Eurostars Project

3DFed – Dynamic Data Distribution and Query Federation

Project Number: E!114681 **Start Date of Project:** 2021/04/01 **Duration:** 36 months

Deliverable 2.1
Report on Data Storage Profile Generation

Dissemination Level	Public
Due Date of Deliverable	September 30, 2023
Actual Submission Date	September 30, 2023
Work Package	WP2, Data Storage Monitoring and Profiling
Deliverable	D2.1
Type	Report
Approval Status	Final
Version	1.0
Number of Pages	17

Abstract:
Deliverable D2.1 (Report on Data Storage Profile Generation) has the goal of describing the results from our activities on T2.1, i.e. the development and deployment of a profile generation module into our SPARQL endpoint monitoring platform, which was developed in T2.2 and described in D2.2. This profile generation module has the purpose of generating useful metadata profiles (VoID, SPARQL Service Description, dataset coherence and relationship specialty) of the monitored SPARQL endpoints, i.e. the monitored data storage solutions. The purpose of the profiles is to provide metadata which is useful for source selection, cardinality estimation, and query planning by SPARQL federation-engine developers, especially in the context of the 3DFed project.

History

Version	Date	Activity	Author
0.1	01/09/2023	Initial Draft	Milos Jovanovik
0.2	14/09/2023	Extended Draft	Milos Jovanovik
0.3	21/09/2023	Input on Draft	Mirko Spasić
0.4	26/09/2023	Updated Draft	Milos Jovanovik, Mirko Spasić
0.5	27/09/2023	Issued for review	Milos Jovanovik
0.6	28/09/2023	Review	Muhammad Saleem
1.0	30/09/2023	Final approval and submission	Milos Jovanovik, Mirko Spasić

Author List

Organization	Name	Contact Information
OpenLink Software	Milos Jovanovik	mjovanovik@openlinksw.com
OpenLink Software	Mirko Spasić	mspasic@openlinksw.com
OpenLink Software	Hugh Williams	hwilliams@openlinksw.com

Contents

1 Introduction **3**

2 Profile Generation **3**

2.1 VoID Profiles . . . 3

2.2 SD Profiles . . . 5

2.3 Coherence . . . 5

2.4 Relationship Specialty . . . 7

3 Updated SPARQL Endpoint Monitoring Platform **7**

3.1 Calculation Task . . . 7

3.2 The Profiles Page . . . 9

3.3 The Endpoint View Page . . . 11

3.4 The Main Page . . . 11

3.5 REST API . . . 14

3.6 Other Improvements . . . 16

4 Conclusion **16**

1 Introduction

In this deliverable we outline the development performed as part of task T2.1. The main goal of T2.1 is to develop an automatic, scalable tool to generate data storage profiles on-the-fly. This tool is intended to work with a set of URLs of SPARQL endpoints (RDF data storage solutions) and automatically generate a metadata profile for each endpoint as an RDF dataset. Such a tool is extremely helpful in a situation where the majority of SPARQL endpoints lack a predefined metadata profile, which is available through the endpoint itself.

As we reported in D2.2 [5], our SPARQL endpoint monitoring platform[1] has detected that over 98% of the 581 monitored endpoints do not provide a predefined VoID metadata profile, and over 94% of them lack SPARQL Service Description (SD) metadata. Given that this metadata is necessary for multiple tasks performed by SPARQL federation engines, such as source selection, cardinality estimation, query planning, etc., especially in the context of the 3DFed project, automatic profile generation on-the-fly was a necessity.

As part of our work on T2.1, we successfully developed the tool, and integrated it into the SPARQL endpoint monitoring platform. With it, we are able to generate a metadata profile for each endpoint, which contains its VoID metadata, SD metadata, the coherence of the datasets in the endpoint, and the relationship specialty (RS) of the data. The information used for the construction of the profiles is collected via SPARQL queries, issued directly to the SPARQL endpoint in question. The resulting profile dataset is then made publicly available via the SPARQL endpoint monitoring platform.

Additionally, the SPARQL endpoint monitoring platform provides statistics for all SPARQL endpoints it monitors regarding their generated profiles, more specifically the VoID profiles, the SD profiles, and the values for coherence and RS. The platform is also open-source and reusable.

2 Profile Generation

Generating a profile for a publicly available SPARQL endpoint can only be done by querying the endpoint itself for the necessary metadata. Given that the main intention of metadata profiles, such as the Vocabulary of Interlinked Datasets (VoID) [1] and the SPARQL Service Description (SD) [10], is for them to be defined by the data publishers themselves, they are not actually intended to be (re)constructed by third-parties. This is due to some of the metadata being based on information only available to the publisher, and not the consumers of the datasets.

In the following subsections we will elaborate on our approach to construct the VoID and SD profiles from the data available from each endpoint, along with our calculations of dataset coherence and relationship specialty, as additional data included in our generated endpoint profiles.

2.1 VoID Profiles

As previously discussed, VoID represents a vocabulary for expressing metadata about RDF datasets [1]. Its purpose is to establish a connection between those who create RDF data and those who use it, enabling various applications such as data exploration, dataset cataloging, and archival activities. Given that its original intent is to be created and provided by data creators and publishers, only part of it can be used to create a VoID metadata profile post-festum, based on the data available from a given SPARQL endpoint.

Therefore, in our automatically generated VoID profiles for the SPARQL endpoints, we use 2 VoID classes, 9 VoID properties, 3 DCTerms properties [2] and 1 FOAF property [3]. An example generated VoID profile for

[1] 3DFed SPARQL endpoint monitoring platform: `https://sparqles.demo.openlinksw.com/`

the DBpedia SPARQL endpoint is given in Listing 1.

