Linked Data Generation for Courses and Events at Óbuda University

Péter Piros
Óbuda University
Budapest, Hungary
Email: piros.peter@nik.uni-obuda.hu

Rita Fleiner
Óbuda University
Budapest, Hungary
Email: fleiner.rita@nik.uni-obuda.hu

Levente Kovács
Óbuda University
Budapest, Hungary
Email: kovacs.levente@nik.uni-obuda.hu

***Abstract*—Óbuda University wanted to publish public information related to the university in the form of Linked Open Data. Information about subjects, courses, events and teachers are successfully published using the related standards. We had to collect and categorize the sources where information come from; define a way for each category to make these information available; and develop algorithms and softwares to generate the gathered information in form of Linked Open Data. Finally, the automation of the generation process was partly achieved.**

I. Introduction

Tim Berners-Lee stated the fundamentals of Linked Data in 2006 as follows [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.

These four rules ensure that datasets are not only published on the web, but they can be connected to each other. The idea behind interlinking structured data is to make machines able to read them automatically. The 1st and 2nd rule ensures that every piece of data is globally identifiable; and the 4th rule is the answer why we call this form of data "linked".

The standards in the 3rd rule are responsible for the widely understandable content. The Resource Description Framework (RDF) is a data model for expressing information about resources [2]. Its data format is a directed, labeled graph which can be represented by triples: *subject*, *predicate*, *object*. All of these three parts can be URIs; the subject and the object represent the two resources being related and the predicate is the nature of their connection. Once we have a dataset of triples, SPARQL (SPARQL Protocol and RDF Query Language) can be used to query and manipulate it.

Linked Universities project shows [3] that Linked Open Data is present in the education area. As members of this alliance, 9 European universities exposed their public informations as linked data. Information about Courses and Subjects can appear in several languages using Linked Open Data format. This can help foreign students to get familiar with course-related data when studying abroad [4].

The objective of this paper is to demonstrate a possible way to collect, methodize and publish public information related to university courses, subjects and events as Linked Data. In this work, an existing ontology, *Ontology for Linked Open University Data* (OLOUD)[1] was used. According to [5], OLOUD is a practical approach to model higher education related concepts.

In this complex activity, the following partial goals were defined:
- to collect and categorize the sources where information related to university subjects, courses and events come from,
- to define a way for each category to make these information available,
- to process the acquired information so we can work with them in details,
- to generate information in form of Linked Open Data,
- to develop the process to be more automatised.

The rest of the paper is organized as follows. In Section 2 we describe the data model that the generated dataset is based on. In Section 3 the steps of data generation and a special subject, the time handling are presented. Section 4 shows the method of tuning the data generation to be more automated. Section 5 provides some conclusive remarks.

II. Definitions, Platform and Data Model

A. Linked Open Data Platform

As the result of the tasks detailed in this paper, a data set in RDF-format was generated. Despite of the fact that this paper focuses on data generation, a related activity has to be emphasized: choosing a LOD server for importing the dataset into. Because of the high number of such softwares [6], developing a new one was eliminated. In this project, two of them, ARC2 and Apache Marmotta were tested and the latter was selected. It has bigger community and important additional functions too, such as User management, Automatic Importing feature and SPARQL Webservice.

B. Definitions and Data Model

Several ontologies exist that describe education-related datasets, but they are quite different in the meaning of terms like course, subject and study programme. The OLOUD ontology is built up according to the Hungarian Higher Educational term usage, it aims to integrate data from several sources

[1] http://lod.nik.uni-obuda.hu/oloud/oloud-20160609.owl

978-1-5090-5655-2/17/$31.00 ©2017 IEEE

and provide personal timetables, course-, subject-, curriculum information and other types of help for students and lecturers. University curriculums can be harmonized with each other by using ontological perspectives [7].

To be understandable later, the following concepts are defined: each student is assigned to a *Curriculum* which consists of a list of Subjects. *Subject* is the basic element of the education, ie. students have to accomplish a list of subjects in their educational years. *Courses* are basic elements of the Subjects, they are concrete lessons with a teacher in specific classrooms. Every Subject has at least one Course. An ontology is defined by determining its classes, properties, individuals, class structure and the meaning of its concepts.

The formal description of the OLOUD ontology is written in *OWL*, uses the prefix *oloud* and the namespace *http://lod.nik.uni-obuda.hu/oloud/oloud#*. The most crucial concepts in the ontology are Curriculums, Subjects and Courses depicted in Figure 1, 2 and 3.

III. Steps of Data Generation

A. Sources

First, we had to collect and categorize the sources where information come from.

