described, where the integration and fusion of data coming from different sensor networks (with varying qualities of service and different throughput rates, geographical scales) and other sources (e.g. static data or archived sensor data) is emphasized.

Correndo, et al. [11] address the issue of representing time entities (i.e. Instants and Intervals) as Linked Data, and describe how to exploit topological temporal relationships in order to increase the connectivity degree within Linked Data sets. They present an approach to describe temporal entities as reusable URIs that can be adopted by data publishers as a temporal context to their information resources. The approach identifies a set of discrete temporal entities as relevant for a certain domain (e.g. financial years for the public sector) and a RESTful API is provided to dynamically create temporal entities. Once a dynamic temporal URI is resolved, information is provided to situate such URI in reference to the relevant domain entities. The URI resolution employs simple topological temporal reasoning in order to exploit the qualitative relationships between entities.

## III. USE CASES

This section highlights use cases for URI unfolding. We have identified a temporal and a spatial use case and a third one combining these.

### A. OWL Time entities

There are multiple ways to model temporal information, but probably the most used ontology for this purpose is the OWL Time Ontology[1]. It provides basic constructs to define and describe points and intervals bounded with a start and endpoint in the temporal space. OWL Time provides two approaches to describe a point of time: either using the xsd:datetime datatype or using the DateTimeDescription class. While the first one offers an easy way to define a point of time by a well-structured string, it lacks some of the features the DateTimeDescription class provides: it uses seconds as the default unit type and requires a full date and time described. On the other hand, manually modeling and maintaining DateTimeDescription entities are error prone and tiring, because these require at least 7-8 triples in a format that is really re-usable in a semantic sense. For example, defining a time interval entity requires the following triples:

```
ex:meetingInterval
 a time:Interval;
 time:hasBeginning ex:meetingStart;
 time:hasDurationDescription ex:meetingDuration.

ex:meetingStart
 a time:Instant;
 time:inDateTime ex:meetingStartDescription.

ex:meetingStartDescription
 a time:DateTimeDescription;
 time:dayOfWeek   time:Monday;
 time:day       "08";
 time:hour      "08";
 time:minute    "00";
 time:month     "09";
 time:unitType  time:unitMinute;
 time:year      "2014".

ex:meetingDuration
 a time:DurationDescription;
 time:minutes 90.
```

In case the given LOD dataset contains lot of temporal information, manually publishing all the necessary triplets would be cumbersome. We suggest instead to use a self-unfolding URI for the time entity and an attached template to auto generate the required triplets based on the information in the URI.

In the example above the URI of the interval entity should contain all the data that is needed to generate the corresponding triples. In our implementation the following structure is suggested:

```
<http://example.org/data/Interval;year=2014;month=9;day=8;hour=8;minute=0;durationHour=1;durationMinute=30>
```

### B. Spatial entities

It is often necessary to specify locations for events or files (e.g. photos). Geo-location can be easily captured and represented by coordinates, so encoding latitude and longitude in the location URI is a plausible method. Additional information can be generated by connecting related entities with the coordinates, like the country, city, region, landmark nearby found in a geo database, like GeoNames (www.geonames.org). With SPARQL Insert

[1] http://www.w3.org/TR/owl-time/

queries, such enrichment can be scheduled or triggered by the entry of a new instance of a geo-location.

The enriched geo-location data for a self-unfolding URI could be the following:

```
@prefix geo:
<http://www.w3.org/2003/01/geo/wgs84_pos#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-
syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix gn: < http://sws.geonames.org/> .

<http://example.org/point;lat=47.158775;long=18.
88149>
 a                geo:Point;
 geo:lat          "47.158775";
 geo:long         "18.88149";
 foaf:based_near  gn:3048446>.
```

### C. Measurement representation from Semantic Sensor Networks

A Semantic Sensor Network publishes measurement data in a predefined cadence. Such data could include the location of the measurement, the exact time, the measured value and the sensor ID. Each measurement event is identified by a URI and represented by a set of triplets. In [9] authors suggest models for URI structures to identify and access Stream Data coming from sensor networks. URI schema is proposed containing the sensor identifier, the time of the measurement and the space information. We go further and suggest putting the value of the measurement also in the URI scheme. In this case the URI containing the sensor ID, the time, the space and the measurement information would be sent to the data processing center, where the data enrichment would take place. The data enrichment would use a template containing extra information in RDF triplets about the sensor geographical features, sensor type, measurement interpretations and references. From the information encoded in the URI with the help of the template the raw results of the measurement and also some interpretations and references of the measurement would be generated. This way it becomes simple to query out-of-normal-range events.

