Journal of Telecommunications and the Digital Economy

Addressing digital transformation in universities

how to effectively govern, trust and value Institutional data

Vincenzo Maltese

Dissemination and Evaluation of Research Results Division,
University of Trento, Italy (vincenzo.maltese@unitn.it)

Abstract: In facing digital transformation challenges, universities need to set up their data governance strategies. They include effective solutions to trace and value data about key assets (such as researchers, publications, courses, research projects) scattered across multiple legacy IT systems. As part of an overall solution to deal with the unavoidable data fragmentation and diversity, we provide the complete code of a simple and very efficient framework that can be employed by universities to develop their own knowledge graph, offering a comprehensive picture of the strategic data of the university, such that it can be consistently exploited by different digital services.

Keywords: Digital Transformation, Data Integration, Knowledge graphs, Vocabularies.

1. Introduction

In pursuing their missions along the three pillars of education, research and societal impact, universities need to find their own way to address the challenges increasingly posed by digital transformation. The term digital transformation is often used to indicate a set of technological, cultural, organizational, social, creative and managerial changes (McDonald et al., 2012). Digital transformation goes beyond the simple adoption of new technologies and makes it possible to provide services, supply goods, live experiences, find, process and make accessible large amounts of content regardless of the real availability of resources (human, intellectual, economic, etc.), pervasively creating new connections between people, places and things.

Digital transformation in higher education institutions is about the development of new more advanced and effective methods and practices in pursuit of the higher education's mission (Alenezi, 2021). Even though it brings new opportunities, digital transformation also poses new challenges for Communication and IT departments of universities (Maltese, 2018a). Recent studies (Safiullin & Akhmetshin, 2019) (Gafurov et al., 2020) (Marks & Al-Ali, 2022) confirm that universities are not yet prepared, in terms of vision, competency, infrastructures, data strategies and digitalization of their services.

This is a pre-print version of the paper that can be found at http://doi.org/10.18080/jtde.v12n1.808
Journal of Telecommunications and the Digital Economy, ISSN 2203-1693, Volume 12 Number 1

March 2014

Our work focuses on digital information challenges faced by universities. Universities need to provide to their users detailed information about a variety of key assets, such as professors, researchers, employees, publications, patents, courses, and research projects. It is however difficult for universities to present a complete, up-to-date and coherent picture about them across the different digital communication channels and services employed. For example, it may happen that a certain person is an associate professor according to the human resources system (the main authority for such data), a research fellow on the main institutional portal (the portal is outdated), and a post-doc researcher on the department website (the website is not only outdated, but it uses different terminology with respect to the institutional portal).

The root of this difficulty stands in the inherent complexity of the IT university ecosystem (Maltese, 2018b) and it is common to many other large-scale organizations (Gartner, 2014). On the one hand, the diversity of IT systems is needed to targets specific business processes and key assets with confined responsibility. On the other hand, the data fragmentation and diversity - that progressively increase with the number of IT systems employed and the growth of data - bring to a sort of entropic effect where: data about the key assets is scattered across multiple information silos; data differs in format, metadata, conventions and terminology used; data gets duplicated; discrepancies and conflicts increase because different versions and descriptions of the same assets coexist.

Solutions to this problem can be altogether referred to as *data governance strategies*. We report our experience matured during research (Giunchiglia et al., 2012b) (Giunchiglia et al., 2014) (Maltese & Giunchiglia, 2016) (Maltese & Giunchiglia, 2017) and innovation (Maltese, 2018b) (Giunchiglia et al., 2022) projects conducted in universities and provide further insights that have been presented during a series of invited talks (Maltese, 2017) (Maltese, 2018a) (Maltese, 2023a) (Maltese, 2023b) (Maltese, 2023c).

(Maltese & Giunchiglia, 2017) proposed a general solution to address data fragmentation and diversity in universities. By adapting and extending the notion of digital library, they introduced the notion of *digital university*, defined as *"a set of key resources, methodologies and tools appropriately organized to effectively support universities' users"*. The solution stands in (a) addressing data diversity via the adoption of well-established Library & Information Science methodologies and tools to curate data and metadata quality and (b) addressing data fragmentation via the adoption of data integration methodologies and tools.

(Maltese, 2018b) provides the description of the system architecture, the tools and the digital services that were developed at the University of Trento in Italy in the context of the Digital University initiative (http://www.unitn.it/en/DU/info) and that constitutes the first implementation of the proposed general solution. The infrastructure follows the Hub-and-

Spoke paradigm. The Hub is an IT system that collects data extracted from various information silos and encodes it as a knowledge graph. This is achieved by means of Extract, Transform and Load (ETL) facilities that run once a day. With the Extract phase data is selected from relevant legacy IT systems. With the Transform phase, data diversity is addressed by codifying data uniformly in schema and terminology, and by consistently assigning a unique identifier to data about the same entity initially scattered across different information silos. With the Load phase, data fragmentation is addressed by collecting and pulling together into the Hub data about the same entity (for instance, a person, a publication, a research project). The knowledge graph provides centralized access to a number of spokes, each of them being a new IT system expressly developed to support a different digital service. In our previous work, we described the organizational, technical, conceptual, legal, security, user-related challenges that typically arise (Maltese & Giunchiglia, 2016) and how we actually addressed them in Italy and in Mongolia (Giunchiglia et al., 2022). Similar challenges, barriers and main factors that can influence the success of digital transformation solutions in higher education institutions have been discussed by (Rodríguez & Bribiesca, 2021) (Tungpantong et al.,2021) (Esmailzadeh et al., 2022) (Gkrimpizi & Peristeras, 2022) (Sułkowski, 2023).

The main contribution of this paper is the description and the complete source code of a new data integration framework that we developed in 2021, and that is now publicly available on GitHub: https://github.com/vinmal74/DU. It entirely substitutes the one employed in the first version of the Digital University system architecture developed in between 2017 and 2018. The framework presented in this paper supports data engineers in the creation of a multilingual knowledge graph that is built from data extracted from multiple sources. In particular, the new framework has been entirely developed in Java (the previous one required several different technologies, including Java, Scala and Coffee scripts) and makes much simpler the development of the ETL facilities. By changing the entity matching algorithm (that is necessary to detect and merge duplicates) and the data structures employed, it allowed us to overcome the technical challenges described in (Giunchiglia et al., 2022), thus reducing the time needed to create the knowledge graph by three orders of magnitude (it is around 1000 times faster) w.r.t. the first version of the framework. In terms of complexity the new algorithm is linear in the number of entities to be integrated, while the previous one was quadratic in the number of entities to be integrated, i.e. the new algorithm is O(n) while the previous one was $O(n^2)$. The new framework is faster also because of the data structures employed (hash tables require constant time for entity search) that are stored entirely on RAM memory (while the previous framework operated entirely on databases stored in the file system).

In the rest of the paper, we briefly summarize the state of the art (Section 2) and recall the system architecture (Section 3) and the methodology (Section 4) employed, already illustrated in our previous work. We then continue with the main contribution of this paper, that is the complete source code (provided in Appendix A) of the new framework we developed to support the creation of the multilingual knowledge graph at the core of the Digital University solution (described in Section 5). Our aim is to provide the methodology and tools such that other universities can replicate our work. At this purpose, we also provide a demonstrative example of data sources (Appendix B) that may need to be integrated (Section 6) and the ETL code (Appendix C) necessary to create the corresponding knowledge graph (Section 7). We also illustrate how the knowledge graph can be consistently used by multiple digital services (Section 8). Finally, in Section 9 we summarize the work done and the future work.

2. State of the art and related work

Several research communities traditionally address data fragmentation and diversity (Maltese et al., 2009). In the following, we focus on the solutions proposed by Business Intelligence (BI) and Library & Information Science (LIS).

The primary purpose of BI is to support decision-making in organizations (Buchanan & O'Connell, 2006). Data-driven decision-making refers to the practice of basing decisions on the analysis of data rather than purely on intuition (Brynjolfsson et al., 2011). Therefore, data needs to be appropriately collected and prepared. To this end, data integration is a fundamental technique in BI to tackle the initial data fragmentation and diversity. In fact, data integration is a process that combines data from different sources and provides users with a uniform view of data (Lenzerini, 2002). Two main alternative approaches exist. In federated systems, data is logically combined at query time. In centralized systems, data is physically combined in a data warehouse via ETL procedures. The Extract phase deals with the selection, assemblage, analysis and processing of data. The Transform phase takes care of converting data into a standard format. The Load phase imports data into the data warehouse. The centralized approach ensures there is one trusted proxy providing data in a timely manner and uniformly. Data warehousing is a fundamental tool of BI, and metadata plays a key role because of the complexity of the data migration process. Therefore, data warehouse teams and business users must understand myriad characteristics of data to manipulate and use it effectively (Watson & Wixom, 2007).

Library Science is traditionally concerned with archiving texts and organizing storage and retrieval systems to give efficient access to texts (Denning, 2003). LIS is the technical and technological innovation of Library Science that employs information technology for documentation and library services (Buckland, 1996). Libraries have a strong tradition in data and metadata curation, especially in terms of standard data models for the representation of intellectual and artistic creations (O'Neill, 2011). Metadata about them includes title, subject, and authors. Authority control makes sure that each entity is assigned a unique header such that each entity can be uniquely identified and referred to (O'Neill, 2011). Unique headers include names and alphanumeric identifiers. Similarly, vocabulary control enforces the usage of standard terms to unambiguously refer to each subject (Zeng et al., 2011). In controlled vocabularies, standard terms are arranged hierarchically from broader to narrower terms (ISO 2596-1:2011). For instance, in biology we may establish that the standard term to denote "any malignant growth or tumour caused by abnormal and uncontrolled cell division" is cancer, that cancer is a disease (broader term) and that melanoma is a type of cancer (narrower term). Thus, a user searching for cancer can be directed also to texts about melanoma. Altogether, the adoption of these practices allows controlling diversity and obtaining high quality data that in turn ensures high precision and recall in search. Data fragmentation is addressed in libraries by employing standard data exchange protocols, such as the OAI-PMH framework (Sompel et al., 2004) and by adopting solutions to map equivalent concepts in different knowledge organization systems (ISO 2596-1:2011) (Giunchiglia et al., 2009) (Maltese et al., 2010) (Giunchiglia et al., 2012a).

A few initiatives provided solutions to support storing, searching, browsing, visualizing and sharing scholarly data. VIVO (Börner et al., 2012) relies on Semantic Web technologies to represent and store data in the RDF standard model (https://www.w3.org/RDF/) and retrieve it using the SPARQL query language (https://www.w3.org/TR/rdf-sparql-query/). However, it has been observed that these initiatives offer limited support to tackle data diversity and data fragmentation (Maltese & Giunchiglia, 2017). In fact, even though URIs play the role of unique headers, nothing prevents the usage of different URIs for the same entity across datasets. Duplicates are handled at importing time by discovering and linking them automatically. The discovery of duplicates can be achieved for instance by means of String similarity. The linking is typically done by defining the owl:sameAs relations between entities, i.e. the property that by linking two individuals specifies that they are actually the same. This means that duplicates remain unmerged. As a result, queries may return multiple equivalent entities that need to be reconciled before exploiting and visualizing the results. These approaches are limited in that they only focus on data representation and do not seem to provide any facility or suggest any methodology to effectively control and enforce terminology.