Listing 1: Example of a Generated VoID Profile for a SPARQL Endpoint

```
@prefix void: <http://rdfs.org/ns/void#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<http://dbpedia.org/sparql/profile> a void:DatasetDescription ;
    dcterms:title "Automatically constructed VoID description for a
       SPARQL Endpoint" ;
    dcterms:creator <https://sparqles.demo.openlinksw.com/> ;
    dcterms:date "2023-09-02"^^xsd:date ;
    foaf:primaryTopic <http://dbpedia.org/sparql> .

<http://dbpedia.org/sparql> a void:Dataset ;
    void:sparqlEndpoint <http://dbpedia.org/sparql> ;
    void:exampleResource <http://dbpedia.org/resource/London> ;
    void:exampleResource <http://dbpedia.org/resource/Skopje> ;
    void:exampleResource <http://dbpedia.org/resource/Belgrade> ;
    void:vocabulary <http://purl.org/dc/terms/> ;
    void:vocabulary <http://xmlns.com/foaf/0.1/> ;
    void:triples 1153000000 ;
    void:entities 2000000 ;
    void:classes 16 ;
    void:properties 31 ;
    void:distinctSubjects 2500000 ;
    void:distinctObjects 3100000 .
```

So, how is this profile constructed? Beforehand, we only know the URL of the SPARQL endpoint, which in case of the example from Listing 1 is `http://dbpedia.org/sparql`. From there, we construct the VoID profile by issuing SPARQL queries to the endpoint, processing the responses and creating the RDF graph for the profile. More specifically, we send separate queries to the endpoint to select the total number of RDF triples, the total number of entities, the total number of classes and properties, the number of distinct subjects, and the number of distinct objects. We also send a query to select three random entities from the endpoint, as example resources available through it. We map the results to the corresponding properties of the VoID profile, and construct the profile as an RDF graph using the Jena framework[2].

Missing VoID profile. If one of these SPARQL queries cannot be successfully executed on a SPARQL endpoint due to the endpoint being unavailable, we abort the process of constructing the profile. This is done to save time, because since the endpoint is unavailable, none of the queries would be executed anyway, and we wouldn't be able to construct the profile. It is important to note that some of the queries are long-running for very large datasets available through a SPARQL endpoint. This is especially the case for selecting the total number of triples, subjects, properties and objects.

Partial VoID profile. If, however, a SPARQL query cannot be successfully executed due to a timeout, we accept that we cannot get that value for the endpoint. But, we still continue with the construction of the profile in this case, and continue executing the other SPARQL queries for the other values. If the other queries are successful, we construct a partial profile.

The SPARQL queries used in the construction of the VoID profile, as well as the code for constructing it as an RDF graph, are available as part of the code of the SPARQL monitoring platform, publicly available on GitHub[3].

[2] `https://jena.apache.org`
[3] `https://github.com/OpenLinkSoftware/sparqles`

2.2 SD Profiles

Similarly to our approach with the VoID profiles, we construct the SPARQL Service Description (SD) profiles only with the data available through the endpoints themselves. We use 3 classes from the SD vocabulary, 3 properties from the SD vocabulary and 1 property from the VoID vocabulary. An example of a generated SD profile for the DBpedia SPARQL endpoint is given in Listing 2.

Listing 2: Example of a Generated SD Profile for a SPARQL Endpoint

```
@prefix sd: <http://www.w3.org/ns/sparql-service-description#> .
@prefix void: <http://rdfs.org/ns/void#> .

<http://dbpedia.org/sparql> a sd:Service ;
    sd:endpoint <http://dbpedia.org/sparql> ;
    sd:defaultDataset [
        a sd:Dataset ;
        sd:defaultGraph [
            a sd:Graph ;
            void:triples 1153000000
        ]
    ] .
```

The SD profiles are a bit easier to construct than VoID profiles, because we only need one piece of information from the endpoint in order to construct it; we only need the total number of RDF triples in the endpoint. But, we already select that information for the construction of the VoID profile, meaning we already have all the data we need in order to generate the RDF graph of the SD profile for the SPARQL endpoint in question. We construct the correct format of the SD profile using the Jena framework, as well.

Partial SD profile. Similarly as with the VoID profiles, if we cannot get the information about the total number of RDF triples from a SPARQL endpoint, we construct a partial SD profile for it.

The code for the construction of the SD profiles is also part of the SPARQL monitoring platform, and the code is publicly available on GitHub, as previously stated.

2.3 Coherence

Coherence, or *dataset coherence*, to be more precise, is a measure introduced in [4]. It measures the structuredness of a dataset in values in the range of $[0, 1]$, expressing the amount of data / values present in the dataset, per available property. This means that the structuredness of a dataset can range between 0 (a dataset where no properties have any values, for any of the entities they are related to) and 1 (a dataset where all available properties for each entity in the dataset have explicit data / values defined for them).

More rigorously, the structuredness of the dataset D with respect to the type system $\mathcal{T}$, is expressed as follows:

$$CH(\mathcal{T}, D) = \sum_{T \in \mathcal{T}} WT(CV(T, D)) \cdot CV(T, D)$$

This is the weighted sum of the coverage $CV(T, D)$ of individual types $T \in \mathcal{T}$, where the weight coefficient $WT(CV(T, D))$ depends on the number of properties for a type T, the number of entities in dataset D of type T, and their share in the totality of the dataset D among the other types. Its rationale is to give higher impact to types with more instances and properties. $CV(T, D)$ represents the coverage of type T on the dataset D. It depends on whether the instances of the type T set a value for all its properties. If that is the case for all the

instances, the coverage will be 1 (perfect structuredness), otherwise it will take a value from $[0, 1)$ [8].

This measure was introduced in [4] in order to calculate and conclude that there is a clear distinction in the structuredness, i.e. the coherence between the datasets used in RDF benchmarks and real-world RDF datasets. The authors come to the conclusion that most of the datasets used for benchmarking have a coherence value near 1 (or exactly 1), while real-world datasets are below or around 0.6. This shows a clear discrepancy between datasets used for benchmarking and datasets used in real-world applications. These implications mean that this measure is very important, especially in scenarios such as ours in the 3DFed project, where an algorithm has to conclude which dataset or endpoint will be used for a specific distributed SPARQL query.

Coherence of a SPARQL Endpoint. In our case, we don't measure the coherence of different datasets within a SPARQL endpoint, but we measure the coherence of the entire content available by default from the SPARQL endpoint. This means that our queries used for calculating the coherence do not use specific named graphs, but rather target the default graph, which in general means all RDF data available through the endpoint. Therefore we use the term *coherence*, instead of *dataset coherence*.

Similarly to the previous data needed for constructing the profiles, we send a set of SPARQL queries for a given SPARQL endpoint, in order to select all data necessary for calculating the coherence of the endpoint. If one of the queries fails, we abort the calculation. If all calculations pass successfully, we calculate the coherence as a value in the range $[0, 1]$, and include it in the VoID profile of the endpoint. An example VoID profile in RDF, which includes the coherence metric for the endpoint, is shown in Listing 3. The code for the calculations is publicly available as part of the project on GitHub.