The main source was a relational database structure located in the university student information system called *Neptun*, at the Óbuda University. This SQL-styled content could be converted into spreadsheet format in the same system. The resulted file contained all available information about subjects and courses.

The second source was a part of the official web page of the University[2]. The page titled with "Current Events" lists the upcoming and former events related to the university's informatics faculty. This HTML-based content provided the second basis to reach public information for the data source.

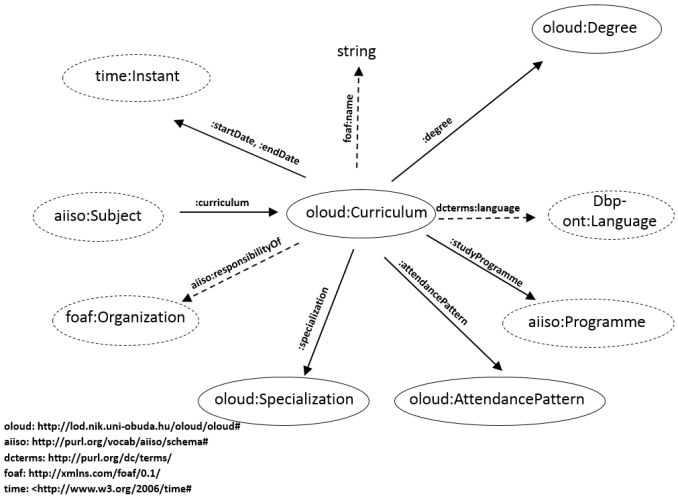

Fig. 1. A subdiagram of the OLOUD ontology regarding the Curriculum concept

[2]http://www.nik.uni-obuda.hu/hu/esemenyek

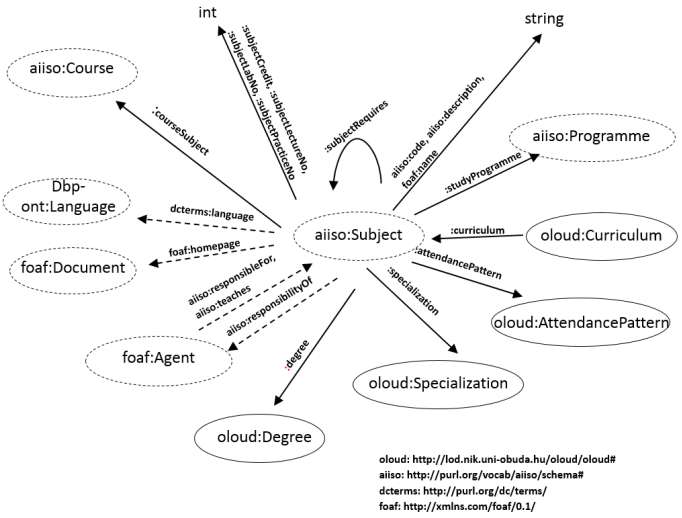

Fig. 2. A subdiagram of the OLOUD ontology regarding the Subject concept

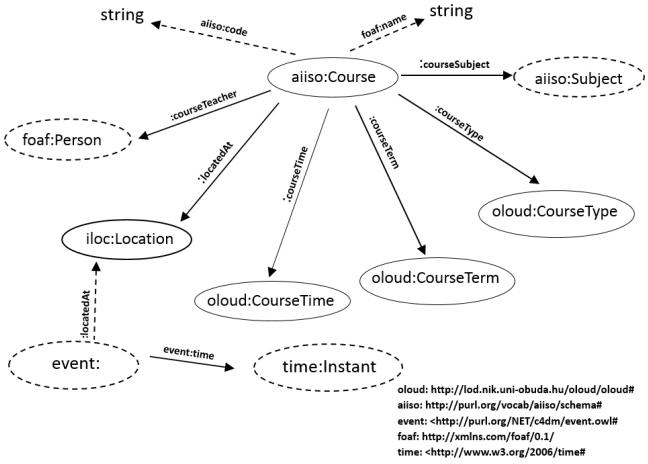

Fig. 3. A subdiagram of the OLOUD ontology regarding the Course concept

B. Subjects

The spreadsheet file generated from SQL contained the following parts of information about each subject: subject code and name, credit number, exam type, prerequisite subjects, URL address of the subject, description, responsible person, and organization where the subject belongs to.

As one of the most important basic of all databases, we had to determine a unique value that identifies every single subject. Resulted from the university workflow, the first field (subject code) in the previous list meets this criteria.