Our approach is compliant with other solutions designed for universities, such as VIVO, or more in general for digital libraries, such as DSPACE (Smith et al., 2003). For instance, a converter can be easily developed to translate our knowledge graph in to the VIVO model and ontology such that it can be exploited by VIVO applications, such as the VIVO portal. The main contribution of our work is given by the framework that makes the creation of the knowledge graph simple and very efficient.

3. The system architecture

The system architecture that we set up in Trento (Figure 1) was first introduced in (Maltese, 2018b) and described further in (Giunchiglia et al., 2022). The knowledge graph is built, by reusing the data that progressively become available through ETL facilities, and then it is employed as a Hub in an incrementally built Hub-and-Spoke architecture. Each spoke supports a different digital service. The idea is that new spokes are added incrementally whenever there is a need for a new service which cannot be provided by the existing spokes.

This architecture was chosen as it represents a more efficient and scalable alternative to point-to-point communication in that the number of connectors between IT systems is reduced drastically, thus reducing complexity and maintenance costs (Hopkins et al., 2015).

In our architecture, the Hub collects data extracted from various data sources (Extract), encodes data according to a uniform model and terminology (Translate) and creates a knowledge graph through an integration framework (Load). Through dedicated Application Programming Interfaces (APIs), a number of Spokes get access to the knowledge graph stored in the Hub.

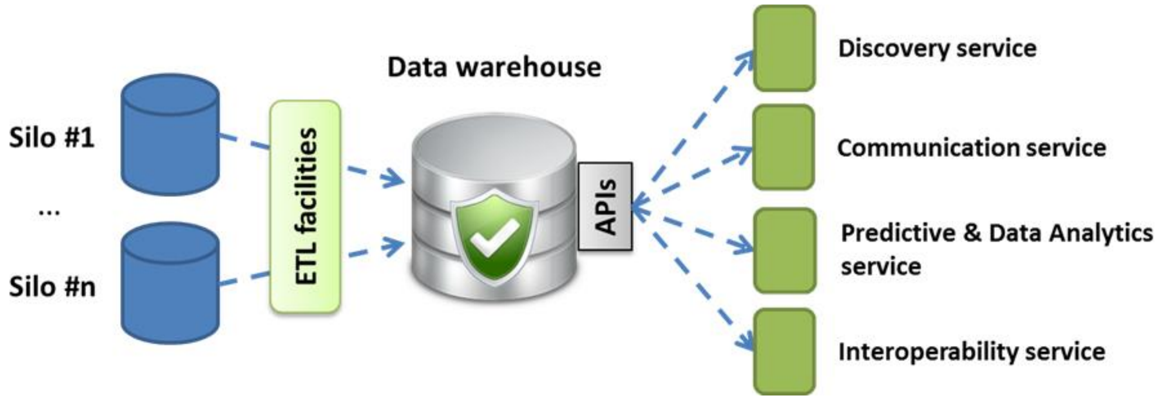

Figure 1 - The system infrastructure of Digital Universities

Overall, the Hub fulfils the following requirements (Maltese, 2018b).

The Hub provides centralized access to data that are natively stored in the heterogeneous data sources (different schema, model, and format) managed by legacy IT systems. This separation of duties is necessary to ensure that legacy systems can continue to function as usual, thus benefitting from all the advantages that come from their vertical end-user applications. Advantages include contained costs, dedicated business processes, focused data, dedicated users and confined responsibilities. Relevant data about all the key entities that are necessary to support the centralized services is reused in the Hub by means of ETL facilities. They ensure that data about the same entity extracted from multiple sources is appropriately collected, transformed, merged and correlated. In particular, entity matching (e.g. (Wang et al., 2011)) and merge facilities are essential to avoid the presence of duplicates.

The Hub supports knowledge and language localization, in that the knowledge graph is built according to a local customized data model and terminology, tailored to be functional to the local digital services. We suggest that localization can take place starting from a reference data model (the knowledge) and vocabulary (the language) designed specifically for universities (Maltese, 2018b). Their main purpose is to provide a common core of entity types, properties and terminology in multiple languages necessary to fulfil typical services of a university and to favour interoperability among universities, similarly to what is done by VIVO. At the same time, the different institutional needs of universities across the globe demand for the capability of the system to support their customization and extension as required locally by the services of a certain university.

The Hub supports the development of centralized services via dedicated APIs that provide access to the knowledge graph. APIs support the development of university services on the spokes such that they can consistently query the Hub and exploit the same content (i.e. the knowledge graph). Centralized services can be developed incrementally and include: (a) *Discovery services* supporting browsing and search, through which users can issue expressive queries seeking any entity based on their properties (Giunchiglia et al., 2014); (b) *Communication services* conveying information to university stakeholders (Maltese & Giunchiglia, 2017), uniformly and consistently across different institutional information channels (Maltese, 2018b); (c) *Predictive & data analytics services* supporting decision-making processes (Waller & Fawcett, 2013), thus allowing the governance to explore and discover correlations between data as well as to identify areas that require intervention for the enhancement of organizational efficiency (Brdesee, 2021); (d) *Interoperability services* supporting the import/export of data from/to existing standards - such as the publication of institutional Open Data (Tran & Scholtes, 2015), or according to the VIVO model and ontology - or to answer queries across multiple federated universities.

Among other things, in our previous work (Maltese, 2018b) we described how we comply with Intellectual Property Rights (IPR), licensing and privacy concerns and guarantee secure access to data.

In terms of IT security, we selected technologies by making sure that they satisfy security levels demanded by Italian law. Our IT staff constantly ensures that adequate security measures are in place. Data sources and system components are secured and not accessible from outside of the University intranet. Access to them is granted to administrators only. Data is accessed exclusively via database views expressly arranged to provide access to relevant data only.

Among other things, this makes system maintenance easier in that such database views can be seen as *contracts* that cannot be violated even in case the data source changes, e.g. because of an update of the corresponding IT system. Regular backups guarantee for the data integrity.

To protect the privacy of users, and to be compliant with the General Data Protection Regulation (GDPR), in designing and developing the system and the services we followed well-established privacy-by-design principles, suggested also by the European Data Protection Supervisor (EDPS). For instance, our privacy policies are publicly available, only relevant and non-sensitive data is managed, data is stored in separate indexes in different Spokes in order to prevent unwanted correlations. Each Spoke receives only the data that is strictly relevant for the digital service it supports. In terms of IPR and licensing, we promote and support Open Science principles by allowing the download of scientific publications of our researchers with Creative Commons licenses through the institutional portal we developed.

4. The methodology

The methodology, introduced in (Maltese & Giunchiglia, 2017) and refined in (Giunchiglia et al., 2022), defines an iterative process composed of sequential steps, briefly illustrated below, which are followed every time a new digital service needs to be designed and developed. In our previous work we illustrated the advantages of this methodology that include scalability, cost-effectiveness, and facilitated compliance with legal constraints.

Step 1. Collecting service requirements. It consists of collecting the requirements of the new service in terms of functionalities, target users and necessary data.

Step 2. Knowledge localization. The reference data model, providing the schema which is enforced to store the data in the Hub in the form of a knowledge graph, is adapted to local needs. It is constituted by entity types and properties necessary to describe typical key entities of universities such as people, courses, publications, dissertations and research projects. (Chatterjee et al., 2016) presents a methodology that can be followed to design the reference data model in a domain based on a set of user queries. The model should include identifiers, i.e. those properties necessary to identify univocally an entity of a certain type such that entity matchers can work properly (Bouquet et al, 2007). Knowledge adaptation means adding or specializing entity types and properties that are necessary to support the new service.

Step 3. Language localization. The controlled vocabulary, providing the terms in multiple languages needed to express data according to the data model, is adapted to local needs. As previously described in (Maltese, 2018b), we employ well-established Library and Information Science (LIS) methodologies for vocabulary development. For instance, the vocabulary should provide the terminology necessary to describe the various positions occupied by people (i.e.

full professor, associate professor, researcher), the various kinds of publications (i.e. journal article, conference paper), the statuses of a research project (i.e. submitted, approved). Language adaptation means adding or specializing concepts, selecting preferred terms from the vocabularies or adding new languages that are necessary to support the new service. A non-trivial issue to be solved in this step is that of handling lexical gaps (Giunchiglia et al., 2018), i.e. concepts which do not have a precise translation in the target language. This fact happens quite frequently because of the different geo-political context and local organizations of universities in the world. In our work we basically addressed language diversity by representing knowledge as language independent concepts whose meaning is approximated in each language by means of terms that are the closest in meaning.

Step 4. Data hunting. The legacy IT systems are assessed in order to identify the possible sources for the data required by the service. The following cases can arise: (a) there is only one system that can provide the necessary data; (b) multiple systems, possibly maintained by different academic or administrative departments, can provide part of the necessary data, which can eventually partially overlap or even be in conflict; or (c) existing systems cannot provide all the necessary data. In the latter case, it is necessary to develop new IT systems able to complement the missing data.

Step 5. Building the knowledge graph. ETL facilities are implemented in order to Extract and Translate data according to the localized knowledge and language, and to Load them into the Hub in form of a knowledge graph. Mechanisms to resolve conflicts in data may include authority (based on the ordering of importance of the sources) or voting (based on the majority of the sources) schemes (Dong & Naumann, 2009). Overlaps are handled through entity matching and merging techniques. This task requires an adequate infrastructure able to semi-automate the process and to keep the Hub aligned with the sources, by running ETL facilities regularly (e.g. once a day). An example of a case in which human intervention is required is to fix mistakes in the data (whenever possible, once discovered they should be fixed in the data sources), accommodate for missing terms in the controlled vocabulary (thus requiring an extension of the vocabulary) and when the schema of the data sources changes (e.g. an attribute was supposed to have n possible values and the n+1 value appears). Fixes are recorded and applied automatically in the next updates (Giunchiglia et al., 2021).

Step 6. Implementing the service. The service is implemented and deployed by accessing the knowledge graph data from the Hub via dedicated APIs.

5. The Digital University framework

We developed a dedicated framework to support the creation of the multilingual knowledge graph at the core of the Digital University solution. It appears very simple, as a result of an accurate engineering process.

The framework is simple in that it is entirely developed in Java, it is constituted by only 291 lines of code (that we fully provide in Appendix A), comments included, and it is based on the well-known object-oriented programming paradigm. The Entity Relationship model (Chen, 1976) is employed to represent the various entities and how they are interconnected. The ETL paradigm that is typical of data warehousing approaches to data integration (El-Sappagh et al., 2011) is employed to extract data from the original data sources, to convert them in to entities and to incrementally construct the knowledge graph. Such simplicity allows any programmer, with no specific knowledge of representation languages and Semantic Web technologies, to adopt it very quickly and easily.