Listing 3: Example of a Generated VoID Profile for a SPARQL Endpoint with Coherence and RS Values

```
@prefix void: <http://rdfs.org/ns/void#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dddfed: <https://www.3dfed.com/ontology/> .

<http://vocabulary.semantic-web.at/PoolParty/sparql/OpenData/profile>
   a void:DatasetDescription ;
        dcterms:creator <https://sparqles.demo.openlinksw.com> ;
        dcterms:date "2023-09-24" ;
        dcterms:title "Automatically constructed VoID description for
           a SPARQL Endpoint" ;
        foaf:primaryTopic <http://vocabulary.semantic-web.at/PoolParty
           /sparql/OpenData> .

<http://vocabulary.semantic-web.at/PoolParty/sparql/OpenData> a void:
   Dataset ;
        void:classes "88" ;
        void:distinctObjects "1357" ;
        void:distinctSubjects "311" ;
        void:entities "309" ;
        void:exampleResource <1a1b4318:18a538349a3:-330> ,
                <1a1b4318:18a538349a3:-32f> ,
                <1a1b4318:18a538349a3:-331> ;
        void:properties "80" ;
        void:sparqlEndpoint <http://vocabulary.semantic-web.at/
           PoolParty/sparql/OpenData> ;
        void:triples "3456" ;
        dddfed:coherence "0.9051679485692863" ;
        dddfed:relationshipSpecialty "18.590026915709394" .
```

2.4 Relationship Specialty

Within datasets, some attributes are more common and associated with multiple resources. Additionally, some attributes are multi-valued, e.g., a person can have more than one phone number or professional skill. The number of occurrences of a predicate associated with each resource in the dataset provides useful information on the graph structure of an RDF dataset, and makes some resources distinguishable from others [6]. In real-world datasets, this kind of relationship specialty (RS) is commonplace. For example, several million people can like the same movie. Likewise, a research paper can be cited in several hundred other publications. The authors in [6] suggest that synthetic datasets are limited in how they reflect this relationship specialty. This is either due to the simulation of uniform relationship patterns for all resources, or a random relationship generation process.

Therefore, similar to the coherence measure, this metric becomes important for algorithms which need to determine the structure of the data available via a given SPARQL endpoint, to determine the target for a specific distributed SPARQL query.

Relationship Specialty of a SPARQL Endpoint. The calculation for the relationship specialty of an RDF dataset is given in [7]. It first calculates the predicate RS, which is then used to calculate the overall dataset RS value. Similarly to the coherence metric, we calculate the relationship specialty metric for the default dataset of the SPARQL endpoint in question, which usually means the entire set of RDF data available through the endpoint.

The calculations for the RS value are done using several SPARQL queries for a given SPARQL endpoint, which select the data necessary for calculating it. If one of the queries fails, we abort the calculation. If all calculations pass successfully, we calculate the relationship specialty, and include it in the VoID profile of the endpoint. An example VoID profile in RDF, which includes the relationship specialty metric for the endpoint, is shown in Listing 3. The code for the calculations is publicly available as part of the project on GitHub.

3 Updated SPARQL Endpoint Monitoring Platform

Our SPARQL endpoint monitoring platform, which was deployed on our server [5], was upgraded a month ago. It was extended to include the profile generation module described in Section 2.

The required algorithms are executed for each endpoint once a week, starting on Sundays at 23:30. The algorithms are implemented within the newly added task type described in Section 3.1. From a user perspective, the monitoring platform has been improved by adding a new page providing an overview of the newly generated information about the endpoints, i.e., the Profiles page, which is described in more detail in Section 3.2. In addition, a corresponding section has been added to the pages for each individual endpoint, described in Section 3.3. On the main platform page, two new diagrams have been added, which are explained in Section 3.4. Section 3.5 explains the new and updated REST APIs that are supported by the platform and can be used by third parties to access the relevant data. Finally, Section 3.6 contains a couple of other improvements.

3.1 Calculation Task

With the aim of integrating the algorithms required for profile generation, it was necessary to change the backend of the monitoring platform. Previously, there were four basic task types that the platform performed during the endpoint monitoring: *Availability*, *Performance*, *Interoperability*, and *Discoverability* [9], analyzing the ratio of time that a given endpoint is responsive via the SPARQL protocol, performance of streaming, lookups, and joins, the supported SPARQL features, and descriptions of endpoints provided by themselves, respectively.

The platform code has now been expanded to include a new one, i.e., the fifth task type called *Calculation*.

Its goal is to generate VoID and SD metadata for a given endpoint by executing a series of SPARQL queries, and calculate the coherence and RS values of the datasets that the endpoint contains. For this type of task, it was necessary to implement the main class `CTask`, which extends the base class `EndpointTask` and overrides the `process()` method accordingly. Furthermore, it was necessary to implement the class `CAnalyzer`, which inherits the base class `Analytics` and additionally analyzes and systematizes the results of the `process()` method within its own `analyze()` method.

This new task type can be executed in two ways:

Manually: by invoking it from the command line using the predefined options:

```
sh bin/sparqles -p src/main/resources/sparqles_docker.properties -run ctask
```

Automatically: by defining special `cron` jobs that are repeated at predefined intervals. The default schedule for this task type specifies one execution per week, during the night between Sunday and Monday.

The results of the processing and analysis tasks for all available endpoints are stored in MongoDB in `ctasks` and `ctasks_agg` collections, respectively. A record from the first collection contains information about the relevant endpoint (URI and datasets), the exact times the task was started and finished, the number of triples, entities, classes, properties, distinct subjects and objects, examples of resources, the coherence and RS values, while an example of a record is shown in Listing 4. The corresponding record from the second collection for this endpoint contains flags when VoID and SD metadata can be successfully, partially, or not at all generated based on the corresponding record from `ctasks`, but also VoID and SD metadata themselves in the form of RDF datasets.

