Using RDF to describe WLCG's topology with the CRIC data model

Simone Rossi Tisbeni

November 2022

1 Introduction

The Worldwide LHC Computing Grid (WLCG) project is a global collaboration of more than 170 computing centres in 42 countries, linking up national and international Grids to build and maintain of a distributed computing infrastructure. The purpose of WLCG is to give near real-time access to data generated by the Large Hadron Collider (LHC), for scientific computing [1]. It's the world's largest computing grid, effectively merging several GRID infrastructures – such as the European Grid Infrastructure (EGI) [2] and Open Science Grid (OSG) [3]. The resources WLCG provides access to include: data storage capacity, processing power, sensors, visualisation tools and more.

The WLCG is composed of computing centres organised by levels, called "Tiers"; each Tier provides specific services, storage capacity, CPU power and network connectivity. This structure made of hierarchical Tiers was formalised in the work done by the "Models of Networked Analysis at Regional Centres for LHC Experiments" team (MONARC), which produced a model still referred today as "the MONARC model" [4]. Following this model, the Tier levels are labelled with numbers from 0 to 3, in which the number indicates the level and complexity of services offered by the centre: namely, the lower the number the higher the level and quantity of services.

The LHC is currently in its 3rd Run, scheduled to last until 2024 included. Subsequently, the Phase-II upgrade will take place between 2024 and 2026, and will eventually complete the increase in luminosity, yielding to the next generation of the LHC experiments - High Luminosity LHC (HL-LHC) [5]. Such a large and wide scope project requires large investments also in the R&D of software and computing systems needed to acquire, manage, process, and analyse the huge amount of data that will be recorded.

The new computing requirements for HL-LHC are significant: the system should be able to manage multiple Exabyte of data per year and would probably require about 20M cores for data processing [6]. This infrastructure would require heterogeneous resources including HPC, specialised clusters, commercial and non-commercial clouds. In addition, different types of computing platforms like CPU, GPU and FPGAs should be used to increase throughput.

Any tool used to describe the distributed resources of the WLCG, should be able to manage all these heterogeneous systems. The efforts of the WLCG Information System is to provide easy service discovery of shared physical resources and detailed description of service configurations; this is done through the use of the Computing Resource Information Catalogue (CRIC), that will be described in detail in the following section.

2 CRIC

CRIC, as a high-level information system, is focused onto describing the topology of the WLCG infrastructure as well as experiment-specific configuration required to exploit this infrastructure according to the experiments computing models, performing data validation on top [7].

CRIC was build as the evolution of the ATLAS Grid Information System (AGIS) [8], from which it inherits the main concepts for resource declaration: the description provides a clear separation between the resources *provided by* the distributed sites and the information on how the resource is *used by* the experiments.

In particular, *provided by* is used mainly for topology description of the distributed computing infrastructure; listing services endpoints, site location, implementation, software versions, etc. *Used by*, on the other hand, defines how the experiments use a particular resource and how it is integrated into the experiment's computing system. Both properties are provided by different user roles (i.e. site admin and service providers for the first and operations teams or experiment-support teams at the sites for the second), with different access privileges in CRIC.

Additionally, CRIC is built to be easily deployed and configured by any collaboration, even beyond LHC. This, in addition to the modular and easily extendable architecture, enables any team or experiment to build on top of CRIC, describing their own topology. This description of their resources can then be used to configure any possible third party clients.

2.1 Architecture

CRIC follows a modular architecture with clear building blocks that can be combined to create different CRIC instances. Every building block can be modified and plugged-in to create a custom deployment.

The modular structure allows the experiments to customise the system's functionality in the experiment plugins while performing common tasks, like collecting data from the primary information sources, with CRIC core. The information is collected from multiple sources and stored into the CRIC database. The data can then be accessed vie WEB UI and exposed to applications via REST APIs. The overall architecture is summarised in Figure 1.

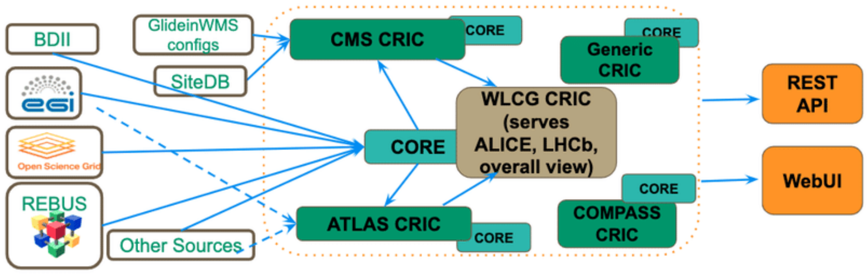

Figure 1: Diagram of CRIC's architecture

2.2 Data models

CRIC's data models are divided into two categories: core and experiment-specific [9].