The framework is also very efficient for two reasons. The first is that the data integration algorithm is linear in computational complexity. In fact, it employs hash tables to store and retrieve the entities of the knowledge graph. Insertion and retrieval of entities in hash tables takes constant time. The second is that all data structures are stored in RAM memory to guarantee the maximum performance at runtime.

For instance, the knowledge graph of the University of Trento is currently constituted by around 225.000 entities, appropriately selected. Entity types are Person, Organization, Role, Course, Project, Thesis, Publications and Files. The creation of the knowledge graph takes 2-3 minutes (depending also on the network load, given that data sources are located on different servers) on a laptop equipped with an Intel Core i5-7200 dual core 2.50GHz and 2.71 GHz and 8 GB of RAM memory. The total memory usage is around 370 MB.

The framework consists of 10 Java classes. Figure 2 provides an exemplification of the data structures used to represent the knowledge graph. It shows three entitybases and three entities interconnected between them. The mapping between the classes of the framework to the standard W3C RDF schema (https://www.w3.org/TR/rdf-schema/) is trivial.

The knowledge graph is represented as a set of **entitybases** (lines 002-055), one for each entity type required. Within each entitybase, we employ a HashMap. We represent entity types as an integer. We suggest that types could be encoded as constant values, e.g. Person = 0, Organization = 1, Role = 2, Course = 3. Each **entity** (lines 056-095) is characterized by its type, a unique identifier (that we represent as a String) and a set of attributes.

Attributes (lines 096-134) are <name, value> pairs. The current framework supports three different types of attributes (Java classes can be extended to support additional types). **String attributes** (lines 135-144) are language independent data attributes whose value is stored as a String; numbers and dates are converted into strings. **Relational attributes** (lines 145-154) represent relations between entities; in fact, their values (lines 155-180) are <type, id> pairs where type is the entity type and id is the identifier of the target entity. **Concept attributes** (lines 181-190) are language dependent data attributes whose values (lines 191-210) are stored as a Concept (that are simply represented as integers) used to codify values whose labels need to be read according to the languages used, e.g. in English and Italian.

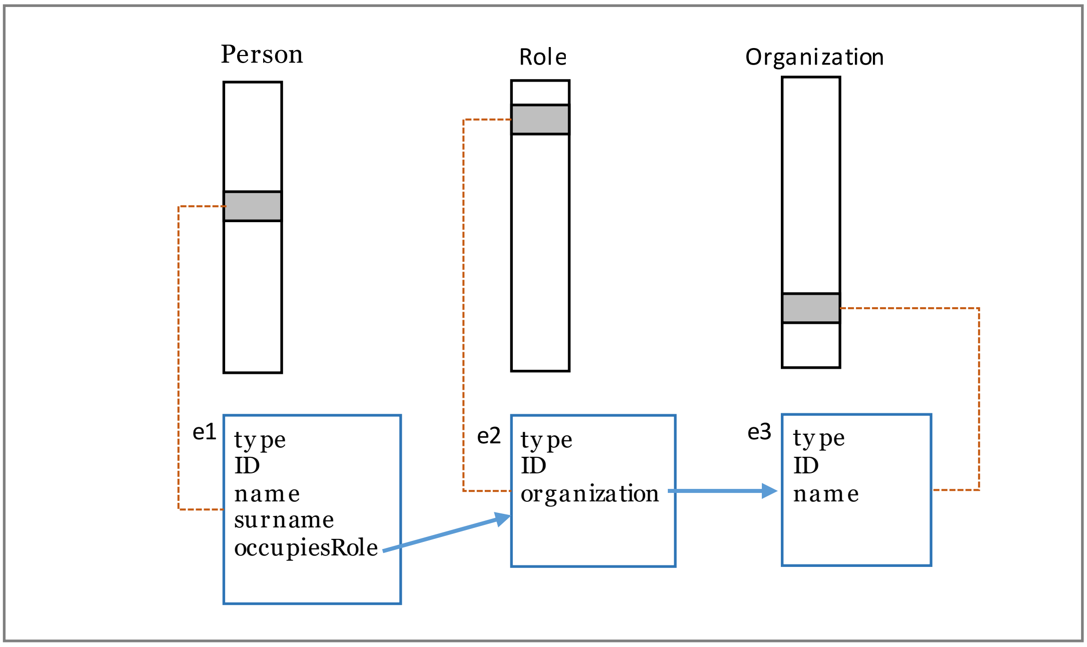

Figure 2 – An exemplification of an entityStore

In order to translate a concept in to a specific language, it is necessary to define one or more **vocabularies** (lines 211-267). Each vocabulary is characterized by a reference language and a list of **concepts** (lines 268-291). Each concept is a triple <id, label, definition>. For example, the concept of researcher in English is given by the triple <56569, "researcher", "(role) a person who conducts research activities">, while in Italian it is given by the triple <56569, "ricercatore", "(ruolo) una persona che svolge attività di ricerca">. In line with the ISO 25964 standard for the representation of vocabularies (https://www.iso.org/standard/53657.html), the identifier must obviously be the same in all vocabularies. The definition is needed to keep track of the meaning of the labels.

The current framework has two main limitations. The first is that it may need significant amount of RAM memory in case of data sources of huge size. For the purposes we envisioned in Trento (see Section 8), the RAM memory used is actually approximately 370 MB only. Such cheap usage of memory is possible also because we only select relevant data to be extracted from the data sources. The second limitation is that the framework may require an extension of the entity matching libraries in case not all sources already provide unique identifiers for all entities or in case similar entities are stored in different databases with different identifiers. We overcome this limitation by making sure that identifiers are always available for all entities, either as a single attribute, or as a result of a combination of multiple attributes.

6. The demonstrative example

Suppose we want to develop a University portal in two languages, English and Italian, whose functionalities have been identified by collecting requirements from the various stakeholders of the university. The local data model will have to define the various entity types and their attributes necessary to accommodate for such requirements. For instance, it may establish that a Person must have name, surname, gender, email, phone and set of positions occupied in administrative units. For sake of simplicity, we assume that data sources have been already pre-processed (for instance, as a result of a job that runs daily in order to get up-to-date information) and that the result is available as CSV files (see Appendix B):

- **people.csv** contains the people affiliated to the University, and in particular each row contains the unique identifier of the person, the surname, the name, the gender (M for male and F for female), the email address and the phone number;
- **units.csv** contains administrative units of the University, where each row contains the unique identifier of the unit, the identifier of the type of unit, the unit name, the identifier of the parent administrative unit on which it depends (in order to reconstruct the organization chart), the email address and the phone number of the unit;
- **types_of_units.csv** contains information about the types of administrative units of the University, and in particular each row contains the identifier of the type of unit, the name of the type and the identifier of the corresponding concept in the vocabularies (for instance as a result of a manual or automatic mapping);
- **positions.csv** contains information about the affiliations of each person, and in particular each row contains the identifier of the person, the identifier of the type of position and the identifier of the administrative unit;
- **types_of_positions.csv** contains information about the types of positions that can be appointed to people in the administrative Units of the University, and in particular each row contains the identifier of the type of position, the name of the type and the identifier of the corresponding concept in the vocabularies (for instance as a result of a manual or automatic mapping);

- **courses.csv** contains information about the courses offered by the University, and in particular each row contains the unique identifier of the degree program, the degree program name, the program type, the identifier of the course, the name of the course, the identifier of the department (an administrative unit), the identifier of the person who teaches the course, a field that is equal 0 in case the person is a professor and 1 in case the person is an assistant.

The two vocabularies are stored in TXT files. Each row contains the identifier of the concept, the label and the definition in the corresponding language. They can be extended as needed. In Figure 3 we provide an example of the content in the English vocabulary.

```
118      | person               | a human being
52974    | rector               | (role) the head of a university
53485    | director             | (role) the person in charge of managing a department or directorate
118272   | deputy director      | (role) the person appointed to represent or act on behalf of the director
56251    | president            | (role) primary leader of a firm or corporation
54235    | director general     | (role) the manager with the highest ranking
53282    | coordinator          | (role) the person responsible for coordinating the activities
54173    | full professor       | (role) a professor of first rank in a university
52409    | associate professor  | (role) a professor of second rank in a university
56569    | researcher           | (role) a person who conducts research activities
118261   | PhD student          | (role) a student who is enrolled in a doctorate school
118264   | staff                | (role) the people responsible of the administrative and technical tasks
43544    | organization         | a group of people who work together
44331    | administrative unit  | an organization regarded as part of a larger social group
45010    | statutory body       | an institutional unit defined by the statute
45016    | governing board      | a board that manages the affairs of an institution
118249   | supporting board     | a board that supports the governing body of an institution
44452    | division             | an administrative unit of second level in government or business
45084    | office               | an administrative unit of basic level in government or business
43989    | academic department  | a division of a university or school
35792    | degree program       | a course of study leading to an academic degree
4553     | course               | education imparted in a series of lessons or meetings
```

Figure 3 - Example of content of the vocabularies (in English)

7. Developing the ETL facilities

In this section we present a demonstrative toy example of how the knowledge graph can be built by implementing ETL facilities and by employing the Digital University framework.

The main functionality offered by an entitybase is data integration, supported by the load method (lines 021-033 in Appendix A) of the class EntityBase. As from the example reported in Figure 4, suppose we extracted enough data to generate the entity e1. In loading the entity e1 in the entitybase E, if E already contains an entity e2 with the same identifier, the set of attributes of e1 are merged with those of e2 (lines 027-030 in Appendix A), thus obtaining the entity e3, otherwise e1 is loaded in E as it is (line 031).

e1		**e2**		**e3**	
type	**= 118**	**type**	**= 118**	**type**	**= 118**
ID	**= 1000099**	**ID**	**= 1000099**	**ID**	**= 1000099**
Surname	= Sordi	Email	= alberto.sordi@unitn.it	Surname	= Sordi
Name	= Alberto	Phone	= 2005	Name	= Alberto
Phone	= 1000			Email	= alberto.sordi@unitn.it
				Phone	= [1000, 2005]

Figure 4 - Example of data integration

Two attributes are considered to be different when the name or the value do not match. In the current implementation, both the entity matching and the attribute functions simply rely on the standard equality (==) operator (lines 126-133 in Appendix A), but according the specific scenario, it could be a more complex similarity function (Köpcke & Rahm, 2010), for instance to accommodate for approximation of values.

Thus, a data integration pipeline can be designed as a set of ETL facilities where for each data source a dedicated facility extracts data (E), translate it into a set of entities (T) and loads each of them in the corresponding entitybase (L). Given that the load function is characterized by a O(1) computational complexity, the ETL algorithm has been designed to have O(n) computational complexity, where n is the number of entities identified in the data sources.

In the following, we continue the example with code necessary to develop the data integration pipeline. It is constituted by 6 Java classes, that we fully provide in Appendix C.

The **EntityStore** (lines 002-099) is the place in which we store the entitybases that constitute the knowledge graph. We implemented it as an array of entitybases. For the example provided in this paper, we defined four entitybases: EB[0] for Person, EB[1] for Organization, EB[2] for Role, EB[3] for Course.