Listing 4: Example of a record in the MongoDB collection `ctasks`


```
{
    "_id" : ObjectId("64ddf118e4b0b3e8519b23a6"),
    "endpointResult" : {
        "endpoint" : {
            "uri" : "http://dbtune.org/classical/sparql/",
            "datasets" : [
                {
                    "uri" : "http://dbtune.org/classical/sparql/",
                    "label" : "DBTune.org/classical"
                }
            ]
        },
        "start" : NumberLong("1694993533671"),
        "end" : NumberLong("1694993591019")
    },
    "triples" : 419519,
    "entities" : 75625,
    "classes" : 14,
    "properties" : 38,
    "distinctSubjects" : 75625,
    "distinctObjects" : 143068,
    "exampleResources" : [
        "http://dbpedia.org/resource/Alexander_Uninsky",
        "http://dbpedia.org/resource/Anton_Rubinstein",
        "http://dbpedia.org/resource/Antonio_Pappano"
    ],
    "coherence" : 0.31196353889616746,
    "RS" : 961.3786450324915
}
```

3.2 The Profiles Page

In addition to the existing main page and four pages for four different task types, a page dedicated to the generation of VoID and SD metadata and the computation of the coherence and RS values, i.e., a page dedicated to the Calculation task has been added to the front-end of the platform. A link to this page has been added to the header of all other pages for easier navigation. Figure 1 shows screenshots of parts of this page, for simplicity not containing all 581 endpoints.

A brief description and analysis of this page is given below:

- At the very top of the page, the date and time when the last Calculation task was completed is displayed, along with a brief description of this type of task.
- Directly below that, there is a brief statistic about the last execution of this type of task against all listed endpoints:

  VoID: It can be seen in Figure 1 that VoID and SD metadata could only be generated for 67 out of 581 endpoints. The main reason for this is that the vast majority of endpoints are not available at all. Of these 67 endpoints, VoID metadata was fully generated for 41, i.e., it contains all the required information according to the specification, while VoID metadata was partially generated for 26, i.e., some information that should be included in the description could not be calculated. For example, for a given endpoint, the total number of triples, entities and classes is known, while the total number of distinct properties, subjects and/or objects could not be computed, as the relevant SPARQL query is more demanding and cannot be answered before a timeout from the endpoint. The reason for this is the limited execution time of SPARQL queries, specified within the endpoints themselves, which is sometimes too short for data-heavy endpoints.

  SD: The situation is slightly better for the SD metadata, as the relevant SPARQL queries are simpler, resulting that this metadata was fully generated for 62 endpoints and partially generated for only 5 endpoints (Figure 1).

  Coherence and RS values: These values were computed for 36 endpoints (Figure 1), which is to be expected since the number and complexity of queries that need to be performed against a single endpoint in order to compute these values is much larger than while computing VoID and SD metadata. The range of coherence values is from 0.24 (DBnary endpoint[4]) to 1.0 (Lotico endpoint[5]), while the values of RS range from 5.59 (Geological Survey of Austria GBA - Thesaurus endpoint[6]) to over one million (Allie Abbreviation And Long Form Database in Life Science endpoint[7]).

- The main part of this page shows a complete list of all monitored endpoints. For each endpoint it indicates whether VoID and SD metadata was calculated fully, partially, or not at all, using green, orange or gray circles, respectively (as indicated in a legend on the Profiles page), as well as the specific values for coherence and RS. If any of these values were not successfully computed, the word `missing` is shown. The list of endpoints can be sorted by a single click on one of the four values presented here. Clicking on the label of a SPARQL endpoint within this list opens a new page dedicated exclusively to that particular endpoint, which is explained in Section 3.3.

[4] `http://kaiko.getalp.org/sparql`
[5] `http://www.lotico.com:3030/lotico/sparql`
[6] `http://resource.geolba.ac.at/PoolParty/sparql/GeologicTimeScale`
[7] `http://data.allie.dbcls.jp/sparql`