The core CRIC data models have been designed to identify commonalities between various experiments to benefit from handling and validating the shared data structures centrally. Core data models describe the WLCG infrastructure and it's various components such as:

Federations which describes data federations as defined for WLCG, with main attributes Federation name, Accounting name, Tier level, Country;

Regional Center Sites both collected from topology databases and manually added by the experiments, with reference to the federation it belongs to and attributes such as Timezone, Admin email, pledge status, etc;

Services which describes any kind of service of the WLCG infrastructure(i.e. storage, computing, etc.), with generic attributes (i.e. type, flavour, end-point, reference to the RC site where the service is located, state, status, timestamp of the latest modification), and link to additional data models based on their description (i.e. resource , queue, service-protocol, etc.);

The experiment-specific data models provide configuration required by the workload and data management systems of the experiments in order to exploit the WLCG resources. The majority of data models even in the experiment-specific plugins have quite a lot of common structures. At the same time, there are concepts which have been implemented for a particular experiment [10].

For example, the CMS experiment, defined a specific CRIC plugin to satisfy their requirements [11]. For this purpose, new components not present in the core data models have been introduced:

CMS Site which describes how CMS uses and declares Experiment Site, referencing the RC site in CORE, but with single RC site capable to be mapped to several CMS sites;

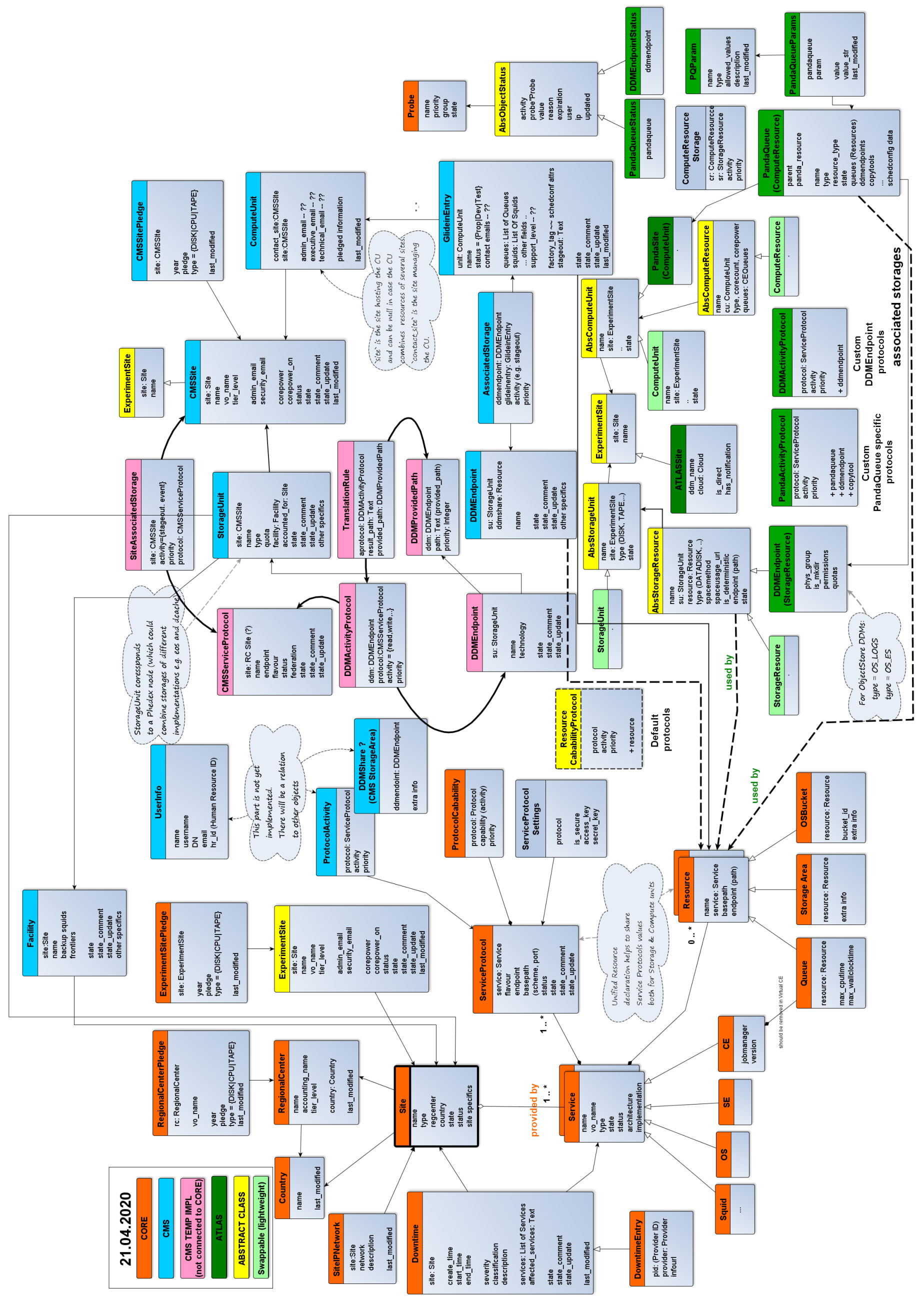

Figure 2: Diagram of the CRIC data model. Only a subset of the data has been used for the sample ontology.

Facility that describe a given administrative domain and combines one or several CMS sites, with main attributes referencing various support units (executives, site admins, storage admins and data managers);

Compute and Storage Units which describes a set of compute or storage resources at a given CMS site, handling computing and storage capacity information of the CMS site, pledge fraction , absolute pledge value, potential max of CPUs and Data quota, CPU bound maximum and I/O bound maximum;

A diagram of the CRIC data model is shown in Figure 2.

3 Linked Data