The **data integration pipeline** (lines 100-135) contains the main method. It creates the English and Italian vocabularies by loading the two TXT files that contain the <id, label, definition> triples, and initializes the EntityStore (lines 104-119). Finally, it launches four different ETL facilities to process the CSV files with the data sources and to incrementally construct the knowledge graph (lines 120-126). They can be executed in any order, thus always obtaining the same result.

The first ETL facility (class People, lines 136-187) processes people.csv. Below we exemplify how for the first row of people.csv it creates one entity of type person to be loaded in EB[0]. Here 118 (see line 011) and 90013 (see line 015) are the concept ID for "person" and for "male", respectively, in the vocabularies. Class is the attribute that can be used to further specialize the type, that in this case remain "person".

type	**= 118**
ID	**= 1000099**
Class	= 118
Surname	= Sordi
Name	= Alberto
Gender	= 90013
Email	= alberto.sordi@unitn.it
Phone number	= 1000

The second ETL facility (class Units, lines 188-248) processes units.csv. Below we exemplify how for the first two rows of units.csv it creates two entities of type organization to be loaded in EB[1]. Here 43544 (see line 013), 44834 and 45016 are the concept ID for “organization”, “university” and “academic senate”, respectively, in the vocabularies. The latter two are taken from types_of_units.csv and with the Class attribute further specialize the type “organization”.

type	**= 43544**
ID	**= UNIT00001**
Class	= 44834
Name	= University of Trento

type	**= 43544**
ID	**= UNIT000002**
Class	= 45016
Name	= Academic Senate
Part of	= (1, UNIT00001)

The third ETL facility (class Positions, lines 249-301) processes positions.csv. Below we exemplify how for the first row of positions.csv it creates three entities. The first entity is of type person to be loaded in EB[0], and corresponds to the same person with ID 1000099 created above; it is therefore merged with the first version of the same entity loaded before in EB[0], thus adding the relational attribute Occupies Role. The second entity is of type organization to be loaded in EB[1], and corresponds to the same organization with ID 43544 created above; it is therefore merged with the first version of the same entity loaded before in EB[1], but no additional attributes are added. The third entity is of type role to be loaded in EB[2], where 118247 is the concept ID for “role” (line 12) and the type of position occupied OTHEXT001 is converted in 118264 that is the concept ID of “other staff” (see types_of_positions.csv) that becomes the value of Class. We represent roles similarly to (Jureta et al., 2007).

type **= 118**
ID **= 1000099**
Occupies Role = (2, UNIT000002_118264)

type **= 43544**
ID **= UNIT000002**

type **= 118247**
ID **= UNIT000002_118264**
Class = 118264
Organization = (1, UNIT000002)

The fourth ETL facility (class Courses, lines 302-351) processes courses.csv. Below we exemplify how for the first row it creates three entities. The first is of type person to be loaded in EB[0]. The second is of type organization to be loaded in EB[1]. The third is of type course to be loaded in EB[3], where 4553 is the Concept ID for "course" (line 14).

type **= 118**
ID **= 1000313**

type **= 43544**
ID **= UNIT08624**

type **= 4553**
ID **= 90065**
Class = 4553
Name = Administrative Law
Degree program = Law (LM5)
Department = (1, UNIT08624)
Professor = (0, 1000313)

8. Using the knowledge graph in multiple digital services

Once the knowledge graph has been created, it has to be stored somewhere. In Trento, we generate it once a day and we store it in Elasticsearch indexes (https://www.elastic.co/what-is/elasticsearch), that is a distributed, free and open search and analytics engine for all types of data that offers simple, very efficient and scalable REST APIs to store and query data. On top of Elasticsearch, we then developed an additional layer of RESTful APIs that are used by the various digital services (described in the next section).

To export the knowledge graph that has been created, the EntityStore offers the toJSON function (lines 042-098 in Appendix C) that converts an entity into a JSON object. It takes in input the identifier of the entity to be converted, the type of entity (such that we can identify the entitybase in which it is contained), the vocabulary to be used to translate concept attributes (for instance, the Italian vocabulary) and the depth of the knowledge graph to be taken, i.e. the maximum number of relational attributes to be followed. In alternative, you can directly use the get functions offered by the various classes of the framework.

So far, in Trento we designed and developed four digital services that access to the same knowledge graph.

The **institutional portal** (https://webapps.unitn.it/du/en) is a public communication service that offers a comprehensive webpage (from the integration of 7 different data sources) for each of the University members, academic departments, governing bodies and administrative units. University members include academic staff (professors, researchers, PhD students), administrative and technical staff, and university executives. It provides contact information (email addresses, phone numbers, addresses), CVs, list of publications (that can be downloaded when available in open access), courses, research projects, master and PhD theses (the latter can be downloaded when available in open access). The portal is in English and Italian and it is visited by around half a million unique users per year.

The **institutional dashboard** is a data analytics service providing insights about the quality of research conducted by the faculty members with a focus on publications, research projects and human resources. It helps decision makers with statistics and interactive graphs that are useful to the University governance to examine trends, strengths, and points of improvement. Access is reserved to University members only via credentials.

Dedicated APIs have been developed for the **publication of Open Data** on the regional (https://dati.trentino.it/organization/universita-di-trento), Italian (https://www.dati.gov.it) and European (https://data.europa.eu) data portals. With this interoperability service, the University of Trento complies with national guidelines about sharing public sector information. It is important to notice that the published data are of top quality (uniform schema and terminology, offered in English and Italian, with entities in different Open datasets that link to each other via unique identifiers, updated regularly every day). Moreover, this step does not require additional costs, only that of selecting relevant data from the knowledge graph and publishing it in the appropriate format.

The **University Mobile App** (https://unitrento.app/) is a communication service that has been developed by the IT staff of the University of Trento for its students. The knowledge graph is accessed through dedicated APIs as one of the data sources used.

9. Conclusions

The digital transformation poses new challenges for universities. In particular, universities need to tune their strategies for effective data governance and identify efficient IT solutions to trace and value information about their key assets initially scattered across multiple legacy systems. As part of an overall solution to deal with the unavoidable data fragmentation and diversity, we illustrated the work done in Trento where we designed and implemented an IT infrastructure based on the Hub-and-Spoke paradigm. In this paper, we presented the new framework and the ETL facilities that we developed in 2021 to construct our knowledge graph more efficiently and easily than the first version developed in between 2017 and 2018. The knowledge graph is used consistently by a number of different digital services. We hope that the source code provided here can be of inspiration and can be employed by other universities to develop their own knowledge graphs and digital services.

References

Alenezi M. (2021). Deep Dive into Digital Transformation in Higher Education Institutions. Education Sciences, 11(12):770. https://doi.org/10.3390/educsci11120770

Börner, K, Conlon, M, Corson-Rikert, J, and Ding, Y (2012). VIVO: A semantic approach to scholarly networking and discovery. Synthesis lectures on the Semantic Web: theory and technology, 7(1), 1-178. http://dx.doi.org/10.1007/978-3-031-79435-3

Bouquet, P., Stoermer, H., & Liu, X. (2007). Okkam4P: A Protégé Plugin for Supporting the Re-use of Globally Unique Identifiers for Individuals in OWL/RDF Knowledge Bases. In Proceedings of the Fourth Italian Semantic Web Workshop (SWAP).

Brdesee, H. (2021). A divergent view of the impact of digital transformation on academic organizational and spending efficiency: A review and analytical study on a university E-service. Sustainability, 13(13), 7048. https://doi.org/10.3390/su13137048

Brynjolfsson, E., Hitt, L. M., & Kim, H. H. (2011). Strength in numbers: How does data-driven decision making affect firm performance? Working Paper, Sloan School of Management, MIT, Cambridge, MA.

Buchanan, L., and O'Connell, A. (2006). A brief history of decision making. Harvard Business Review, 84(1), 32-40.

Buckland, M. (1996). Documentation, information science, and library science in the USA. Information processing & management, 32(1), 63-76.

Chatterjee, U, Giunchiglia, F, Madalli, D P, and Maltese, V (2016). Modeling Recipes for Online Search. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", pp. 625-642, Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-48472-3_37

Chen, P. P. S. (1976). The entity-relationship model—toward a unified view of data. ACM transactions on database systems, 1(1), 9-36. http://dx.doi.org/10.1145/320434.320440

Denning, P J (2003). Computer science. Chichester (UK): John Wiley and Sons Ltd.

Dong, X. L., & Naumann, F. (2009). Data fusion: resolving data conflicts for integration. Proceedings of the VLDB Endowment, 2(2), 1654-1655. http://dx.doi.org/10.14778/1687553.1687620

El-Sappagh, S. H. A., Hendawi, A. M. A., & El Bastawissy, A. H. (2011). A proposed model for data warehouse ETL processes. Journal of King Saud University-Computer and Information Sciences, 23(2), 91-104.

Esmailzadeh, H., Mafimoradi, S., Hemmati, A. R., Rajabi, F. (2022). Challenges and policy recommendations for IT governance in the University of Medical Sciences: a case study. Journal of Health Administration, 25(3), 9-29.

Gafurov, I. R., Safiullin, M. R., Akhmetshin, E. M., Gapsalamov, A. R., Vasilev, V. L. (2020). Change of the Higher Education Paradigm in the Context of Digital Transformation: From Resource Management to Access Control. International Journal of Higher Education, 9(3), 71-85. http://dx.doi.org/10.5430/ijhe.v9n3p71

Gartner (2014). Gartner Says One Third of Fortune 100 Organizations Will Face an Information Crisis by 2017. http://www.gartner.com/newsroom/id/2672515

Giunchiglia, F., Soergel, D., Maltese, V., Bertacco, A. (2009). Mapping large-scale knowledge organization systems. Proceedings of the 2nd International Conference on the Semantic Web and Digital Libraries (ICSD).

Giunchiglia, F., Maltese, V., Autayeu, A. (2012). Computing minimal mappings between lightweight ontologies. International Journal on Digital Libraries, 12, 179-193. http://dx.doi.org/10.1007/s00799-012-0083-2

Giunchiglia, F, Maltese, V, and Dutta, B (2012). Domains and context: first steps towards managing diversity in knowledge. Journal of Web Semantics, 12–13, 53-63. http://dx.doi.org/10.1016/j.websem.2011.11.007

Giunchiglia, F, Dutta, B, and Maltese, V (2014). From Knowledge Organization to Knowledge Representation. Knowledge Organization, 41(1), 44-56. http://dx.doi.org/10.5771/0943-7444-2014-1-44

Giunchiglia, F., Batsuren, K., & Freihat, A. A. (2018). One world–seven thousand languages. 19th international conference on computational linguistics and intelligent text processing, CiCling, 18-24.

Giunchiglia, F., Bocca, S., Fumagalli, M., Bagchi, M., & Zamboni, A. (2021). iTelos--Purpose Driven Knowledge Graph Generation. arXiv preprint arXiv:2105.09418.