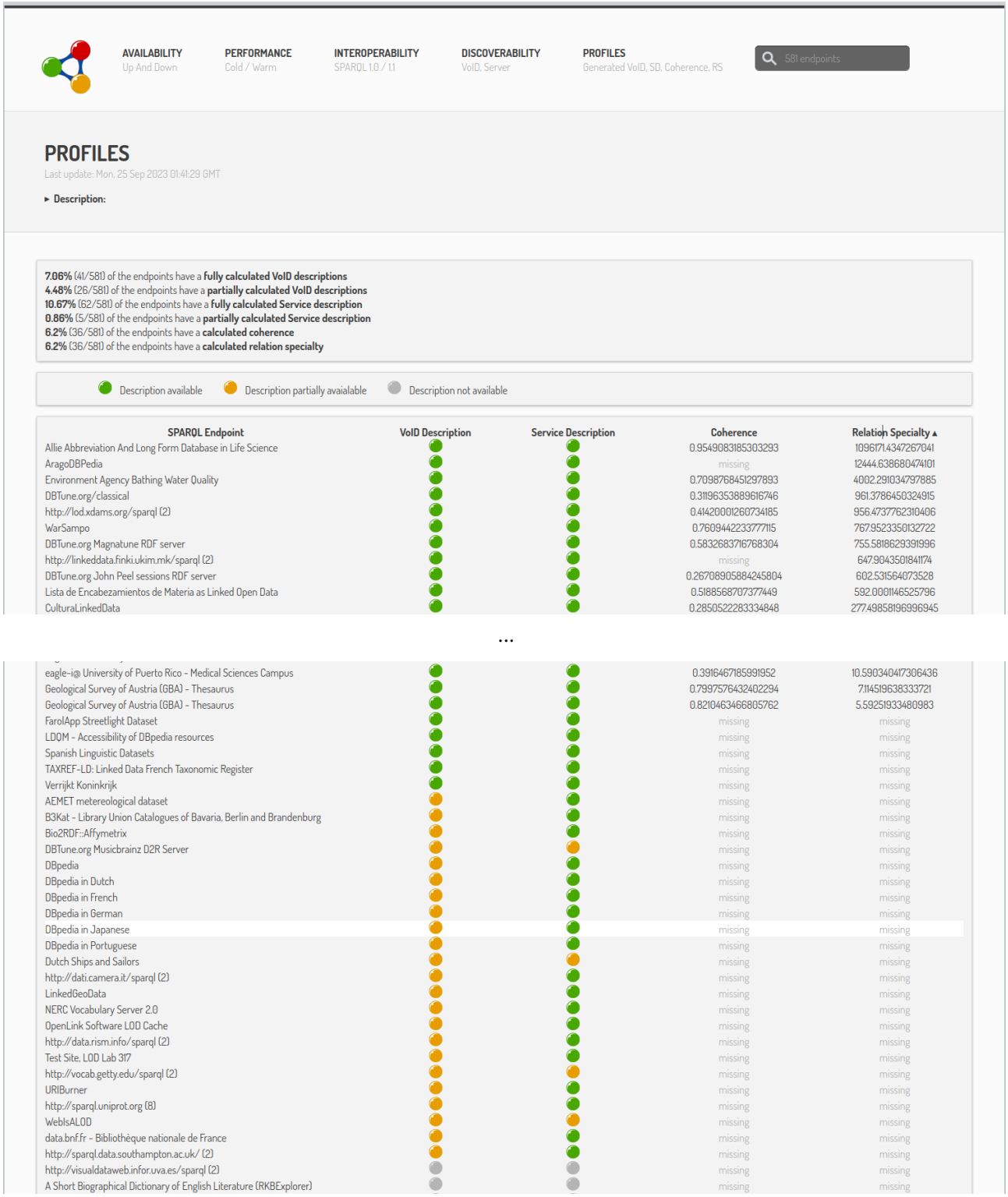

Figure 1: The Profiles Page within the 3DFed SPARQL Endpoint Monitoring Service.

3.3 The Endpoint View Page

For each individual endpoint, there is a separate page that contains information about all the task types against that endpoint. Since there were 4 different task types, this page had 4 sections: Availability, Performance, Interoperability and Discoverability. With the addition of the fifth task type, i.e., Calculation task, a new Profiles section was added to this page where one can see the key statistics of the dataset within the given endpoint:

- the total number of triples,
- the total number of entities,
- the total number of classes,
- the total number of distinct subjects,
- the total number of distinct objects,
- three examples of resources in the dataset,
- the coherence value, and
- the RS value.

In addition, this section contains indicators regarding the generation of VoID and SD metadata in the form of green (fully generated), orange (partially generated), and gray (not generated at all) circles, as well as the ability to download the generated endpoint profile in RDF format. An example of such a page for the endpoint referenced by the MongoDB record in Listing 4 is shown in Figure 2. The newly added Profiles section is located at the bottom of the figure.

3.4 The Main Page

In addition to links to the pages with full results for all task types supported by the platform in the page header, the time of the last completed task, and a brief description of the platform itself, the main page contained four diagrams for each supported task type. By adding a new task type supported by the platform, it was natural to add a new diagram to the main page referring to it. For purely cosmetic reasons[8], instead of one diagram, two new ones were added for this purpose. One refers to VoID metadata and the distribution of coherence values, the other to SD metadata and the distribution of RS values. A screenshot of this page is shown on Figure 3.

From the first diagram, it can be seen that VoID metadata was calculated for only 11.53% of all endpoints (including 0.86% for which it was partially calculated), and the coherence values were calculated for 6.2% of the total number of endpoints. Within these successfully processed endpoints, there are most endpoints whose coherence values were in the range [0.25, 0.5[(19 endpoints or 52.78%), followed by [0.75, 0.95[(8 endpoints or 22.22%), [0.5, 0.75[(6 endpoints or 16.67%), [0.95, 1.00] (2 endpoints or 5.56%), and only one endpoint or 2.78% had a coherence of less than 0.25.

The second diagram, related to SD metadata and the distribution of RS values, has a similar structure: SD metadata were calculated for only 11.53% of all endpoints (including 4.48% for which it was partially calculated), and RS values were calculated for 6.2% of the total number of endpoints. Of these, the most numerous are the endpoints whose RS values were in the range [10, 100[(18 endpoints or 50.00%), followed by [100, 1000[(13 endpoints or 36.11%), [0, 10[and more than 10000 (2 endpoints or 5.56%), and only one endpoint or 2.78% had the RS value between 1000 and 10000.

[8]When displaying the main page on normal computer screens, two diagrams are displayed in each row.