Because of the activities and the tasks detailed in *Section 2* were fully conducted, the expected RDF-format of the subjects became determined. Thus, the next step was developing a program that transforms the data into the expected format. Many programming languages would be suitable for this purpose; PHP was chosen for its simply structure and short footprint.

The algorithm of the script follows the next steps. The same process was used for data generation in case of subjects, courses and teachers.

1) All information was mapped into a two-dimensional array to simplify the reference to each piece of data
2) A template was created manually based on the final version of OLOUD ontology. Predicates of the RDF-triples gave the skeleton of the template, and they were determined by the OLOUD ontology. In this case they are the properties belonging to the aiiso:Subject class. In the template, the RDF objects and RDF subjects were represented by variable names (e.g. *SUBJECT_RESPONSIBLE*).
3) The program looped through all items of the two-dimensional array and stored all class entity identifiers in a *key and value* structure. This step is essential in order to make every subject capable to be referred *(see the complex case point in the next loop)*.
4) This step is responsible for the LOD data generation according to the template. The program looped through all items of the two-dimensional array again.
    a) in simple cases: after some smaller formatting, the variable of the template was replaced by a field from the array.
    b) in complex cases: more processing were needed to get the right value. For instance, the prerequisite subject is not a simple text-based field, but refers to another subject and can have multiple values. Another good example is the subject code field. Because of the used coding scheme in the Neptun student administration system, additional fields can be extracted from it, like the language, the attendance pattern, the specialisation and the degree of the subject.

The following code represents the template used for generating RDF-triples for the university subjects:

```
### http://lod.nik.uni-obuda.hu/data/
{SUBJECT_CODE}
:{SUBJECT_CODE} rdf:type aiiso:Subject ,
owl:NamedIndividual ;
oloud:subjectCredit "
{SUBJECT_CREDIT}"^^xsd:Integer;
aiiso:description
"{SUBJECT_DESCRIPTION}"@hu ;
foaf:homepage "{SUBJECT_HOMEPAGE}" ;
aiiso:code "{SUBJECT_CODE}" ;
foaf:name "{SUBJECT_NAME}"@hu ;
aiiso:responsibilityOf {SUBJECT_RESP};
oloud:studyProgramme
{SUBJECT_STUDYPROGRAMME} ;
aiiso:responsibilityOf {SUBJECT_RESP_2};
oloud:attendancePattern
{SUBJECT_ATTENDANCEPATTERN} ;
dcterms:language {SUBJECT_LANGUAGE} ;
oloud:specialization
{SUBJECT_SPECIALIZATION} ;
oloud:subjectRequires {SUBJECT_REQUIRES}
oloud:degree {SUBJECT_DEGREE} .
```

The following code describes an aiiso:Subject instance in Turtle format generated by the previous template.

```
### http://lod.nik.uni-obuda.hu/
data/NAIIK1SAED
:NAIIK1SAED rdf:type aiiso:Subject ,
owl:NamedIndividual ;
oloud:subjectCredit "2.00"^^xsd:Integer;
aiiso:description "Irodalomkutatás. ...
 Eredmények publikálása és prezentálása."@hu ;
foaf:homepage
"http://users.nik.uni-obuda.hu/to/
tantargy/infokommunikacios-technikak";
aiiso:code "NAIIK1SAED" ;
foaf:name "Infokommunikációs technikák"@hu ;
aiiso:responsibilityOf :AII;
oloud:studyProgramme
oloud:ComputerEngineer ;
aiiso:responsibilityOf :László_Nádai;
oloud:attendancePattern oloud:PartTime ;
dcterms:language
<http://dbpedia.org/resource/
Hungarian_language> ;
oloud:specialization oloud:ObligatoryPart ;
oloud:subjectRequires
"Matematika Szigorlat"@hu,
"Szakmai Szigorlat"@hu;
oloud:degree oloud:BSc  .
```

The full source of the script is available for public in a GitHub repository [8].

C. Courses

Turning university course data into RDF-based content was a complex task because items of course information contained more connection to other data instances. The spreadsheet file generated from SQL contained the following information about each courses: course code, course type, subject code, subject name, timetable information, room, teacher, language, semester, exam type.

The method is the same and the algorithm is similar to the one used in the case of subjects. Some fields which needed special processing are the following.

1) Classroom: identifying rooms was a preliminary task and it was based on the iLOC[3] ontology, designed for indoor description and navigation purposes [9], [10].
2) Teachers: they can be connected with more courses and each teacher must be an RDF-subject as well.
3) Timetable information: its handling is described in more detail in subsection *E*.