Giunchiglia, F., Maltese, V., Ganbold, A., & Zamboni, A. (2022). An Architecture and a Methodology Enabling Interoperability within and across Universities. In 2022 IEEE International Conference on Knowledge Graph (ICKG) (pp. 71-78). IEEE. http://dx.doi.org/10.1109/ICKG55886.2022.00017

Gkrimpizi, T., Peristeras, V. (2022). Barriers to digital transformation in higher education institutions. In Proceedings of the 15th International Conference on Theory and Practice of Electronic Governance (pp. 154-160). http://dx.doi.org/10.1145/3560107.3560135

Gómez-Pérez, A (2001): Evaluation of ontologies. International Journal of intelligent systems. 16(3), 391-409.

Hopkins, B, Owens, L, Goetz, M, Gualtieri, M, and Keenan, J (2015). Deliver On Big Data Potential With A Hub-And-Spoke Architecture. Forrester Research.

International Organization for Standardization (2011). ISO 2596-1:2011. Information and documentation-Thesauri and interoperability with other vocabularies: Part 1: Thesauri for information retrieval. ISO.

Jureta, I. J., Faulkner, S., Kolp, M. (2007). An Agent-Oriented Enterprise Model for Early Requirements Engineering. Handbook of Ontologies for Business Interaction, 122. http://dx.doi.org/10.4018/978-1-59904-660-0.ch008

Köpcke, H., & Rahm, E. (2010). Frameworks for entity matching: A comparison. Data & Knowledge Engineering, 69(2), 197-210. http://dx.doi.org/10.1016/j.datak.2009.10.003

Lenzerini, M (2002). Data integration: a theoretical perspective. In: twenty-first ACM SIGMOD-SIGACTSIGART symposium on principles of database systems. ACM, 233-246. http://dx.doi.org/10.1145/543613.543644

Maltese, V. (2023). "The Digital University", invited talk at the Data Scientia meeting, Trento, Italy.

Maltese, V. (2023). "The data-driven university: how to effectively govern, trust and value university data and offer coherent digital services", invited talk at the Digitalization of Universities Conference.

Maltese, V. (2023). "'Cataloguing' experts by competences: the Digital University project", invited talk at the Look beyond Subject indexing of non-book resources International Conference, Rome, Italy.

Maltese, V. (2018). "The data-driven university: how to effectively govern, trust and value university data to face 2020 challenges", invited talk at the European Association of Communication Professionals in Higher Education (EUPRIO) conference, Sevilla, Spain.

Maltese, V. (2017). "Digital University in Trento: Work Done and Next Steps", invited talk at the 4th Knowledge in Diversity Workshop, Trento, Italy.

Maltese, V (2018). Digital transformation challenges for universities: Ensuring information consistency across digital services. Cataloging & Classification Quarterly, 56, 592-606. http://dx.doi.org/10.1080/01639374.2018.1504847

Maltese, V., Giunchiglia, F., Denecke, K., Lewis, P., Wallner, C., Baldry, A., & Madalli, D. (2009). On the interdisciplinary foundations of diversity. At the first Living Web Workshop at the International Semantic Web Conference (ISWC).

Maltese, V., Giunchiglia, F., & Autayeu, A. (2010). Save up to 99% of your time in mapping validation. Proceedings of On the Move to Meaningful Internet Systems, OTM 2010: Confederated International Conferences: CoopIS, IS, DOA and ODBASE, Part II (pp. 1044-1060). Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-16949-6_28

Maltese, V., & Giunchiglia, F. (2016). Search and Analytics Challenges in Digital Libraries and Archives. Journal of Data and Information Quality, 7(3), 10-12. http://dx.doi.org/10.1145/2939377

Maltese, V., & Giunchiglia, F. (2017). Foundations of Digital Universities. Cataloging & Classification Quarterly, 55(1), 26-50. http://dx.doi.org/10.1080/01639374.2016.1245231

Marks, A., Al-Ali, M. (2022). Digital transformation in higher education: a framework for maturity assessment. In COVID-19 Challenges to University Information Technology Governance (pp. 61-81). Cham: Springer International Publishing. http://dx.doi.org/10.1007/978-3-031-13351-0_3

McDonald, M. P., and Rowsell-Jones, A. (2012). The Digital Edge, Exploiting Information and Technology for Business Advantage. Gartner, Inc.

O'Neill, E. T. (2011). FRBR: Functional requirements for bibliographic records. Library resources & technical services, 46(4), 150-159. http://dx.doi.org/10.5860/lrts.46n4.150

Rodríguez-Abitia, G., & Bribiesca-Correa, G. (2021). Assessing digital transformation in universities. Future Internet, 13(2), 52. http://dx.doi.org/10.3390/fi13020052

Safiullin, M. R., Akhmetshin, E. M. (2019). Digital transformation of a university as a factor of ensuring its competitiveness. International Journal of Engineering and Advanced Technology, 9(1), 7387-7390. http://dx.doi.org/10.35940/ijeat.A3097.109119

Smith, M., Barton, M., Bass, M., Branschofsky, M., McClellan, G., Stuve, D., ... & Walker, J. H. (2003). DSpace: An open source dynamic digital repository. D-Lib Magazine. http://dx.doi.org/10.1045/january2003-smith

Sompel, H. V. D., Nelson, M. L., Lagoze, C., and Warner, S. (2004). Resource harvesting within the OAI-PMH framework. D-Lib Magazine, 10 (12).

Sułkowski, Ł. (2023). Managing the Digital University: Paradigms, Leadership, and Organization. Routledge Studies in Organizational Change & Development Series Editor: Bernard Burnes. http://dx.doi.org/10.4324/9781003366409

Tran, E., and Scholtes, G. (2015). Open Data Literature Review. University of California, Berkeley School of Law, 17.

Tungpantong, C., Nilsook, P., Wannapiroon, P. (2021). A conceptual framework of factors for information systems success to digital transformation in higher education institutions. In 2021 9th International Conference on Information and Education Technology (ICIET) (pp. 57-62). IEEE. http://dx.doi.org/10.1109/ICIET51873.2021.9419596

Waller, M. A., & Fawcett, S. E. (2013). Data science, predictive analytics, and big data: a revolution that will transform supply chain design and management. Journal of Business Logistics, 34(2), 77-84. http://dx.doi.org/10.1111/jbl.12010

Wang, J., Li, G., Yu, J. X., & Feng, J. (2011). Entity matching: How similar is similar. Proceedings of the VLDB Endowment, 4 (10), 622-633. http://dx.doi.org/10.14778/2021017.2021020

Watson, H. J., and Wixom, B. H. (2007). The current state of business intelligence. Computer, 40(9), 96-99. http://dx.doi.org/10.1109/MC.2007.331

Zeng, M. L., Žumer, M., and Salaba, A. (2011). Functional requirements for subject authority data (FRSAD): a conceptual model (Vol. 43). IFLA series on bibliographic control. Walter de Gruyter. http://dx.doi.org/10.1515/9783110263787