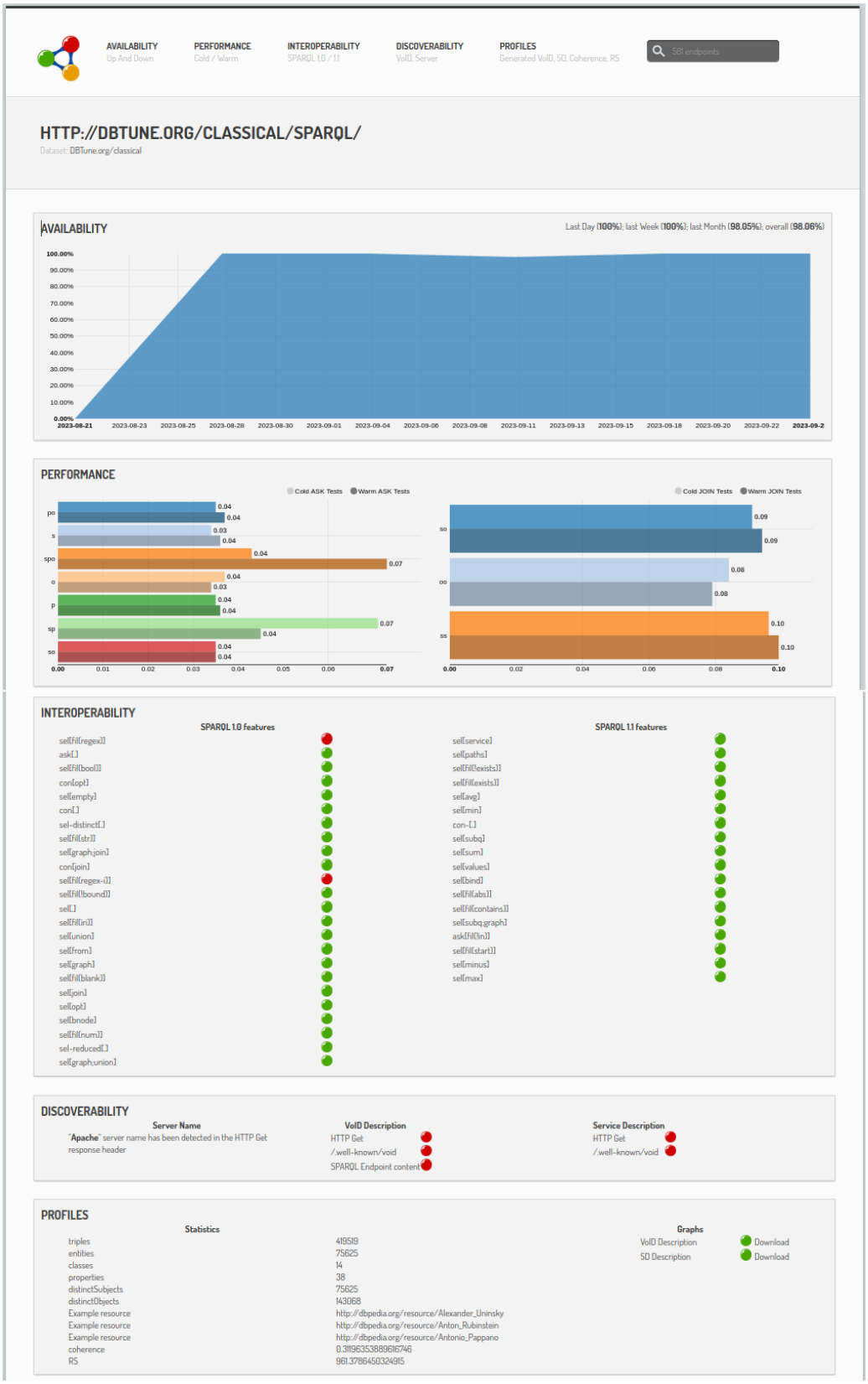

Figure 2: The Endpoint View Page within the 3DFed SPARQL Endpoint Monitoring Service.

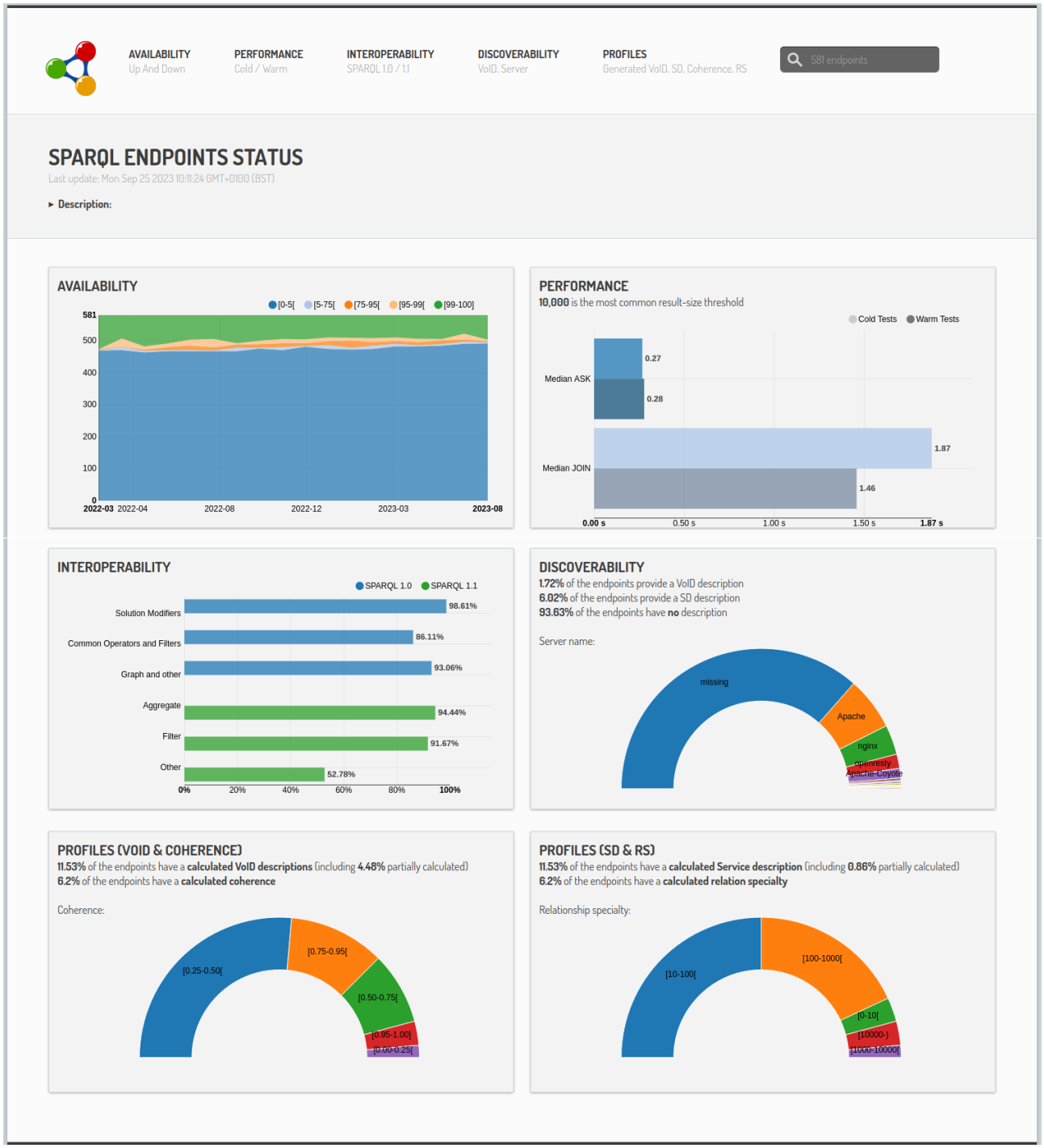

Figure 3: The Main Page of the 3DFed SPARQL Endpoint Monitoring Service.