Data published on CRIC is made available as raw dumps in formats such as CSV or JSON, sacrificing much of its structure and semantics. The same situation can be found in Web, as the relationship between two linked item is implicit, when the data format, i.e. HTML, is not sufficiently expressive to describe links to related entities.

Modern Web best practices have lead to the extension of the Web with a global data space connecting data from diverse domains. This is often referred to as Web of Data, a paradigm which enables new types of applications, linking data to their semantic meaning via explicit structures [12].

Linked Data is simply about creating typed links between data from different sources. These range from databases maintained by two different organisations, or simply heterogeneous systems with the same maintainer. Formally, it refers to data published in a machine-readable format, where its meaning is explicitly defined, can be linked to other external data sets, and can in turn be linked to from external data sets.

Linked Data relies on two technologies do define and link entities: Uniform Resource Identifiers (URIs) [13] and the HyperText Transfer Protocol (HTTP). URIs and HTTP are then supplemented by a technology that is critical to the Web of Data – the Resource Description Framework (RDF)[14].

3.1 RDF

RDF is a framework to represent information on the web. Is defined as a graph-based data model that encodes data in the form of sets of subject-predicate-object triples. The subject and object of a triple are URIs that each identify a resource. The predicate specifies how the subject and object are related, and can also represented by a URI.

A set of such triples is called an RDF graph. An RDF graph can be visualised as a node and directed-arc diagram, in which each triple is represented as a node-arc-node link. An RDF dataset is a collection of RDF graphs. An RDF document is a document that encodes an RDF dataset in a RDF syntax (i.e. XML, TURTLE), and allows the exchange of RDF data between systems.

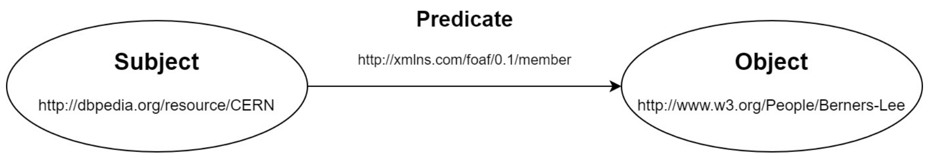
Predicate
http://xmlns.com/foaf/0.1/member
Subject
http://dbpedia.org/resource/CERN
Object
http://www.w3.org/People/Berners-Lee

Figure 3: Sample RDF graph

Two resources in a RDF graph can point to different data sets, allowing data in one data source to be linked to that in another, thereby creating a Web of Data. Figure 3 shows an example of a RDF graph linking entities fro two different data sets. Notice how the predicate is also identified with a URI.

