Semantic Data-Driven Microservices

Ivan Salvadori, Alexis Huf and Frank Siqueira
Graduate Program in Computer Science
Department of Informatics and Statistics
Federal University of Santa Catarina
Florianópolis, SC - Brazil
{ivan.salvadori,alexis.huf}@posgrad.ufsc.br, frank.siqueira@ufsc.br

***Abstract*—Nowadays, data is seen as one of the most valuable assets of organizations. Representing and exposing data in a suitable manner is mandatory for allowing consumers – either human beings or software systems – to properly retrieve and interpret such data. Web Services along with semantic Web techniques may be adopted to address this issue. This paper presents the semantic data-driven microservice, a cloud service capable of providing linked data based on non-semantic data sources. Its main goal is to work as a solution for publishing linked data and for maximizing data reuse.**

***Index Terms*—Semantic Web, Linked Data, Microservices, Web Services, Data-driven Services.**

I. Introduction

Corporations and governments produce, collect and store large volumes of data. Properly managing the data lifecycle is pivotal for the success of their projects and products. An important step is properly organizing and exposing data, which includes making decisions with regard to data structures and formats, as well as mechanisms to allow internal and external consumers to make use of data. Web technologies have been used to address these features. However, there are a variety of approaches that may be adopted to better handle data exchange.

The approach most widely adopted by Web applications consists in exposing data through web pages or data dumps. Solutions better aligned with data exchange principles implement Web Services to expose data in a more suitable format for being consumed by other software components. However, few implementations follow principles such as universality and decentralization. Universality means that the information should be accessible through URIs, while decentralization means that there is no central authority to create and expose new data or to interlink existing one [1], avoiding the creation of data silos.

Microservices are an emerging implementation approach to SOA (Software Oriented Architecture) with architectural attributes such as isolated state, loose coupling, and deployment and operation characteristics [2]. The application of the microservices approach implies the implementation of functionality and management of data independently by different services [3]. Microservices are modeled according to the single responsibility principle, which works as a guide to their implementation. Accordingly, a microservice architecture along with semantic Web technologies may be adopted to implement services that mainly manage the data lifecycle. This argument is aligned with the vision of Pautasso and Zimmermann [4], in which the Web may be seen as a graph of linked resources shared between microservices.

Some proposals to tackle the problem of exposing data on the Web have been published in the literature. Research works such as Ontobroker [5] and DataOps [6] are focused on publishing linked data based on non-semantic data, whereas Linked REST APIs [7] and OntoGenesis [8] are focused on augmenting legacy web services with semantic capabilities. However, none of them is focused on maximizing data reuse, which is one of the key features of the proposal presented in this paper.

This work presents sdd-μs, a specialized cloud service capable of converting non-semantic data into linked data. The proposed service adopts the data-driven approach to implement microservices as data providers. By using sdd-μs, there is no need to implement a Web Service to expose a data source on the Web. It converts simple data entries into semantic Web resources that can be linked to other resources provided by different data sources. In addition, it applies Mining Association Rules techniques to identify patterns, which allows the data structure to be changed in order to maximize data reuse. Finally, it provides means to infer new knowledge based on the available resources.

The remainder of this paper is organized as follows. Section II summarizes the main concepts required for understanding this work. Section III introduces the semantic data-driven microservice and shows in detail its architecture and capabilities. Section IV compares this research work with related proposals found in the literature. Section V presents an evaluation scenario and experimental results. Finally, Section VI draws some conclusions concerning this work and presents perspectives for further improvement.

II. Background

This section describes the main concepts in the context of this research work.

A. Semantic Web and Linked Data

According to Bizer, Heath and Berners-Lee [9], the Web has a great potential to be a global linked data space. In this view, not only documents are made available and linked to other documents, but also their contents. In order

 DOI: 10.1109/COMPSAC.2019.00065

to achieve this purpose, data is structured in RDF triples (`subject-predicate-object`), in which the subject represents the feature being described, the predicate represents a characteristic of the subject, and the object is the value assigned to this characteristic. RDF (Resource Description Framework) is a data model for describing the semantics of resources and interconnecting them with other related information. By semantically describing resources, not only humans, but also machines are able to infer the meaning of data published on the Web. In addition, the semantic description also facilitates data integration and reuse.

Berners-Lee [10] created the concept of linked data, which represents a set of best practices for publishing structured data on the Web. It includes: (i) the use of HTTP URIs to identify and locate resources; (ii) providing useful information from URIs, properly represented in a standard model; and (iii) adding more links to related resources in order to obtain further information. These principles aim at creating a single global data repository, resulting in a network of connections that forms the foundations of a new Web.

B. Data Linking

Data linking is the task of finding equivalent resources that represent the same real-world object [11]. Data linking can be formalized as an operation that takes collections of data as input and produces a set of binary relations between their entities as output.

The data linking problem can be divided into two main groups: connection of data from heterogeneous sources; and comparison of data for data cleaning, duplicate detection or merge/purge records.