Appendix A – The source code of the framework

Please always cite in your code information about the original author: Vincenzo Maltese (vincenzo.maltese@unitn.it), University of Trento, Italy.

```
package Hub;

/** EntityBase: A collection of entities of a certain type */
import java.util.ArrayList;
import java.util.HashMap;
public class Entitybase {
        private int type;                               //The type of the entities in the entitybase
        private HashMap<Integer, Entity> items;         //The entities in the entitybase
        /** It creates a new empty entitybase for entities of the specified type
        * @param       type     the type of the entities in the entitybase
        * @param       capacity       the expected capacity of the entitybase */
        public Entitybase(int type, int capacity) {
                this.type = type;
                this.items = new HashMap<Integer, Entity>(capacity);
        }
        /** @return the type of the entitybase */
        public int getType() {
                return this.type;
        }
        /** It creates a new entity in the entitybase.
        * @return null if the entity is not of the expected type */
        public Entity load(Entity e) {
                if (e == null) return null;                 //Check if the entity is null
                if (e.getType() != this.type) return null;  //Check if it is of the right type
                int key = e.getId().hashCode();             //Gets the key
                //Case A: the entity is already in the entitybase (merge the two entities)
                Entity entityOld = this.items.get(key);
                if (entityOld != null) {
                        ArrayList<Attribute> attributes = entityOld.getAttributes();
                for(int i = 0; i < attributes.size(); i++) e.addAttribute(attributes.get(i));
                }
                this.items.put(key, e); //Case B: the entity is not yet in the entitybase
                return e;
        }
        /** @return the entity with given identifier, or null if it is not in the entitybase */
        public Entity get(String identifier) {
                int key = identifier.hashCode();
                return this.items.get(key);
        }
        /** @return true if the entitybase contains an entity with the same key */
        public boolean contains(String identifier) {
                int key = identifier.hashCode();            //Gets the key
                if (this.items.containsKey(key)) return true; //Gets the entity
                else return false;
        }
        /** It removes the entity with the specified identifier.
        * @return the removed entity, or null if the entry is not in the vocabulary */
        public Entity remove(String identifier) {
                int key = identifier.hashCode();
                return this.items.remove(key);
        }
        /** @return the number of entities in the entitybase */
        public int size() {
                return this.items.size();
        }
}

/** An Entity */
public class Entity {
        private int type;                              //The type of the entity
        private String id;                             //The identifier of the entity
        private ArrayList<Attribute> attributes; //The list of attributes of the entity
        /** It creates a new entity
        * @param        type        the type of the entity
        * @param        id          the identifier of the entity */
        public Entity (int type, String id) {
                this.type = type;
                this.id = id;
                this.attributes = new ArrayList<Attribute>();
        }
```

```
    /** It returns the type of the entity */
    public int getType() {
        return this.type;
    }
    /** It returns the id of the entity */
    public String getId() {
        return this.id;
    }
    /** It returns the attributes of the entity */
    public ArrayList<Attribute> getAttributes() {
        return this.attributes;
    }
    /** It adds the attribute of the entity.
    * @return      false if the attribute was already in the list
    * @return      true if the attribute is correctly inserted in the list */
    public boolean addAttribute(Attribute att) {
        if (this.attributes.contains(att)) return false;
        else return this.attributes.add(att);
    }
    /** It removes the attribute of the entity.
    * @return      false if the attribute was not in the list
    * @return      true if the attribute is correctly removed from the list */
    public boolean removeAttribute(Attribute att) {
        if (!this.attributes.contains(att)) return false;
        else return this.attributes.remove(att);
    }
}

/** A generic attribute.  */
public abstract class Attribute {
    private String name;     //The attribute name
    private Object value;    //The attribute value
    /** It creates a new attribute with no name and no value */
    public Attribute() {
        this.name = null;
        this.value = null;
    }
    /** It creates a new attribute with a name and value */
    public Attribute(String name, Object value) {
        this.name = name;
        this.value = value;
    }
    /** It returns the name of the attribute */
    public String getName() {
        return this.name;
    }
    /** It returns the value of the attribute */
    public Object getValue() {
        return this.value;
    }
    /** It sets the name of the attribute */
    public void setName(String name) {
        this.name = name;
    }
    /** It sets the value of the attribute */
    public void setValue(Object value) {
        this.value = value;
    }
    /** It checks if two attributes have same name and value */
    public boolean equals(Object obj) {
        if (obj == this) return true;
        if (obj == null || obj.getClass() != this.getClass()) return false;
        Attribute att = (Attribute) obj;
        if (att.getName().equals(this.name) && att.getValue().equals(this.value)) return true;
        else return false;
    }
}

/** A String attribute. */
public class StringAttribute extends Attribute {
    /** It creates a new String Attribute
    * @param    name    is the name of the attribute
    * @param    value   it is the value of the attribute */
    public StringAttribute(String name, String value) {
        this.setName(name);
        this.setValue(value);
    }
}
```

```
/** A Relational attribute. */
public class RelationalAttribute extends Attribute{
        /** It creates a new Relational Attribute
        * @param       name    is the name of the attribute
        * @param       value   it is the related Entity */
        public RelationalAttribute(String name, EntityValue value){
                this.setName(name);
                this.setValue(value);
        }
}

/** An Entity Value. */
public class EntityValue{
        private int type;                       //The type of the target entity
        private String id;                      //The identifier of the target entity
        /** It creates a new entity value */
        public EntityValue(int type, String id){
                this.type = type;
                this.id = id;
        }
        /** It returns the type of the target entity */
        public int getType(){
                return this.type;
        }
        /** It returns the identifier of the target entity */
        public String getId(){
                return this.id;
        }
        /** It checks if two values are the same */
        public boolean equals(Object obj){
                if (obj == this) return true;
                if (obj == null || obj.getClass() != this.getClass()) return false;
                EntityValue value = (EntityValue) obj;
                if ((value.getType() == this.type) && (value.getId().equals(this.id))) return true;
                else return false;
        }
}
```

```
/** A Concept attribute. */
public class ConceptAttribute extends Attribute {
        /** It creates a new concept attribute
         * @param     name    is the name of the attribute
         * @param     value   it is the identifier of the concept */
        public ConceptAttribute(String name, ConceptValue value) {
                this.setName(name);
                this.setValue(value);
        }
}

/** A concept Value. */
public class ConceptValue {
        private int id;    //The value (identifier of the target concept)
        /** It creates a new concept value */
        public ConceptValue(int id) {
                this.id = id;
        }
        /** It returns the value of the concept value */
        public int getId() {
                return this.id;
        }
        /** It checks if two values are the same */
        public boolean equals(Object obj) {
                if (obj == this) return true;
                if (obj == null || obj.getClass() != this.getClass()) return false;
                ConceptValue value = (ConceptValue) obj;
                if (value.getId() == this.id) return true;
                else return false;
        }
}

/** A vocabulary in a language. */
import java.util.HashMap;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
public class Vocabulary {
        private String language;                       //The language of the vocabulary
        private HashMap<Integer, Concept> items;       //The concepts in the vocabulary
```

```
/** It creates a new vocabulary in the specified language */
public Vocabulary(String lang) {
    this.language = lang;
    this.items = new HashMap<Integer, Concept>();
}
/** @return the language of the vocabulary */
public String getLanguage() {
    return this.language;
}
/** It creates new entries in the vocabulary taken from a file
* @param      path     the path of the file with the vocabulary entries
* @return     true     if the file is read correctly */
public boolean load(String path) {
  try {
    File myObj = new File(path);
    Scanner myReader = new Scanner(myObj);
    while (myReader.hasNextLine()) {
        String data = myReader.nextLine();
        //Selects the elements of the vocabulary item
        String[] entry = data.split("\\|");
        //Note: if any element is missing or malformed, it throws an Exception
        if ((entry[0] != null) && (entry[1] != null)) {
            int key = Integer.parseInt(entry[0].replace("\t", "").trim());
            String word = entry[1].replace("\t", "").trim();
            String definition = "";
            if (entry[2] != null) definition = entry[2].replace("\t", "").trim();
            this.put(new Concept(key, word, definition));
        }
    }
    myReader.close();
  } catch (Exception e) { return false; }
  return true;
}
/** It creates a new entry in the vocabulary.
* @return null if the entry is already in the vocabulary */
public Concept put(Concept c) {
    if (c != null) {
        int key = c.getId();
        if (this.items.containsKey(key)) return null;
```

```
                this.items.put(key, c);
            }
            return c;
        }
        /** @return the concept with the specified identifier.
        * @return null if the entry is not in the vocabulary */
        public Concept get(int identifier) {
            return this.items.get(identifier);
        }
}

/** A concept in a language. */
public class Concept {
        private int id;                     //The identifier
        private String word;                //The word of the concept in the language
        private String definition;          //The definition of the concept in the language
        /** It creates a new concept */
        public Concept(int id, String word, String definition) {
            this.id = id;
            this.word = word;
            this.definition = definition;
        }
        /** It returns the identifier of the concept */
        public int getId() {
            return this.id;
        }
        /** It returns the word of the concept */
        public String getWord() {
            return this.word;
        }
        /** It returns the description of the concept */
        public String getDefinition() {
            return this.definition;
        }
}
```

Appendix B – The demonstrative example

This data is distributed under the CC-BY license. Please always cite in your code information about the original author: Vincenzo Maltese (vincenzo.maltese@unitn.it), University of Trento, Italy.

people.csv

1000099	Sordi	Alberto	M	alberto.sordi@unitn.it	1000
1000100	Favino	Pierfrancesco	M	favino@unitn.it	2000
1000313	Chiari	Walter	M	walter.chiari@unitn.it	3000
1000356	Preziosi	Alessandro	M	alessandro.preziosi@unitn.it	4000
1000383	Loren	Sophia	F	loren@unitn.it	5000
1000421	Haber	Alessandro	M	alessandro.haber@unitn.it	6000
1000537	Gassman	Vittorio	M	vittorio.gassman@unitn.it	7000
1000772	Albanese	Antonio	M	antonio.albanese@unitn.it	8000
1000778	Bellucci	monica	F	monica.bellucci@unitn.it	9000
1000906	Spencer	Bud	M	bud.spencer@unitn.it	1010
1002951	Verdone	Carlo	M	carlo.verdone@unitn.it	1110
1003072	De Sica	Cristian	M	cristian.desica@unitn.it	1210
1003162	Franchi	Franco	M	franco.franchi@unitn.it	1310
1003263	De Filippo	Eduardo	M	eduardo.defilippo@unitn.it	1410
1003412	Volo	Fabio	M	fabio.volo@unitn.it	1510
1003638	Giannini	Giancarlo	M	giancarlo.giannini@unitn.it	1001
1003709	Vitti	Monica	F	monica.vitti@unitn.it	1002
1003946	Zingaretti	Luca	M	luca.zingaretti@unitn.it	1003
1004406	Troisi	Massimo	M	massimo.troisi@unitn.it	2020
1004712	Magnani	Anna	F	anna.magnani@unitn.it	2100
1004729	Rossellini	Isabella	F	i.rossellini@unitn.it	3400
1004783	Villaggio	Paolo	M	paolo.villaggio@unitn.it	3501
1004886	Pozzetto	Renato	M	renato.pozzetto@unitn.it	3575
1006142	Capotondi	Cristiana	F	cristiana.capotondi@studenti.unitn.it	
1006317	Benigni	Roberto	M	roberto.benigni@studenti.unitn.it	
1010507	Memphis	Ricky	M	memphis@studenti.unitn.it	
1020302	Lisi	Virna	F	virna.lisi@studenti.unitn.it	
1035092	Accorsi	Stefano	M	accorsi@unitn.it	3677
1050467	Cardinale	Claudia	F	c.cardinale@unitn.it	5555
1127966	Mastroianni	Marcello	M	marcello.mastroianni@unitn.it	
1169172	Maria Grazia	Cucinotta	F	mg.cucinotta@unitn.it	
1180169	Cortellesi	Paola	F	paola.cortellesi@unitn.it	3333
1185453	Ricci	Elena Sofia	F	es.ricci@unitn.it	
1188141	Tognazzi	Ugo	M	u.tognazzi@unitn.it	
1223105	Hill	Terence	M	terence@unitn.it	1944
1228874	Valle	Anna	F	anna.valle@studenti.unitn.it	
1242040	Scamarcio	Riccardo	M	r.scamarcio@unitn.it	
1244287	Gerini	Claudia	F	c.gerini@unitn.it	1717

units.csv

UNIT00001	TSOATAT000	University of Trento			
UNIT000002	TOIATDR002	Academic Senate	UNIT00001		
UNIT000274	TOIATGI020	Ethical Committee	UNIT00001		
UNIT00012	TSOTATA007	Central Management Directorate	UNIT00001	man@unitn.it	1234
UNIT00143	TSOTATA007	Human Resources Directorate	UNIT00001	hr@unitn.it	
UNIT00237	TSOTATA009	Payroll office	UNIT00143	payroll@unitn.it	
UNIT00477	TSODRDI002	Master in Law	UNIT08624		
UNIT00484	TSODRDI002	Master in Materials Engineering	UNIT08625		
UNIT00867	TSOTATA008	Employee Relations Division	UNIT00143		
UNIT00870	TSOTATA007	IT Directorate	UNIT00012	it@unitn.it	3412
UNIT08624	TSODRRD001	School of Law	UNIT00001		1111
UNIT08625	TSODRRD001	School of Industrial Engineering	UNIT00001		4321
UNIT08628	TSODRRD001	School of Mathematics	UNIT00001		1324
UNIT08898	TSOTATA008	Industrial Engineering Staff	UNIT08625	ii.taff@unitn.it	
UNIT08909	TSOTATA008	Law Staff	UNIT08624	law.staff@unitn.it	
UNIT09626	TSOTATA009	Archive and Postal Services	UNIT00012	postalserv@unitn.it	
UNIT09701	TSODRDI008	Doctorate School in Engineering	UNIT08625		
UNIT12278	TSOTATA008	Law Division	UNIT00012		
UNIT12831	TSOTATA009	Recruitment Office	UNIT00143		

types_of_units.csv

TOIATDR002	Academic Senate	45016
TOIATGI020	Commission for the implementation of the Ethics Code	118249
TSOATAT000	University	44834
TSOTATA007	First level unit	118251
TSOTATA008	Second level unit	44452
TSOTATA009	Third level unit	45084
TSODRRD001	Department	43989
TSODRDI008	Doctorate program	35792
TSODRDI002	Master program	35792

positions.csv

1000099	OTHEXT001	UNIT000002
1000099	OTHEXT001	UNIT00001
1000099	RUIISTI020	UNIT000002
1000099	RUIISTI088	UNIT00001
1000100	FACADD019	UNIT00001
1000100	RUOORGA001	UNIT00001
1000313	FACREG002	UNIT08624
1000313	OTHEXT001	UNIT08624