The RDF Vocabulary Definition Language (RDFS) [15] and the Web Ontology Language (OWL) [16] provide a series of formalized vocabularies of terms that can be used to describe entities in RDF graphs. Vocabularies are themselves expressed in RDF and provide classes, properties, individuals, and data values to model the domains of interest.

The use of HTTP as a standardised data access mechanism and RDF as a data model simplifies data access compared to APIs, which rely on heterogeneous data models and ad-hoc access interfaces. In addition, applications that make use of Linked Data standard do not have to be restricted to a static set of data sources, but can discover new data at run-time by following RDF links.

3.2 Generating data from RDBMS

The majority of data in many domainsare stored in Relational Data Base (RDB), thanks to their scalability, efficiency, powerful querying capabilities, and reliability. Compared to the relational data model, RDF is more expressive, since the data represented contains information on its semantic meaning and can be interpreted and processed, and new relationship can be inferred by reasoner software.

For this and many other reasons, mapping RDB to RDF is an important aspect to fully leverage the expressivity of RDF models and incorporate domain semantics to the raw data

Moreover, data in different RDBs can be integrated, using RDF mechanisms, with information that comes from other data sources; once data are published in the Web of Data, queries can span different data sources and more powerful retrieval methods can be built, making it accessible for humans and machines [17].

We can classify the methods used to generate the mappings between RDB and RDF into two categories: Direct Mapping (or Automatic Mapping Generation) and Domain (or Semantic-driven) Mapping Generation with Transformation Rules.

3.2.1 Direct Mapping

The direct mapping defines an RDF Graph representation of the data in a relational database. The direct mapping takes as input a relational database (data and schema), and generates an RDF graph that is called the direct graph.

Typically, it is assumed that each table in a spreadsheet adheres to a relational model where each row in the table describes a different entity and each column describes an attribute for that entity [18], namely:

1. A RDB record is a RDF node
2. The column name of a RDB table is a RDF predicate
3. A RDB table cell is a value

Many systems leverage direct mappings to automatically generate mappings between RDB and RDF. This method is fast and easy to use. Unfortunately, there are numerous RDB that do not follow to this simple data model, as many data bases are extremely flexible and do not restrict the tabular structures.

Moreover, relational databases allow links between different tables that do not directly reflect the simple subject-predicate-object of RDF graphs, and allow for relationship better described by OWL ontologies, not compatible with Direct Mapping.

Even though these automatically generated mappings often do not capture complex domain semantics that are required by many applications, they can serve as a useful starting point to create more customised, domain-specific mappings.

3.2.2 Domain specific Mapping

The second mapping approach is used to generate more complex mappings from RDB to RDF, by incorporating domain semantics that is often implicit or not present at all in the original RDB schema.

This approach is called Domain Mapping and the process is the same as an ontology population technique where the transformed data are instances of the concepts defined in the ontology schema. This method allow the users to create customised mapping rules in addition to the automatically generated rules.

Many systems exists that make use of the Domain mapping techniques, applying a set of mapping definitions to transform non-RDF data sources to RDF.

The W3C standard R2RML is one language which allows for specifying the mappings needed to generate RDF datasets from relational databases [19]. R2RML mappings are themselves expressed as RDF graphs and are tailored to a specific database schema. The input to an R2RML mapping is a relational database that conforms to that schema. The output is an RDF dataset.

A different language, mainly used for spreadsheet, is Mapping Master (M^2) a domain specific language for describing OWL ontologies. M^2 allows to reference spreadsheet content in expressions: any transformation rule that indicates an

OWL class, property, individual, data type, or data value can be substituted with this reference clause [20].

4 Semantic Implementation

The CRIC data set presents many characteristics that make it a good fit for a semantic implementation: the data is vast, with a clear distinction of features and scope of its element; many of the elements are linked with each other with complex relationships and properties, such as the *providedBy* and *usedBy* concepts; different data sets are present, for experiment-specific deployment of the CRIC model that would benefit from different schema and URI based look-up.

The CRIC dataset is provided with the Django framework as an SQL based relational database [21]. Through its Web interface is possible to export single tables as *json* or *csv* files. The tables are provided with a row for each element and a column for every property in the schema; multiple attribute for a single property are separated by a comma (,).