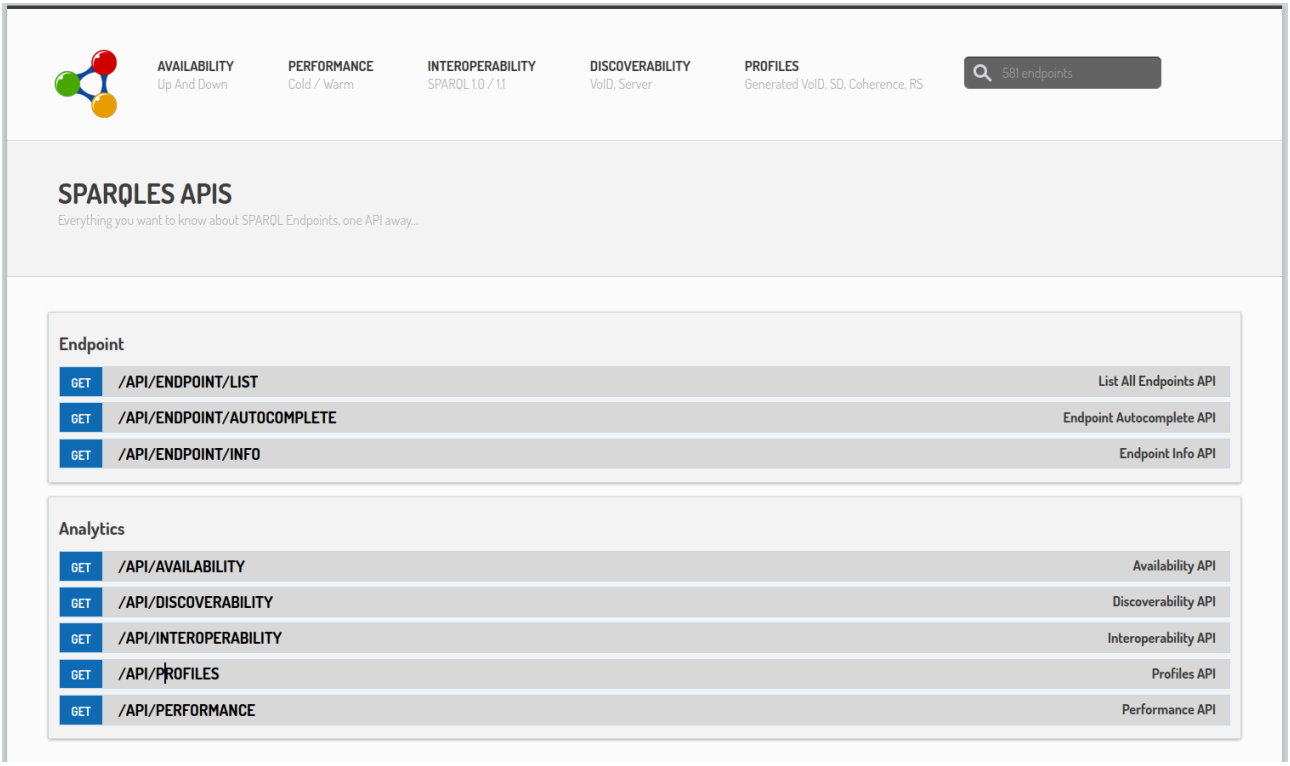

Figure 4: The REST APIs of the 3DFed SPARQL Endpoint Monitoring Service.

3.5 REST API

Due to the addition of the Calculation task, we had to add a new API that can retrieve analytical data for all endpoints related to this task type. In addition, the APIs related to all information about a particular endpoint needed to be updated to incorporate information about the newly created profiles, as well. An up-to-date list of all APIs with brief explanations is available online[9], and a screenshot of the corresponding page is shown in Figure 4.

A newly introduced API is the Profiles API[10], an analytical API that returns the aggregated results of all endpoints in terms of VoID and SD metadata, as well as the coherence and RS values. For each endpoint, it provides information on whether VoID and SD metadata were successfully generated, whether the metadata is complete or partial, the coherence and RS values (if successfully calculated), and the time of the latest executed Calculation task.

The most useful API is certainly the one that, for a given endpoint, returns all the information about it[11], i.e., in addition to the URL and the label of the dataset, the results of all types of tasks performed against it, including availability, performance, interoperability, and discoverability data. It has been updated to expand its result by including newly generated profiles, i.e., VoID and SD metadata, as well as the coherence and RS values. Unlike the analytics APIs, this API takes the URL of a specific endpoint as input (parameter `uri`), therefore, if someone is interested in the endpoint whose endpoint view page is presented in Figure 2, an example of its call follows:

```
https://sparqles.demo.openlinksw.com/api/endpoint/info?uri=
                                                http://dbtune.org/classical/sparql/
```

[9] https://sparqles.demo.openlinksw.com/api
[10] https://sparqles.demo.openlinksw.com/api/profiles
[11] https://sparqles.demo.openlinksw.com/api/endpoint/info

while its current result in `json` format is shown in Listing 5. Availability, performance, interoperability, and discoverability data are not shown (replaced by three dots) for simplicity and shorter presentation.