Data linking is similar to database record linkage and to ontology schema matching, both widely explored in the literature [12]–[14]. Data linking makes use of techniques from these areas, which can be divided into three main categories: value matching, individual matching and dataset matching. The value matching technique applies to linking entities that contain the same property value expressed in different ways. The individual matching technique is used for deciding whether two entities correspond to the same real-world object by analyzing their property values. Dataset matching takes into account all entities from two different data sources in order to create an optimal alignment between them.

C. Data Services and Microservices

In a broad sense, data services may be seen as services specialized in providing and managing data. Several terms have been used to address these Web Services, such as: data services, data providing services, data as a service, among others. These terms have different definitions. Carey, Onose and Petropoulos [15] define data services as a specialization of Web Services capable of providing data by encapsulating a wide range of data-centric operations. Speiser and Harth [16] have a more restrictive notion, which protects data services from any side effects, taking into account only read-only operations. Vaculín et al. [17] present a different definition, in which a data providing service encapsulates one or more data sources into a set of Web Service operations. Paik et al. [18] state that the idea of data as a service is providing a uniform access through a standard interface for data consumers, bypassing the business logic layer.

Despite some differences, these notions have in common the focus on exposing data through a Web interface. They also lead to the separation between the business logic layer and the data access layer as distinct services. As a result, a data-driven Web Service could be seen as a particular implementation approach to SOA, where data, instead of business functionalities, plays the central role.

According to Zimmermann [2], microservices can be seen as a particular implementation approach to build SOA applications, comprising both service development and deployment. Thus, microservices present an evolutionary and complementary strategy to develop SOA applications, adhering to tenets such as fine-grained interfaces, polyglot programming and persistence, and decentralized continuous delivery.

Microservices may interact with their peers by using Web Services and Web APIs, or by using a message broker that supports message queues and allows communication through publisher/subscriber mechanisms. Microservices can also employ semantic Web Services technology, resulting in semantic microservices. The current trend to implement different services in the context of software containers, for example, is more of a coincidence than a direct consequence of microservice design [19].

III. Semantic Data-driven Microservice

In recent years, the interest in data produced and published by companies and governments has increased. Such data has not been necessarily attached to a specific logic layer. In this context, possible consumers are interested only in accessing the data, and not interacting with business procedures.

The vast majority of Web Services are implemented according to two main approaches: SOAP and REST. The former is defined in terms of the messages exchanged between a service provider and its clients. The latter provides a uniform interface to manage resources by following the semantics of the HTTP protocol. However, both approaches are used to expose functions defined in an application logic layer [18].

According to Pautasso et al. [20], microservices should be developed by following the business-driven service design, which focuses on business goals. However, the authors keep an open question about the existence of a microservice responsible for providing all customer data. This question is not that simple to answer, since it combines aspects such as decoupling and granularity, which are handled mostly by modeling microservices around bounded contexts that implement a given domain logic. However, even data-driven implementations are domain-oriented. In a broad sense, a business logic layer implements a functionality based on user expectation. The same can be said about data-driven services, which provide data aligned with a given consumer expectation.

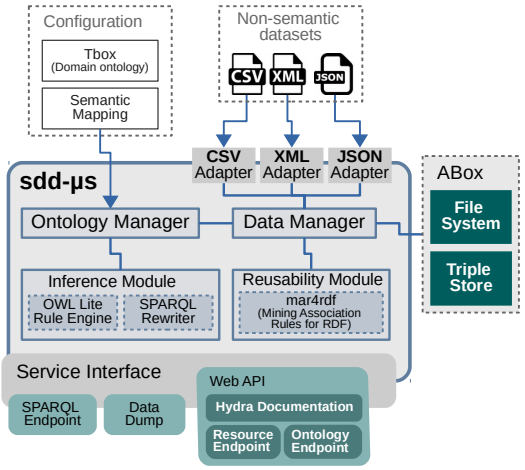


Fig. 1: The reference architecture

Considering the set of characteristics that differentiate microservices from other implementation approaches – such as single-responsibility units, isolated state, distribution, elasticity and loose coupling – the microservices architecture turns out as a suitable alternative to develop data-driven services. By explicitly separating data from business operations in distinct microservices, users can freely interact with data without the restrictions imposed by business operations. Due to this characteristic, data may be more effectively reused for different purposes.

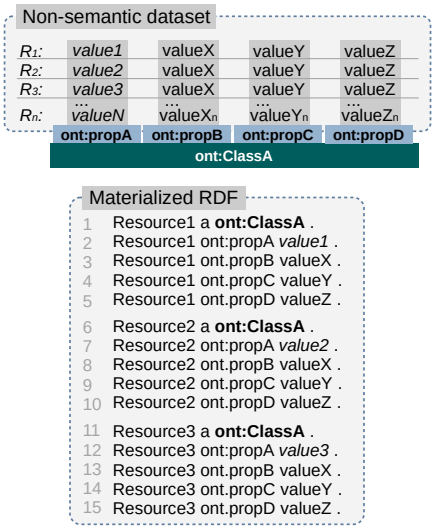
That been said, sdd-µs adopts the data-driven approach, which implies that the service does not implement any business operations, but only functionalities for managing the lifecycle of read-only data. Its main goal is to facilitate publishing linked data based on a non-semantic dataset. Consumers can interact with the data through multiple service interfaces in order to fulfill different expectations and uses. However, what differentiates the sdd-µs from other proposals is the capability of optimizing data in order to improve its reusability. Two processes are performed to achieve such a feature, which are semantic enrichment and resource structure optimization. In addition, sdd-µs provides an efficient support for inference, which allows refining the data retrieval behavior based on conceptual terminologies.