In a possible semantic implementation, a single table define a Class in the CRIC ontology. The table columns could be data property, with a value of a defined type (i.e. string, integer), as well as object property, that links an individual of a Class to an element of a different Class. In other cases a column in the dataset could be describe an inverse relationship (such as all the Experiment Sites that are proivided by a specific Regional Center) or derived by a more complex query (such as the number of Sites in a federation).

For this reason a direct mapping of the dataset to RDF is not advisable, and a Domain Mapping approach based on Mapping Master was used as a prototype implementation. The mapping was performed by building the ontology with Protégé, an ontology development environment developed by the Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine [22], with the Cellfie plugin [23] (based on the M^2 language) used to import data from the *xlsx* file created by merging on different spreadsheet the *csv* tables exported from CRIC.

4.1 Transformation rules

M^2 transformation rules are based on the OWL Manchester Syntax [24] and loaded as json files. Cellfie extends the default language adding suport for spreadsheet and cell ranges. As an example, the following transformation rule reads the first column of the spreadsheet as a list of individual of class ExperimentSite; the folowing column are used for data and object properties; a M^2 directive is used to add the prefix Tier to the column listing the Tier number of the Site.

```
Individual: @A*
Types:  ExperimentSite
```

```
Facts:  hasState @C*,
        hasTier @F*(mm:prepend("Tier")),
        hasStatus @G*,
        hasCountry @D*,
        hasVOName @E*
```

4.2 Ontology description

Due to the heterogeneous nature of the CRIC dataset, most of its ontology is built on classes and few subclasses. Most of the properties are data properties, except for those describing relationship between different classes (i.e. providedBy and usedBy) which are object properties with flexible domains and ranges.

The following section provides a top-down description of each class and its properties, in alphabetical order.

4.2.1 ComputeUnit

Experiment-spcific class, used to group CMS Compute units. In this sample ontology is used to present an example of experiment-specific definitions. The properties present in the ontology are:

hasState Data properties of type *xsd::string* (text string) with the state (active or inactive) of the unit;

providedBy Object property that link the Compute unit to the CMSSite that hosts it.

4.2.2 Country

This class is used as range for the hasCountry property of Sites and Federations. It is implemented as a class instead of a simple string to allow for a connection to common ontologies, such as *dbpedia* or *schema.org*.

This is done with the RDF graph:

```
dbpedia:Country owl:equivalentClass cric:Country
```

In this sample ontolgy is used to present an example of connection to an external dataset.

4.2.3 DDMEndpoint

Experiment-specific class, used to describe the Distributed Data Management endpoints – the storage units for the ATLAS experiment. In this sample ontology is presented to show the usage of the *usedBy* property. The properties present in the ontology are:

hasState Data properties of type *xsd::string* (text string) with the state (active or inactive) of the unit;

hasType Data properties of type *xsd::string* that describes the typology of storage (i.e. disk, tape);

providedBy Object property that link the Storage unit to the ATLASSite that hosts it;

usedBy Object property that link the Storage unit to a Resource class.

4.2.4 Facility

Experiment-spcific class, used to group CMS Sites. The property present in the ontology are:

hasState Data properties of type *xsd::string* (text string) with the state (active or inactive) of the facility;

4.2.5 Federation

Core class, used to represent federations as defined by WLCG. The properties present in the ontology are:

hasCountry Object properties that links the class with a Country object;

hasTier Object property to specify the Tier level of the federation.

4.2.6 Resource

Core class, used to define a Resource. In the dataset is used to share attributes between Storage and Compute unites that use the same service protocol The properties present in the ontology are:

providedBy Object property that link the Resource to the Site that hosts it. Multiple instances of the properties can exists for the same resource, linking it to different sites.

4.2.7 Service

Core class, used to define any kind of service of the WLCG infrastructure. The properties present in the ontology are:

hasType Data properties of type *xsd::string* that describes the type of service (i.e. disk, tape);

hasFlavour Genric Data properties of type *xsd::string* that describes attributes of the service (can be Null);

hasState Data properties of type *xsd::string* (text string) with the state (active or inactive) of the service;

providedBy Object property that link the Service to the Site that hosts it.

4.2.8 Site

Primitive class, used to as archetipe for the different type of Sites in the ontology. The properties present in the ontology are common to the subClasses:

hasState Data properties of type *xsd::string* (text string) with the state (active or inactive) of the site;

hasCountry Object properties that links the class with a Country object;

hasFederation Object properties that links the class with a Federation object.

The class has two subclasses: ExperimentSite and RCSite.