Listing 5: An example result of the endpoint info API for `http://dbtune.org/classical/sparql/`.


```
[
  {
    "endpoint": {
      "uri": "http://dbtune.org/classical/sparql/",
      "datasets": [
        {
          "uri": "http://dbtune.org/classical/sparql/",
          "label": "DBTune.org/classical"
        }
      ]
    },
    "availability": ...,
    "performance": ...,
    "interoperability": ...,
    "discoverability": ...,
    "calculation": {
      "triples": 419519,
      "entities": 75625,
      "classes": 14,
      "properties": 38,
      "distinctSubjects": 75625,
      "distinctObjects": 143068,
      "exampleResources": [
        "http://dbpedia.org/resource/Alexander_Uninsky",
        "http://dbpedia.org/resource/Anton_Rubinstein",
        "http://dbpedia.org/resource/Antonio_Pappano"
      ],
      "VoID": "<http://dbtune.org/classical/sparql/>\n
                    a   <http://rdfs.org/ns/void#Dataset> ;\n
                    <http://rdfs.org/ns/void#classes>\n
                        \"14\" ;\n
                    <http://rdfs.org/ns/void#distinctObjects>\n
                        \"143068\" ;\n
                    <http://rdfs.org/ns/void#distinctSubjects>\n
                        \"75625\" ;\n
                    <http://rdfs.org/ns/void#entities>\n
                        \"75625\" ;\n
                    <http://rdfs.org/ns/void#exampleResource>\n
                        <http://dbpedia.org/resource/Antonio_Pappano> ,
                        <http://dbpedia.org/resource/Anton_Rubinstein> ,
                        <http://dbpedia.org/resource/Alexander_Uninsky> ;\n
                    <http://rdfs.org/ns/void#properties>\n
                        \"38\" ;\n
                    <http://rdfs.org/ns/void#sparqlEndpoint>\n
                        <http://dbtune.org/classical/sparql/> ;\n
                    <http://rdfs.org/ns/void#triples>\n
                        \"419519\" ;\n
                    <https://www.3dfed.com/ontology/coherence>\n
                        \"0.31196353889616746\" ;\n
                    <https://www.3dfed.com/ontology/relationshipSpecialty>\n
                        \"961.3786450324915\" .\n\n

                <http://dbtune.org/classical/sparql//profile>\n
                    a   <http://rdfs.org/ns/void#DatasetDescription> ;\n
                    <http://purl.org/dc/terms/creator>\n
                        <https://sparqles.demo.openlinksw.com> ;\n
                    <http://purl.org/dc/terms/date>\n
                        \"2023-09-24\" ;\n
                    <http://purl.org/dc/terms/title>\n
                        \"Automatically constructed VoID description for a SPARQL Endpoint\" ;\n
                    <http://xmlns.com/foaf/0.1/primaryTopic>\n
                        <http://dbtune.org/classical/sparql/> .\n",
      "VoIDPart": false,
      "SD": "<http://dbtune.org/classical/sparql/>\n
                a   <http://www.w3.org/ns/sparql-service-description#Service> ;\n
                <http://www.w3.org/ns/sparql-service-description#defaultDataset>\n
                    [   a   <http://www.w3.org/ns/sparql-service-description#Dataset> ;\n
                        <http://www.w3.org/ns/sparql-service-description#defaultGraph>\n
                            [   a   <http://www.w3.org/ns/sparql-service-description#Graph> ;\n
                                <http://rdfs.org/ns/void#triples>\n
                                    \"419519\"\n
                            ]\n
                    ] ;\n
                <http://www.w3.org/ns/sparql-service-description#endpoint>\n
                    <http://dbtune.org/classical/sparql/> .\n",
      "SDPart": false,
      "coherence": 0.31196353889616746,
      "RS": 961.3786450324915
    }
  }
]
```

3.6 Other Improvements

To facilitate adding a new endpoint for the platform to monitor, rather than entering it into the DataHub catalog (which can be complicated and time-consuming), the user of the backend part of the application has the option to add a new endpoint by calling a Java program with the appropriate options. All that is required is to specify the endpoint URL, and from that moment on, the endpoint is monitored by the platform:

```
sh bin/sparqles -p src/main/resources/sparqles_docker.properties --addEndpoint EndpointUrl
```

During the development of the platform, an issue where the availability diagram was not displayed on the main page [5] was fixed. The issue occurred because an analytical collection was not created in the MongoDB database to aggregate endpoint availability data by month. After generating this collection, the diagram is displayed correctly, as can be seen in Figure 3.

4 Conclusion

The main goal of T2.1 was to develop an automatic, scalable tool to generate data storage profiles on-the-fly. More specifically, this tool was intended to work with a set of URLs of SPARQL endpoints (RDF data storage solutions) and automatically generate a profile for each endpoint as an RDF dataset. As part of our work on T2.1, we successfully developed the tool, and integrated it into the SPARQL endpoint monitoring platform. With it, we are able to generate a profile for each endpoint, which contains its VoID data, SD data, the coherence of dataset and the relationship specialty. All this information is collected via SPARQL queries, issued directly to the SPARQL endpoint in question. The resulting dataset is then publicly available via the SPARQL endpoint monitoring platform.

Additionally, the SPARQL endpoint monitoring platform provides statistics for all SPARQL endpoints it monitors regarding their generated profiles, more specifically the VoID profiles, the SD profiles, and the values for coherence and RS. On the platform, users can browse, explore the available profiles online and are able to compare them using different graphical charts and tables.

The upgraded SPARQL endpoint monitoring platform is open-source, meaning its code is publicly available on GitHub, and can be deployed, hosted, used and upgraded by any interested party in the future.

References

[1] Keith Alexander, Richard Cyganiak, Michael Hausenblas, and Jun Zhao. Describing Linked Datasets with the VoID Vocabulary. 2011. https://www.w3.org/TR/void/.

[2] DCMI Usage Board. DCMI Metadata Terms. 2020. https://www.dublincore.org/specifications/dublin-core/dcmi-terms/.

[3] Dan Brickley and Libby Miller. FOAF Vocabulary Specification. 2004. http://xmlns.com/foaf/0.1/.

[4] Songyun Duan, Anastasios Kementsietsidis, Kavitha Srinivas, and Octavian Udrea. Apples and Oranges: A Comparison of RDF Benchmarks and Real RDF Datasets. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 145–156, 2011.

[5] Milos Jovanovik and Mirko Spasić. Report on Monitoring the Data Storages. https://www.3dfed.com/wp-content/uploads/2023/03/3DFed-D2.2-Report-on-Monitoring-the-Data-Storages.pdf, March 2023. Project 3DFed Deliverable 2.2.

[6] Shi Qiao and Z Meral Özsoyoğlu. RBench: Application-Specific RDF Benchmarking. In *Proceedings of the 2015 acm sigmod international conference on management of data*, pages 1825–1838, 2015.

[7] Muhammad Saleem, Gábor Szárnyas, Felix Conrads, Syed Ahmad Chan Bukhari, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. How Representative is a SPARQL Benchmark? An Analysis of RDF Triplestore Benchmarks. In *The World Wide Web Conference*, pages 1623–1633, 2019.

[8] Mirko Spasić, Milos Jovanovik, and Arnau Prat-Pérez. An RDF Dataset Generator for the Social Network Benchmark with Real-World Coherence. In *Workshop on Benchmarking Linked Data (BLINK 2016), at the 15th International Semantic Web Conference (ISWC 2016)*, 2016.

[9] Pierre-Yves Vandenbussche, Jürgen Umbrich, Luca Matteis, Aidan Hogan, and Carlos Buil-Aranda. SPARQLES: Monitoring Public SPARQL Endpoints. *Semantic Web*, 8:1–17, 01 2017.

[10] Gregory Todd Williams. SPARQL 1.1 Service Description. 2013. https://www.w3.org/TR/sparql11-service-description/.