In case of classrooms and teachers, the following method was chosen:

[3]http://lod.nik.uni-obuda.hu/iloc/iloc-20160409.owl

1) the program collected all classroom and teacher names,
2) the template-based algorithm was used to generate classroom and teacher entities,
3) the generated entities were stored in a key-value store,
4) when the main algorithm ran into a place of a classroom or teacher, its identification value was retrieved from key-value store and it was used.

As an example, the following code describes an aiiso:Course instance in Turtle format:

```
### http://lod.nik.uni-obuda.hu/
data/IK1_EA_E_2014-15-1
:IK1_EA_E_2014-15-1 rdf:type aiiso:Course ,
owl:NamedIndividual ;
aiiso:code "IK1_EA_E"@hu ;
foaf:name "Infokommunikációs technikák"@hu ;
oloud:courseSubject :NAIIK1SAED ;
oloud:locatedAt :F07 ;
oloud:courseTeacher :Valeria_Poser ;
oloud:courseTime
<http://lod.nik.uni-obuda.hu/
data/CourseTime;courseTerm=2014Fall;
hour=16;minute=15;durationHour=1;
durationMinute=35;dayofweek=2> ;
oloud:courseType oloud:Lecture  ;
oloud:courseTerm :2014Fall .
```

The full source of the script is available for public in a GitHub repository [8].

D. University events

The second source was a part of the official web page of the University. This page lists the upcoming and former events related to the university's informatics faculty. It is displayed with the Word Wide Web's standard markup language, HTML. This well-structured and standardised content ensured that the list of university events can be reached, analysed and processed.

To generate LOD-based entities from HTML source, a new program had to be developed. According to [11], crawler is a program that automatically collects web pages to create a local index and/or a local collection of web pages. In this meaning, a crawler software had to be made to reach event list and then the specific items. Then, a screen scraper software analysed the content and extracted the information needed.

The following process was used to solve the above task:

1) analysing the full HTML source code of the given URL where the list of events is available,
2) finding recurrences to recognise particular events. This conception was based on the fact that in a well-structured HTML page the repeating elements are declared with the same HTML-tags (called *DOMSs*) and have the same style classes (*CSS classes*),
3) constructing rules to extract the title and the URL of each event,
4) based on the previous steps developing the software to get all available information about each university event.

Table 1 shows a concrete example about the route in the HTML-structure from the highest (*body*) tag into the title of a university event (a clickable *a* tag). The first column is the level of the current HTML-element in the whole HTML hierarchy, the second is the HTML-element itself, while the third shows the CSS class of the specific element.

Hierarchy	HTML-element (DOM)	CSS Class
#1	body	*(none)*
#2	div	page
#n	...	..
#n+1	div	view-content
#n+2	div	views-rows
#n+3	div	view-field-title
#n+4	h2	field-content
#n+5	a	*(none)*

As you can see in the table, the information needed to be extracted is at the n+5 hierarchy level represented by the *a* tag. After getting the information needed, we followed the template-based algorithm just like in the case of subjects, courses and teachers. The following example shows a university event description in Turtle format:

```
:E-tdk-2016-tavaszi-felev rdf:type
oloud:Event ,
owl:NamedIndividual ;
rdfs:label "TDK 2016. tavaszi felev"@hu ;
rdfs:comment "Description..."@hu ;
oloud:locatedAt :OE_Main_Building ;
event:time <http://lod.nik.uni-obuda.hu/
data/UnitDayInstant;year=2016;
month=04;day=20> ;
rdfs:seeAlso <http://www.nik.uni-obuda.hu/hu/
esemenyek/2016-04-20/
tdk-2016-tavaszi-felev> .
```

E. Handling of Time Data

While working with courses, the biggest challenge was to describe time information in a unified, solid format. A way to declare time information which has a starting moment, a duration and a recurrence had to be determined.

In earlier versions, we declared 4 RDF pattern for each time data (starting moment, duration, repeating, and one which synthesizes the formers). This method works, but according to [12], another solution was used in the process of generation: only a special URI (so-called Self-unfolding URI) was created and inserted into the RDF triples. The URI has been taken apart later with the help of a template and some SPARQL CONSTRUCT queries.

A template for Self-Unfolding URI is as follows:

```
http://example.org/data/CourseTime;
courseTerm={term};hour={hour};
minute={minute};
durationHour={durationHour};
durationMinute={durationMinute};
```

```
dayofweek={dayOfWeek}
```

In [12] the list of SPARQL CONSTRUCT queries can be found[4], that are needed to run in order to unfold these time-related URIs.

IV. Automation of Data Generation

In the previous section the crawling and screen scraping software was outlined. In the following, we present the automation of data generation related to university subjects, courses and events.