4.2.9 ExperimentSite

This is an abstract Class used to define the attributes of the object of its subclasses: CMSSite and ATLASSite. The properties present in the ontology are:

hasTier Object property to specify the Tier level of the site;

hasVOName Data properties that specify to which experiment the site belongs to;

Based on the VOName, individuals of the ExperimentSite class are inferred to be members of the ATLASSite or CMSSite subclasses.

4.2.10 RCSite

Core class used to define Regional Center sites. In the data set the individual of this class are populated by a specific database, but the experiments can define other Sites. This is the reason for the separation from the Experiment Site class.

4.2.11 StorageUnit

Experiment-spcific class, used to group CMS Storage units. The properties present in the ontology are:

hasState Data properties of type *xsd::string* (text string) with the state (active or inactive) of the unit;

providedBy Object property that link the Storage unit to the CMSSite that hosts it.

hasType Data properties of type *xsd::string* that describes the typology of storage (i.e. disk, tape);

4.2.12 Tier

Defined class, used to list the possible Tier a Site can be (Tier 0, Tier 1, Tier 2 or Tier 3). Used in place of a Data property, to limit the range of values the property can assume.

4.3 Queries

SPARQL (Simple Protocol and RDF Query Language) is a query language for RDF based on SQL [25]. SPARQL finds query results through pattern matching within the graph that must satisfy what is specified in the WHERE clause of the query.

Classes list
The following query selectis from the database all the item with type Class. The *type* property is described in the *rdf* ontology, while the Class entity is described in the *owl* ontology. The query syntax includes a series of ontologies by using prefixes.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX cric: <http://www.semanticweb.org/rossitisbeni/CRIC#>
PREFIX dbpedia: <http://dbpedia.org/resource/>

SELECT ?x
WHERE {
    ?x rdf:type owl:Class .
}
```

In the following queries the PREFIX commands are omitted for the sake of simplicity.

Sites list
This query lists all sites in the dataset. With the keyword *OPTIONAL* it allows to show attributes that are not present in every instance of the Class.

```
SELECT ?site ?Country ?Tier ?VOName ?State ?Status ?Facility
WHERE {
    ?site   a cric:Site ;
            cric:hasCountry ?Country ;
            cric:hasState ?State ;
            cric:hasStatus ?Status ;
            cric:hasTier ?Tier .
    OPTIONAL {  ?site cric:hasFacility ?Facility . }
    OPTIONAL {  ?site cric:hasVOName ?VOName . }
}
```

Count sites per federation
Standard SQL operations are possible. In this example for every federation instance, it hows the amount of sites belonging to that specific federation.

```
SELECT ?federation ?AccountingName ?Country (COUNT (?site) AS ?Sites)
WHERE {
    ?site   a cric:Site ;
```