The following (highly simplified) example shows the result of the generation of sensor observation RDF data from the URI defining the measurement:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-
syntax-ns#> .
@prefix dul: <http://www.loa-
cnr.it/ontologies/DUL.owl#> .
@prefix ssn:
<http://purl.oclc.org/NET/ssnx/ssn#> .

<http://example.org/observation;instant=2015-01-
03T10-38-
43;lat=60.158775;long=24.88149;value=12.2>
a ssn:Observation;
ssn:observationResult "12.2";
ssn:observationResultTime
<http://example.org/instant;instant=2015-01-
03T10-38-43>;
dul:hasLocation
<http://example.org/point;lat=60.158775;long=24.
88149>.
```

## IV. Generalized Scenarios

In this section generalized Linked Data publishing scenarios are described, where automatic enrichment can play a useful role. We have identified three problem types that are frequently present in Linked Data publishing and where automatic data generation with self-unfolding URI could give solution. These are Blank Node (Anonymous Resources) Candidates, entities with composite keys in URI and reasoning on information encoded in the URI.

### A. Generating Blank Node entities from URIs

In some cases, the URI of a RDF entity is not important or not used, for this reason the concept of Blank Node was introduced. Blank nodes indicate the existence of a thing, without using, or saying anything about the name of that thing. The use of blank nodes is problematic. One drawback is the locality of their scope, since it is not possible to create RDF links to them from external Linked Data sources. Another drawback is the difficulty to merge data from different sources when blank nodes are used, as there is no URI to serve as a common key. Because of this there is a recommendation to avoid the use of blank nodes [12], [13] and to name all resources in a data set using URI references.

Our proposal is to encode the necessary information in the URI of the Blank Node entity thus it becomes a named entity and use automatic triplet generation based on URI patterns and templates. This proposal is useful only if the blank node contains information that can serve as primary key. Otherwise identity generation cannot be avoided.

### B. Modeling composite keys in URI

As URIs identify the entities, it is advisable to encode the primary key information of the entity in its URI. Entities without a single primary key candidate in RDF have their own challenges. Composite keys identify such entities, however defining a good and representative URI with all keys included will result redundancy in the data, since the keys should be encoded in the entities URI syntactically and also as triplets semantically.

Our proposal to resolve this dilemma is to encode the composite key values in the URI of such entities and use automatic triplet generation based on URI patterns and templates. It would provide mechanism to maintain the redundancy of information encoded in the URI and the paraphrased triplets in the same time the consistency of the data would be ensured as well.

### C. Reasoning on information encoded in URI

There are datasets where new triples have to be generated according to specific rules. In [14] Polleres et al. describe the problem when Linked Data sets contain numerical properties and from these numerical properties lot of implicit information could be expressed in the form of simple mathematical equations. For example, expressing simple conversions between different currencies or functional dependencies between multiple properties might be needed. As such equations are not expressible in RDFS or OWL itself, the authors present an approach in [14] to extend the RDFS and OWL languages by attribute equations in order to enable the inclusion of additional numerical knowledge in the reasoning processes. Additionally, SPARQL Insert queries could provide an expressive way to implement information

extraction encoded in URI. Our solution expresses the new relationships determined by the specific rule in the template and generates the new information using the data from the entities URI and the template.

## V. IMPLEMENTATION DETAILS

In this section the functional requirements are described: the URI pattern definition, the corresponding RDF templates and the structure of the SPARQL Insert queries achieving the data generation.

### A. URI Patterns

As the SPARQL language has limited expressiveness, it is important to design the structure of the URIs in a way that supports the processing by SPARQL Inserts and also conforms to other best practices. In [15] there are 8 URI design patterns describing how to assign identifiers within a dataset, where all 8 patterns are widely used and tested in the field. It should be noted that none of them is a good candidate to apply for composite key modeling. From the 8 URI design patterns the closest one to our need is the Hierarchical URI pattern, which can be applied when a natural hierarchy exists within the set of resources (e.g. /books/001/chapters/1). However, in the case of composite key modeling the concept hierarchy does not always exist, i.e. the parts of the key are often in coequal relationship.

Because of this we propose the use of Matrix URI pattern[2] as it can use multiple independent parameters of the entity. A URI following this pattern starts with a base part, followed by the type of the resource and then optionally each part of the composite key:

```
<Base URI>/<type>[;<Property>=<Value>]
```

The example URIs in section 3 are in this proposed structure.

### B. Conceptual RDF Template

The purpose of using templates in the self-unfolding URI process is to express rules, which determine the generation of the new information based on the entity's URI. There is a constant challenge for Linked Data consumers that the noise in the data can lead to fuzzy data structures too. The aim of defining a template for a given type of entity is to support the query consistency. Each template implementation has to define the rules to extract information from the URI and optionally rules to infer any implicit relations or attribute values. The template provides a URI pattern, and prescribes RDF triplets to be generated based on the values extracted from the URI. Following is the high level structure of such a template:

```
<URI> rdf:type  <type with namespace>
       [; <property> <value>|<related URI>] .
```

Related entities can be generated recursively applying the same template. An example for a template generating an OWL Time Instant entity can be the following:

```
<http://example.org/Instant;year={year};month={month};day={day};hour={hour};minute={minute}>
=====>

<http://example.org/Instant;year={year};month={month};day={day};hour={hour};minute={minute}>
 rdf:type time:Instant;
 time:inDateTime.

<http://example.org/DateTimeDescription;year={year};month={month};day={day};hour={hour};minute={minute}> .

<http://example.org/DateTimeDescription;year={year};month={month};day={day};hour={hour};minute={minute}>
 rdf:type time:DateTimeDescription ;
 time:hour "{hour}"^^xsd:nonNegativeInteger ;
 time:minute "{minute}"^^xsd:nonNegativeInteger;
 time:month "{month}"^^xsd:nonNegativeInteger ;
 time:day "{day}"^^xsd:nonNegativeInteger ;
 time:year "{year}"^^xsd:nonNegativeInteger .
```

### C. Implementation with SPARQL Insert statements

SPARQL Insert operation can be utilized to implement templates defined earlier. For each entity having a URI matching the URI pattern a SPARQL Insert statement produces the corresponding properties and objects or attribute values. These update queries are the implementation of the templates describing the entity.

As the proof of the concept we implemented templates for generating time related data in order to represent temporal information of university courses as Linked Open Data. Entities providing temporal description of the data are based on the OWL Time[3] and TimeAggregates Ontologies[4] . The main difficulty we faced was to express the time of recurring events, like lectures on every Monday from 8 am till 9.30 am in the 2015 Fall semester. The precise implementation of such information as Linked Open Data needs the introduction of several additional entities, hence the management of such information is time consuming, error prone and tedious process without automatization.

For the demonstration of our implementation let us consider the following example in Turtle for the description of the time of a specific course as an RDF triple:

```
:c001 oloud:courseTime <http://lod.nik.uni-obuda.hu/data/CourseTime;courseTerm=2015Fall;hour=10;minute=0;durationHour=1;durationMinute=30;dayofweek=2>.
```

The URI of the object part contains all necessary information to generate the RDF triples needed for describing the time of recurring events as Linked Open Data in a proper way. The unfolding of this URI results 6 new LOD entities, each consisting of 3 new RDF triples on average. Due to lack of space and complexity of the SPARQL code and the generated data we omit the presentation of implementation details here, but it is available at [16].

One drawback of this method is the potential inconsistency during the (slight) delay between an entity was created and unfolded or deleted and the remaining orphan related entities were removed. An entity is orphan if there is no triplet where the entity is the object: <?s ?p {Orphan URI}>. To completely remove an entity represented by a URI pattern defined earlier in section 5, all triplets containing the corresponding URI need to be deleted. Altering or removing the original entity from the triplestore needs to be reflected by the related entities too,

[2] http://www.w3.org/DesignIssues/MatrixURIs.html

[3] http://www.w3.org/TR/owl-time/

[4] http://ontology.ihmc.us/temporalAggregates.owl#/

a garbage collection algorithm can look after the related orphan entities. If an auto-generated entity is orphan, it means the parent entity it was generated from no longer exists, so it can be deleted as well as all the sibling entities recursively, as shown in this example:

```
DELETE { ?orphan ?p ?v }
WHERE
 { ?orphan ?p1 ?v .
   OPTIONAL { ?parent ?p2 ?orphan. }
   FILTER (!BOUND(?parent))
   FILTER(STRSTARTS(STR(?orphan),
"http://example.org/point"))
 }
```

## VI. CONCLUSION AND FUTURE WORK

In this paper we present a method for the handling of re-occurring complex inserts in a dataset. The method combines URI patterns with RDF templates to ensure that the full RDF representation of the inserted new entity is automatically generated.

We have demonstrated through scenarios and some example use cases the importance of defining meaningful URIs as well as detailed RDF description for a given entity. By dereferencing these URIs, the results obtained (e.g., in RDF) should also reflect the same information as the information provided in the URI.

We think that the proposed URI structure may have a wider use as a Linked Data pattern [15], since there is no other pattern at the moment to deal with the specific scenarios described in Section 4. Applying the self-unfolding URIs eases the authoring and maintenance of Linked Data especially in cases when simple, frequently occurring concepts (such as date intervals) need to be represented in a rather complex way. The suggested mechanism also provides an elegant way to generate standard URIs for frequent entity types.

Our proof of concept implementation was based on a scheduled operation (e.g. once a day) for the recurring entity generation. While this proved to be a simple solution, there is a period of time when not all entities are unfolded thus leading to data consistency issues. In future work we plan to study consistency maintenance issues. It is advised to maintain meta information about any auto-generated data, in order to keep it in sync with the manually created part. It can be a good practice to maintain these in a separate graph, or in the case of Linked Data Platform (LDP) [2] in the LDP Indirect Container so all data can be viewed with or without the generated information. Future work should include implementing trigger type mechanism to capture changes within a triplestore and to reduce the inconsistent period to a more acceptable one.

Closely tied to the publication of Linked Data is the specification of a standard read/write interface, which is the goal of the Linked Data Platform. Such a platform could provide a natural place to implement the self-unfolding URI mechanism. In LDP a triplet would be sent with POST to an LDP Indirect Container (LDP-IC) and the platform could unfold the entity. In future work the possibility of implementing the self-unfolding URI mechanism in LDP is planned to be examined.

## REFERENCES


[1] T. Berners-Lee: Linked data-design issues, 2006, http://www.w3.org/DesignIssues/LinkedData.html

[2] Speicher, S., Arwe, J., & Malhotra, A. Linked Data Platform 1.0. Working draft, w3c (Mar 2014). http://www.w3.org/TR/2013/WD-ldp-20130730/

[3] Haslhofer, B., Momeni, E., Gay, M., & Simon, R. (2010, September). Augmenting Europeana content with linked data resources. In *Proceedings of the 6th International Conference on Semantic Systems* (p. 40). ACM.

[4] Parundekar, R., Knoblock, C. A., & Ambite, J. L. (2010). Linking and building ontologies of linked data. In *The Semantic Web–ISWC 2010* (pp. 598-614). Springer Berlin Heidelberg.

[5] Euzenat, J., and Shvaiko, P. *Ontology matching*. Springer-Verlag New York, Inc., Secaucus, NJ USA, 2007.

[6] Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *the VLDB Journal*, *10*(4), 334-350.

[7] Haase, P., Mathäß, T., & Ziller, M. (2010, September). An evaluation of approaches to federated query processing over linked data. In *Proceedings of the 6th International Conference on Semantic Systems* (p. 5). ACM.

[8] Hasan, S., Curry, E., Banduk, M., & O'Riain, S. (2011). Toward Situation Awareness for the Semantic Sensor Web: Complex Event Processing with Dynamic Linked Data Enrichment. *SSN*, *839*, 69-81.

[9] Sequeda, J. F., & Corcho, O. (2009). Linked stream data: A position paper.

[10] Corcho, O., & García-Castro, R. (2010). Five challenges for the semantic sensor web. *Semantic Web-Interoperability, Usability, Applicability*, *1*(1-2), 121-125.

[11] Correndo, G., Salvadores, M., Millard, I., & Shadbolt, N. (2010). Linked timelines: Time representation and management in linked data. In *First International Workshop on Consuming Linked Data (COLD 2010), Shanghai, China*.

[12] Mallea, A., Arenas, M., Hogan, A., & Polleres, A. (2011). On blank nodes. In *The Semantic Web–ISWC 2011* (pp. 421-437). Springer Berlin Heidelberg.

[13] Heath, T., Bizer, C.: Linked Data: Evolving theWeb into a Global Data Space, vol. 1. Morgan & Claypool (2011)

[14] Polleres, A., Hogan, A., Delbru, R., & Umbrich, J. (2013). RDFS and OWL reasoning for linked data. In *Reasoning Web. Semantic Technologies for Intelligent Data Access* (pp. 91-149). Springer Berlin Heidelberg.

[15] Dodds, L., & Davis, I. Linked Data patterns-a pattern catalogue for modelling, publishing, and consuming Linked Data (2010). http://patterns. dataincubator.org/book/.

[16] Szász, B., Fleiner, R., & Micsik, A.: Linked Data Enrichment with Self-Unfolding URIs – examples, http://lod.nik.uni-obuda.hu/unfolding/example.html