A. Automation of Generating University Events

By its very nature the crawler software is also capable of automating its functioning. Every time it runs, new university events are generated into RDF-styled format. To achieve full automation, two questions had to be solved: how to call the script continuously without manual activity; and how the generated RDF triples get into the RDF platform.

1) Scheduling: Under Linux operating system, *crontab* command is a widely used solution to run scheduled, repeated tasks. Under Windows operating system, a similar command, *Schtasks* and (in older versions) *Windows Task Scheduler* can be used [13] for the same purpose.

2) Importing: Two tested and working methods are available to pipeline the generated triples into the RDF store. The first: Marmotta provides a REST-based Webservice which other softwares can use to send and receive information from/to Marmotta. One of the options is a POST request (named *import/upload*) which we used to send content into the system. The other available option is to use Marmotta's import folder. When a supported file is copied into this special folder, Marmotta imports it within a few minutes.

B. Automation of Subjects and Courses

In the case of Subjects and Courses, a continuously refreshed data source was not available. For this reason, achieving fully automation was not possible; the first step has to be made by a human - this is because of the nature of the source (see at *Sources*). This first step is when a human uploads the spreadsheet file into the system. To support this process, we developed another PHP-based software which has 4 steps. In Step 1, the user selects the content type; in Step 2, user uploads the file; in Step 3 the columns can be identified; and in Step 4, the generated file can be downloaded after checking a preview about it.

In the last step, the user can also send the generated triples into the RDF storage (the method is the same as in the case of the university events).

V. Conclusion

In this paper we presented a case study on generation of public university information in the form of Linked Open Data. In this work, we started with the list of information sources and finalized with the automation of the data generation. Because of the problem, that a continuously refreshed data source was not available for subjects and courses, achieving fully automation was not possible. In the case of university events, a full automation has been achieved using PHP-based scripts and built-in operation system services. The new algorithms and softwares use OLOUD ontology and Apache Marmotta as Linked Open Data platform.

[4]http://lod.nik.uni-obuda.hu/unfolding/example.html

References

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

[2] World Wide Web Consortium. (2014). *RDF 1.1 Primer.* http://www.w3.org/TR/rdf11-primer

[3] Linked Universities, http://linkeduniversities.org

[4] Pham, X. H., Jung, J. J., Nguyen, N. T., & Kim, P. (2016). Ontology-based Multilingual Search in Recommendation Systems. Acta Polytechnica Hungarica, 13(2) (pp. 195-207).

[5] Szász, B., Fleiner, R. & Micsik, A. (2016). A case study on Linked Data for University Courses. In: '16: 5th International Workshop on Methods, Evaluation, Tools and Applications for the Creation and Consumption of Structured Data for the e-Society, 24 Oct 2016, Rhodes, Greece.

[6] W3C: Sparql Implementations http://www.w3.org/wiki/SparqlImplementations

[7] Mandić, M., Konjović, Z., & Ivanović, M. (2016). Platform for Computer-aided Harmonization of Informatics Curricula. Acta Polytechnica Hungarica, 13(3) (pp. 159-179).

[8] GitHub Repository (user: pp887), https://github.com/pp887/php-linked-data-oe

[9] Fleiner, R., Szász, B., & Piros, P. (2016, May). Indoor navigation Linked Data at Óbuda University. In Applied Computational Intelligence and Informatics (SACI), 2016 IEEE 11th International Symposium on (pp. 25-30). IEEE.

[10] Szász, B., Fleiner, R. & Micsik, A. (2016). iLOC - Building In-Door Navigation Services using Linked Data. In: Michael Martin, Martí Cuquet, Erwin Folmer Posters&Demos@SEMANTiCS 2016 and SuCCESS'16 Workshop. Conference place, time: Leipzig, Germany, 2016.09.12-2016.09.15. CEUR-WS.org, Paper http://ceur-ws.org/Vol-1695/paper30.pdf. 4 p.

[11] Cho, J., & Garcia-Molina, H. (1999). The evolution of the web and implications for an incremental crawler. Technical Report. Stanford. http://ilpubs.stanford.edu:8090/376/1/1999-22.pdf

[12] Szász, B., Fleiner, R. & Micsik, A. (2016). Linked Data Enrichment with Self-Unfolding URIs. In: SAMI 2016: IEEE 14th International Symposium on Applied Machine Intelligence and Informatics, 2016.01.21-2016.01.23, Herlany, Slovakia. New York: IEEE, 2016. pp. 305-309.

[13] Windows Technet: Command-Line Reference https://technet.microsoft.com/en-us/library/cc725744.aspx