```
            cric:hasFederation ?federation .
    ?federation cric:hasAccountingName ?AccountingName ;
                cric:hasCountry ?Country .
}
GROUP BY ?federation ?AccountingName ?Country
```

5 Conclusion

In the past years, CRIC has been evolved to becom the central topology system for WLCG. It has been successfully used by many services and is development has been focused toward making it more dynamic and heterogeneous.

Its dynamic structure makes it a good candidate for it to follow the Linked data principles. The data already present different schemas based on the domain specific use of resources; it is used to connect different datasets under the same system and allows developer to access it via Web UI and APIs. All these functionalities can be satisfied with the tools provided by the Semantic Web stack.

The RDF implementation of the CRIC dataset shown in these page is a prototype of the possible semantic definition for the resources present in the database. The project was limited in scope, with the purpose of showing a possible ontology, and its use for querying the system.

A more realistic approach to the problem would be to use a system capable of directly accessing the RDB hosting the CRIC data and, through the use of ad hoc R2RML mapping, expose a SPARQL endpoint for querying the data.

A possible solution could make use of a custom deployment of the Ontop Virtual Knowledge Graph System [26]; the software exposes the content of a relational databases as a knowledge graph. This graph is virtual, with the data remaining in the source instead of being moved to another database.

Acknowledgements

This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

References

[1] *CERN - Accellerating Science.* URL: http://wlcg.web.cern.ch/.

[2] *EGI Advanced Computing Services for Research.* URL: https://www.egi.eu/.

[3] *Open Science Grid.* URL: https://opensciencegrid.org/.

[4] *The MONARC Project.* URL: https://monarc.web.cern.ch/MONARC/.

[5] *The HL-LHC Project.* Apr. 2019. URL: http://hilumilhc.web.cern.ch/.

[6] The HEP Software Foundation et al. “A Roadmap for HEP Software and Computing R&D for the 2020s”. In: *Computing and Software for Big Science* 3.1 (Mar. 2019), p. 7. ISSN: 2510-2044. DOI: 10.1007/s41781-018-0018-8.

[7] Anisenkov, Alexey et al. “CRIC: a unified information system for WLCG and beyond”. In: *EPJ Web Conf.* 214 (2019), p. 03003. DOI: 10.1051/epjconf/201921403003.

[8] Alexey Anisenkov, Alessandro Di Girolamo, and Maria Alandes Pradillo. “AGIS: Integration of new technologies used in ATLAS Distributed Computing”. In: *Journal of Physics: Conference Series* 898 (Oct. 2017). DOI: 10.1088/1742-6596/898/9/092023.

[9] *CRIC documentation.* URL: https://core-cric-docs.web.cern.ch/core-cric-docs/latest/.

[10] Dost, Jeffrey et al. “Exploiting CRIC to streamline the configuration management of GlideinWMS factories for CMS support”. In: *EPJ Web Conf.* 245 (2020), p. 03023. DOI: 10.1051/epjconf/202024503023.

[11] *CMS CRIC documentation.* URL: https://cms-cric-docs.web.cern.ch/cms-cric-docs/latest/cms/index.html.

[12] Christian Bizer, Tom Heath, and Tim Berners-Lee. “Linked Data - The Story So Far”. In: *International Journal on Semantic Web and Information Systems* 5.3 (July 2009), pp. 1–22. DOI: 10.4018/jswis.2009081901.

[13] *Uniform Resource Identifier (URI): Generic Syntax.* URL: https://www.ietf.org/rfc/rfc3986.txt.

[14] Richard Cyganiak, David Wood, and Markus Lanthaler. *RDF 1.1 Concepts and Abstract Syntax.* 2014. URL: https://www.w3.org/TR/rdf11-concepts/.

[15] Dan Brickley and R.V. Guha. *RDF Schema 1.1.* 2014. URL: https://www.w3.org/TR/rdf-schema/.

[16] W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview (Second Edition).* 2012. URL: https://www.w3.org/TR/owl2-overview/.

[17] Sören Auer et al. *Use Cases and Requirements for Mapping Relational Databases to RDF.* 2010. URL: https://www.w3.org/TR/rdb2rdf-ucr/.

[18] Satya Sahoo et al. “A Survey of Current Approaches for Mapping of Relational Databases to RDF”. In: *W3C* (Jan. 2009). URL: https://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport.pdf.

[19] Souripriya Das, Seema Sundara, and Richard Cyganiak. *R2RML: RDB to RDF Mapping Language.* 2012. URL: https://www.w3.org/TR/r2rml/.

[20] Martin J. O'Connor, Christian Halaschek-Wiener, and Mark A. Musen. "Mapping Master: A Flexible Approach for Mapping Spreadsheets to OWL". In: *The Semantic Web – ISWC 2010.* Ed. by Peter F. Patel-Schneider et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. ISBN: 978-3-642-17749-1.

[21] Django Software Foundation. *Django Databases.* 2019. URL: `https://docs.djangoproject.com/en/4.1/ref/databases/`.

[22] Mark A Musen and Protégé Team. "The Protégé Project: A Look Back and a Look Forward". en. In: *AI Matters* 1.4 (June 2015), pp. 4–12. DOI: `10.1145/2757001.2757003`.

[23] Protégé Developers. *Cellfie plugin.* URL: `https://github.com/protegeproject/cellfie-plugin`.

[24] Matthew Horridge and Peter F. Patel-Schneider. *OWL 2 Web Ontology Language Manchester Syntax (Second Edition).* URL: `https://www.w3.org/TR/owl2-manchester-syntax/`.

[25] W3C SPARQL Working Group. *SPARQL 1.1 Overview.* URL: `https://www.w3.org/TR/sparql11-overview/`.

[26] Guohui Xiao et al. "The Virtual Knowledge Graph System Ontop". In: *The Semantic Web – ISWC 2020.* Ed. by Jeff Z. Pan et al. Cham: Springer International Publishing, 2020, pp. 259–277. ISBN: 978-3-030-62466-8.