1000313	RUIISTI069	UNIT08624
1000313	RUOORGA001	UNIT08624
1000313	RUOORGA021	UNIT08624
1000356	FACREG002	UNIT08628
1000356	RUOORGA001	UNIT08628
1000383	PTAREG002	UNIT09626
1000383	RUOORGA001	UNIT09626
1000421	OTHEXT001	UNIT00001
1000421	PTAREG003	UNIT00012
1000421	PTAREG003	UNIT12278
1000421	RUIISTI126	UNIT00001
1000421	RUOORGA007	UNIT00012
1000421	RUOORGA012	UNIT12278
1000537	FACADD001	UNIT08624
1000537	PTAREG002	UNIT08909
1000537	RUOORGA001	UNIT08624
1000537	RUOORGA001	UNIT08909
1000772	FACREG003	UNIT08628
1000772	RUOORGA001	UNIT08628
1000778	FACREG002	UNIT08624
1000778	OTHEXT001	UNIT000002
1000778	OTHEXT001	UNIT00001
1000778	RUIISTI002	UNIT00001
1000778	RUIISTI020	UNIT000002
1000778	RUOORGA001	UNIT08624
1000906	FACREG002	UNIT08624
1000906	OTHEXT001	UNIT000274
1000906	RUIISTI018	UNIT000274
1000906	RUOORGA001	UNIT08624
1002951	OTHEXT001	UNIT000002
1002951	RUIISTI020	UNIT000002
1003072	FACREG003	UNIT08624
1003072	RUOORGA001	UNIT08624
1003162	FACLOC001	UNIT09701
1003162	FACREG002	UNIT08625
1003162	FACREG002	UNIT09701
1003162	RUOORGA001	UNIT08625
1003162	RUOORGA004	UNIT09701
1003162	RUOORGA008	UNIT09701
1003263	PTAREG003	UNIT00143
1003263	RUOORGA012	UNIT00143
1003412	FACREG006	UNIT08624
1003412	RUOORGA001	UNIT08624
1003638	FACREG002	UNIT08625

1003638	OTHEXT001	UNIT000002
1003638	OTHEXT001	UNIT00001
1003638	RUIISTI001	UNIT00001
1003638	RUIISTI018	UNIT000002
1003638	RUOORGA001	UNIT08625
1003709	PTAREG002	UNIT00237
1003709	RUOORGA012	UNIT00237
1003946	PTAREG002	UNIT00867
1003946	RUOORGA012	UNIT00867
1004406	FACREG002	UNIT08625
1004406	OTHEXT001	UNIT08625
1004406	RUIISTI069	UNIT08625
1004406	RUOORGA001	UNIT08625
1004406	RUOORGA021	UNIT08625
1004712	FACREG002	UNIT08628
1004712	OTHEXT001	UNIT08628
1004712	RUIISTI069	UNIT08628
1004712	RUOORGA001	UNIT08628
1004712	RUOORGA021	UNIT08628
1004729	PTAREG002	UNIT08909
1004729	RUOORGA012	UNIT08909
1004783	PTAREG002	UNIT12831
1004783	RUOORGA012	UNIT12831
1004886	FACADD004	UNIT08624
1004886	OTHEXT001	UNIT000274
1004886	RUIISTI020	UNIT000274
1004886	RUOORGA001	UNIT08624
1006142	RUOORGA009	UNIT00477
1006142	STUPOR001	UNIT00477
1006317	RUOORGA009	UNIT00477
1006317	STUPOR001	UNIT00477
1010507	RUOORGA009	UNIT00484
1010507	STUPOR001	UNIT00484
1020302	RUOORGA009	UNIT00484
1020302	STUPOR001	UNIT00484
1035092	PTAREG002	UNIT09626
1035092	RUOORGA012	UNIT09626
1050467	PTAREG002	UNIT08898
1050467	RUOORGA001	UNIT08898
1127966	FACADD006	UNIT08628
1127966	RUOORGA001	UNIT08628
1169172	FACADD006	UNIT08624
1169172	RUOORGA001	UNIT08624
1180169	FACADD017	UNIT08625

1180169	RUOORGA001	UNIT08625
1185453	FACADD006	UNIT08625
1185453	RUOORGA001	UNIT08625
1185453	RUOORGA009	UNIT09701
1185453	STUPGR001	UNIT09701
1188141	FACADD003	UNIT08628
1188141	RUOORGA001	UNIT08628
1223105	OTHEXT001	UNIT00001
1223105	PTAREG003	UNIT00870
1223105	RUOORGA012	UNIT00870
1228874	OTHEXT006	UNIT00870
1228874	RUOORGA001	UNIT00870
1242040	RUOORGA009	UNIT09701
1242040	STUPGR001	UNIT09701
1244287	PTAREG002	UNIT00237
1244287	RUOORGA001	UNIT00237

types_of_positions.csv

FACADD001	Contract professor	118256
FACADD004	Teaching Assistant	118256
FACLOC001	Teacher	118256
FACADD006	Junior researcher	118259
STUPGR001	PhD student	118261
OTHEXT001	Other staff	118264
PTAREG002	Administrative staff	118264
PTAREG003	Executive manager	118266
OTHEXT006	Student worker	118269
RUIISTI126	General director	54235
RUIISTI002	Vice-rector	118274
RUIISTI088	Rector's delegate	118275
RUIISTI020	Member	37
RUOORGA001	Affiliate	52183
FACREG003	Associate professor	52409
RUIISTI001	Rector of the University	52974
RUOORGA004	Coordinator	53282
RUIISTI069	Director of the department	53485
RUIISTI070	Director of the research center	53485
FACADD019	Emeritus professor	53787
RUOORGA009	Enrollee	53826
FACREG002	Full professor	54173
RUOORGA012	Head	54485
RUOORGA014	Head of the University	54485
RUOORGA021	Head of the department	54485

RUOORGA007	General director	54485
RUIISTI018	President	56251
FACREG006	Fixed-term researcher	56569
FACADD017	Researcher	56569
STUPOR001	Student	57408
STUUGR001	Bachelor student	57408
STUGRA001	Master student	57408
STUGRA002	Master student 5 years degree	57408
RUOORGA008	Teaches in	57580
FACADD003	Visiting professor	57983

courses.csv

10055	Law	LM5	90065	Administrative Law	UNIT08624	1000313	0
10055	Law	LM5	90065	Administrative Law	UNIT08624	1006317	1
10055	Law	LM5	86600	Criminal Law	UNIT08624	1003072	0
10055	Law	LM5	85287	Comparative Private Law	UNIT08624	1003072	0
10055	Law	LM5	90058	Roman Law	UNIT08624	1000313	0
10055	Law	LM5	90059	Comparative Legal Systems	UNIT08624	1003412	0
10115	Mathematics	L2	87747	Algorithms and data structures	UNIT08628	1000099	0
10115	Mathematics	L2	87747	Algorithms and data structures	UNIT08628	1006142	1
10115	Mathematics	L2	89070	Numerical Analysis	UNIT08628	1004712	0
10115	Mathematics	L2	94873	Algebra	UNIT08628	1004712	0
10115	Mathematics	L2	89858	Geometry	UNIT08628	1000356	0
10170	Mathematics	LM	92173	History of Mathematics	UNIT08628	1188141	0
10170	Mathematics	LM	94245	Advanced Number Theory	UNIT08628	1000099	0
10170	Mathematics	LM	90659	Statistics	UNIT08628	1004712	0
10170	Mathematics	LM	91161	Group theory	UNIT08628	1002951	0
10170	Mathematics	LM	92623	Mathematical Physics	UNIT08628	1000772	0
10563	Materials Engineering	LM	92502	Biomechanics	UNIT08625	1004406	0
10563	Materials Engineering	LM	92070	Glass engineering	UNIT08625	1003638	0
10563	Materials Engineering	LM	95321	Properties of materials	UNIT08625	1004406	0
10563	Materials Engineering	LM	95319	Product design	UNIT08625	1004406	0
10563	Materials Engineering	LM	92075	Sustainable materials	UNIT08625	1004406	0
10563	Materials Engineering	LM	92075	Sustainable materials	UNIT08625	1242040	1
10563	Materials Engineering	LM	92075	Sustainable materials	UNIT08625	1185453	1

Appendix C – The source code of the ETL facilities

Please always cite in your code information about the original author: Vincenzo Maltese (vincenzo.maltese@unitn.it), University of Trento, Italy.

```
package ETL;

/** EntityStore: A collection of entitybases */
import Hub.*;
import java.util.ArrayList;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;
public class EntityStore {
        //Vocabulary entries
        static final int voc_person              = 118;          //Class Person
        static final int voc_role                = 118247;       //Class Role
        static final int voc_org                 = 43544;        //Class Organization
        static final int voc_course              = 4553;         //Class Course
        static final int voc_male                = 90013;        //Gender
        static final int voc_female              = 90019;        //Gender
        //Expected capacities (recommended initial size should be the at least the double of expected entities divided by 0.75)
        static final int personCapacity          = 100;  //People
        static final int organizationCapacity    = 100;  //Organizations
        static final int roleCapacity            = 300;  //Roles
        static final int courseCapacity          = 100;  //Courses
        //Entitybases
        private static final int size = 4;       //The number of entitybases
        public static final int personType       = 0;
        public static final int organizationType = 1;
        public static final int roleType         = 2;
        public static final int courseType       = 3;
        public static Entitybase[] EB = new Entitybase[size];
```

```
    /**Initialization of the EntityStore */
    public EntityStore() {
        EB[personType] = new Entitybase(personType, personCapacity);
        EB[organizationType] = new Entitybase(organizationType, organizationCapacity);
        EB[roleType] = new Entitybase(roleType, roleCapacity);
        EB[courseType] = new Entitybase(courseType, courseCapacity);
    }
    /** It converts the entity into a JSON object
    * @param id the ID of the entity to be converted
    * @param type the type of the entity to be converted
    * @param v the vocabulary to be used to translate concept attributes
    * @param depth         the number of relational attributes to be followed
    * @return a JSON representation of the entity */
    public static JSONObject toJSON(String id, int type, Vocabulary v, int depth) {
        if (depth < 0) return null; //Depth must be >= 0
        //It gets the Entitybase of the specified type
        Entitybase base = EB[type];
        if (base == null) return null;
        //It gets the entity
        Entity e = base.get(id);
        if (e == null) return null;
        //It computes the string
        JSONObject obj = new JSONObject();
        obj.put("Identifier", e.getId());
        //It processes the attributes
        JSONArray attArray = new JSONArray();
        ArrayList<Attribute> attributes = e.getAttributes();
        for(int i = 0; i < attributes.size(); i++) {
            Attribute a = attributes.get(i);
            JSONObject att = new JSONObject();
            //Concepts attributes
            if (a instanceof ConceptAttribute) {
                ConceptValue value = (ConceptValue)a.getValue();
                int target = value.getId();
                Concept c = v.get(target);
                att.put("Type", "Concept");
                att.put("Name", a.getName());
                if (c != null) {
                    att.put("Value", c.getWord());
                    att.put("Target", target);
                }
                else {
                    att.put("Value", "");
                    att.put("Target", target);
                }
            }
            //Relational attributes
            else if (a instanceofRelationalAttribute) {
             EntityValue value = (EntityValue)a.getValue();
             int targetType = value.getType();
             String targetValue = value.getId();
             att.put("Type", "Relational");
             att.put("EType", targetType);
             att.put("Name", a.getName());
             att.put("Target", targetValue);
             if (depth > 0)
              att.put("Value", toJSON(targetValue, targetType,  v, depth-1));
            }
            //String attributes
            else {
             att.put("Type", "String");
             att.put("Name", a.getName());
             att.put("Value",a.getValue());
            }
            //Add the attribute
            attArray.add(att);
        }
        obj.put("Attributes", attArray);
        return obj;
    }
}

/** ETL: Extract, Translate and Load facilities */
import Hub.Vocabulary;
public class ETL {
  public static void main(String[] args) {
    //Create a new English vocabulary
    Vocabulary english = new Vocabulary("English");
    String path = "C:\\...\\src\\data\\english.txt";
```

```
        boolean read = english.load(path);
        if (!read) {
                System.out.println("DEBUG:  Vocabuary has not been read correctly!");
                System.exit(0);
}
        //Create a new Italan vocabulary
        Vocabulary italian = new Vocabulary("Italian");
        path = "C:\\...\\src\\data\\italian.txt";
        read = italian.load(path);
        if (!read) {
                System.out.println("DEBUG:  Vocabuary has not been read correctly!");
                System.exit(0);
        }
        //Create the EntityStore with all the necessary Entitybases
        EntityStore ES = new EntityStore();
        //Process the various data sources
        People.process();
        Units.process();
        Positions.process();
        Courses.process();
        //TEST: Get entities with a certain identifier
        System.out.println("An example of generated String in English");
        System.out.println(EntityStore.toString("92075", EntityStore.courseType, english, 2));
        System.out.println("An example of generated JSON in English");
        System.out.println(ES.toJSON("92075", EntityStore.courseType, english, 4).toJSONString());
        System.out.println("An example of generated JSON in Italian");
        System.out.println(ES.toJSON("92075", EntityStore.courseType, italian, 4).toJSONString());
    }
}

/** People: ETL for people */
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
public class People {
    //Data files
    private static final String people = "C:\\...\\src\\data\\people.csv";
    private static final String splitBy = ";";
    //Processing entities
    public static void process() {
        //Readers for data
        BufferedReader bufferPeople = null;
        try {
            bufferPeople = new BufferedReader(new FileReader(people));
        } catch (FileNotFoundException e1) { e1.printStackTrace(); }
        String line = "";
        try {
          //Extract People
          while ((line = bufferPeople.readLine()) != null) {

            //Attributes of a Person
            String[] person = line.split(splitBy, -1);
            String ID      = person[0];
            String surname = person[1];
            String name    = person[2];
            String gender  = person[3];
            String email   = person[4];
            String phone   = person[5];
            //Prepare the new entity with the specified ID
            Entity e = new Entity(EntityStore.personType, ID);
            //Class
            e.addAttribute(new ConceptAttribute("Class", ConceptValue(EntityStore.voc_person)));
            //Surname
            e.addAttribute(new StringAttribute("Surname", surname));
            //Name
            e.addAttribute(new StringAttribute("Name", name));

            //Gender
            int sex = EntityStore.voc_male;
            if (gender.compareTo("F") == 0) sex = EntityStore.voc_female;
            e.addAttribute(new ConceptAttribute("Gender", new ConceptValue(sex)));
            //Email
            if ((email != null) && (email.length() > 0))
```

```
                e.addAttribute(new StringAttribute("Email", email));
                //Phone
                if ((phone != null) && (phone.length() > 0))
                e.addAttribute(new StringAttribute("Phone number", phone));
                //Load the entity into the entitybase
                EntityStore.EB[EntityStore.personType].load(e);
            }
        } catch (IOException e) { e.printStackTrace(); }
    }
}

/** Units: ETL for administrative units */
import java.util.HashMap;
public class Units {
    //Data files
    private static final String unitTypes = "C:\\...\\src\\data\\types_of_units.csv";
    private static final String units = "C:\\...\\src\\data\\units.csv";
    private static final String splitBy = ";";
    private static final int orgTypesCapacity = 100;   //Expected capacity - Types of organizations
    public static void process() {
        //Readers for data
        BufferedReader bufferTypes = null;
        BufferedReader bufferUnits = null;
        try {
            bufferTypes = new BufferedReader(new FileReader(unitTypes));
            bufferUnits = new BufferedReader(new FileReader(units));
        } catch (FileNotFoundException e1) { e1.printStackTrace(); }
        String line = "";
        try {
            //Extract Unit Types: prepares HUB IDs of the organization types
            HashMap<String, String> orgTypes = new HashMap<String, String>(orgTypesCapacity);
            while ((line = bufferTypes.readLine()) != null) {
                String[] unitType = line.split(splitBy, -1);
                String typeID = unitType[0];
                String conceptID = unitType[2];
                orgTypes.put(typeID, conceptID);
            }
            //Extract Units
            while ((line = bufferUnits.readLine()) != null) {
                //Attributes of a Unit
                String[] unit = line.split(splitBy, -1);
                String unitID     = unit[0];
                String unitType = unit[1];
                String unitName         = unit[2];
                String unitParentID = unit[3];
                String unitEmail        = unit[4];
                String unitPhone        = unit[5];
                //Prepare the organization entity with the specified ID
                Entity e = new Entity(EntityStore.organizationType, unitID);
                //Class
                Integer conceptID = Integer.parseInt(orgTypes.get(unitType));
                e.addAttribute(new ConceptAttribute("Class", new ConceptValue(conceptID)));
                //Name
                e.addAttribute(new StringAttribute("Name", unitName));
                //Email
                if ((unitEmail != null) && (unitEmail.length() > 0))
                e.addAttribute(new StringAttribute("Email", unitEmail));

                //Phone
                if ((unitPhone != null) && (unitPhone.length() > 0))
                e.addAttribute(new StringAttribute("Phone number", unitPhone));
                //Parent unit entity
                if (unitParentID.length() != 0) {
                    Entity parent = new Entity(EntityStore.organizationType, unitParentID);
                    EntityStore.EB[EntityStore.organizationType].load(parent);
                    e.addAttribute(new RelationalAttribute("Part of", new EntityValue(EntityStore.organizationType,unitParentID)));
                }
                //Load the entity into the entitybase
                EntityStore.EB[EntityStore.organizationType].load(e);
            }
        } catch (IOException e) { e.printStackTrace(); }
    }
}
```

```
/** Positions: ETL for positions of people within administrative units */
public class Positions {
        //Data files
        private static final String positionTypes = "C:\\...\\src\\data\\types_of_positions.csv";
        private static final String positions = "C:\\...\\src\\data\\positions.csv";
        private static final String splitBy = ";";
        private static final int posTypesCapacity = 50;    //Expected capacity - Types of positions
        public static void process() {
          //Readers for data
          BufferedReader bufferTypes = null;
          BufferedReader bufferPosit = null;
          try {
                bufferTypes = new BufferedReader(new FileReader(positionTypes));
                bufferPosit = new BufferedReader(new FileReader(positions));
          } catch (FileNotFoundException e1) { e1.printStackTrace(); }
          String line = "";
          try {
                //Extract Position Types: prepares HUB IDs of the position types
                HashMap<String, String> posTypes = new HashMap<String,
                String>(posTypesCapacity);
                while ((line = bufferTypes.readLine()) != null) {
                        String[] posType = line.split(splitBy, -1);
                        String typeID = posType[0];
                        String conceptID = posType[2];
                        posTypes.put(typeID, conceptID);
                }
                //Extract Positions
                while ((line = bufferPosit.readLine()) != null) {
                        String[] unit = line.split(splitBy, -1);
                        String personID = unit[0];
                        String positionType = unit[1];
                        String unitID = unit[2];

                        //Prepare the person entity with the specified ID
                        Entity e = new Entity(EntityStore.personType, personID);
                        //Prepare the organization entity with the specified ID
                        EntityStore.EB[EntityStore.organizationType].load(new
                        Entity(EntityStore.organizationType,unitID));
```

```
            //Compute the ID of the Role
            String roleID = unitID + "_" + positionType;
            //Prepare the role entity
            Entity r = new Entity(EntityStore.roleType, roleID);
            //Add the role attributes
            Integer conceptID = Integer.parseInt(posTypes.get(positionType));
            r.addAttribute(new ConceptAttribute("Class", new ConceptValue(conceptID)));
            r.addAttribute(new RelationalAttribute("Organization", new EntityValue(EntityStore.organizationType,unitID)));
            //Load the role entity into the entitybase
            EntityStore.EB[EntityStore.roleType].load(r);

            //Add the relational attribute to the person
            e.addAttribute(new RelationalAttribute("Occupies Role", new EntityValue(EntityStore.roleType,roleID)));
            //Load the person entity into the entitybase
            EntityStore.EB[EntityStore.personType].load(e);
        }
      } catch (IOException e) { e.printStackTrace(); }
    }
}

/** Courses: ETL for courses */
public class Courses {
    //Data files
    private static final String courses = "C:\\...\\src\\data\\courses.csv";
    private static final String splitBy = ";";
    public static void process() {
        //Readers for data
        BufferedReader bufferCourses = null;
        try {
            bufferCourses = new BufferedReader(new FileReader(courses));
        } catch (FileNotFoundException e1) { e1.printStackTrace(); }
        String line = "";
        try {
          //Extract Courses
          while ((line = bufferCourses.readLine()) != null) {
            //Attributes of a Course
            String[] course = line.split(splitBy, -1);
```

```
                    String programName    = course[1];
                    String programType    = course[2];
                    String courseID       = course[3];
                    String courseName     = course[4];
                    String departmentID   = course[5];
                    String personID       = course[6];
                    int assistant         = Integer.parseInt(course[7]);
                    //Prepare the course entity with the specified ID
                    Entity e = new Entity(EntityStore.courseType, courseID);
                    //Class
                    e.addAttribute(new ConceptAttribute("Class", new ConceptValue(EntityStore.voc_course)));
                    //Course name
                    e.addAttribute(new StringAttribute("Name", courseName));
                    //Degree program
                    String completeProgramName = programName + "(" + programType + ")";
                    e.addAttribute(new StringAttribute("Degree program", completeProgramName));
                    //Prepare the Department entity (we just get the ID)
                    Entity org = new Entity(EntityStore.organizationType,departmentID);
                    EntityStore.EB[EntityStore.organizationType].load(org);
                    //Add the relational attribute to the course
                    e.addAttribute(new RelationalAttribute("Department", new EntityValue(EntityStore.organizationType, departmentID)));
                    //Prepare the Person entity (we just get the ID)
                    Entity person = new Entity(EntityStore.personType,personID);
                    EntityStore.EB[EntityStore.personType].load(person);
                    //Add the relational attribute to the course
                    if (assistant == 1)     e.addAttribute(new RelationalAttribute("Assistant", new EntityValue(EntityStore.personType,personID)));
                    else   e.addAttribute(new RelationalAttribute("Professor", new EntityValue(EntityStore.personType,personID)));
                    //Load the entity into the entitybase
                    EntityStore.EB[EntityStore.courseType].load(e);
                }
            } catch (IOException e) { e.printStackTrace(); }
    }
```

```
}
```