A. Semantic Enrichment

Producing semantically enriched data is the first step towards the Web of data. However, most of the information produced by governments, universities and enterprises is not available as such. Usually, data is not described by an ontology and is not structured as RDF triples, posing obstacles to data integration and reuse. In order to address this issue, sdd-µs provides the means to semantically enrich datasets. As depicted by Fig. 1, sdd-µs accepts as input a configuration and a non-semantic dataset. The configuration contains a TBox (i.e., a set of terminological statements that conceptualize the dataset) and a semantic mapping that associates attributes of a non-semantic dataset with terms defined in the TBox.

```
{
  "@context": {
    "onto": "http://example.com/ontology/",
    "@type": "onto:ClassA",
    "CSV_columnmHeaderA": "onto:propA",
    "CSV_columnmHeaderB": "onto:propB",
    "CSV_columnmHeaderC": "onto:propC",
    "CSV_columnmHeaderD": "onto:propD"
  }
}
```

(a) Semantic mapping file example



(b) RDF materialization example

Fig. 2: Semantic Mapping and the resulting RDF materialization example

There are a variety of approaches for accessing the dataset managed by a data provider [21]. Data materialization and virtual integration are broadly adopted. In the former, the data is previously loaded, materialized according to a previously defined schema and stored. In the latter approach the data remains in the source and such materialization is performed to answer a query at runtime, without physically storing the data. The sdd-µs adopts the materialization approach, which means that the dataset is converted into RDF triples and stored as files or in a triple store when the service is initialized for the first time. Once the non-semantic dataset is converted into RDF triples, it is not used anymore. It is important to mention that sdd-µs also accepts a materialized RDF dataset as input. In this case, the materialization step can be skipped.

Fig. 2 (a) shows an example of the semantic mapping for enriching a CSV file. The mapping is defined in JSON-LD syntax, in which keys are represented by CSV column headers and values are represented by the properties' URIs defined in the ontology. The reserved key `@type` is used to define a semantic class for each CSV record. In this example, `ont:propA` `ont:propB`, `ont:propC` and `ont:propD`, properly defined in the domain ontology, were associated with CSV column headers in order to be part of an independent Web resource instance of the semantic class `ont:ClassA`. Ad-

 DOI: 10.1109/COMPSAC.2019.00065

vanced configurations can be applied to handle more complex mapping designs, such as mapping more than on CSV record per Web resource and hierarchical structures. The resulting RDF materialization can be seen in Fig. 2 (b).

Once materialized, the dataset can be retrieved through multiple service interfaces, which include a SPARQL endpoint, a data dump interface and a Web API. The SPARQL endpoint[1] allows the execution of SPARQL queries on the dataset through HTTP GET requests. The data dump interface allows consumers to download a file that consists of all stored triples. Finally, the Web API is capable of providing linked data documents (entities with unique HTTP URLs) as well as a Hydra [22] documentation that describes how to retrieve such entities. Entities may be retrieved in a variety of formats, such as XML/RDF, Turtle, JSON-LD, among others, according to content negotiation. In addition, the Web API interface allows the domain ontology to be managed, which permits adding or changing concepts.

B. Support for Inference

Reasoning is an important feature, specially for data-driven implementations, which are focused on data and on the potential knowledge that may be inferred. Moreover, it may be seen as the most important reason to adopt semantic Web techniques. This feature allows the derivation of new facts from those explicitly present in the data and the concepts defined in the TBox. Such definitions can be used to provide distinct perspectives over the data and to empower data integration. The former may be implemented by defining class hierarchy such as subclasses, or by modeling restrictions to create new concepts based on specific conditions. The latter may be implemented by defining equivalences between classes and properties as well as between resources that represent the same object in real world, usually expressed with `owl:sameAs`. It can also be seen as a built-in mechanism for dealing with data heterogeneity.

For this purpose, the Inference module provides reasoning support based on two different implementations: Apache Jena OWL Rule Engine and a SPARQL rewriter. The former uses the Apache Jena implementation[2] to perform inferences. The latter, a contribution of this work, rewrites a SPARQL query that requires reasoning into a new query that incorporates such elements based on a domain ontology analysis. Rewriting queries represents an alternative to the standard inference support offered by Jena. This alternative is necessary given that Jena may present a significant performance degradation, depending on the size of the dataset and on the complexity of the query.

Fig. 3 shows a simple example of query rewriting. R1 presents a materialized resource, and T1 shows a TBox that defines an equivalence between two properties. Q1 presents a SPARQL query that requires inference support to retrieve R1, since the resource has been previously materialized with an equivalent property. Finally, Q2 presents the resulting SPARQL query after the rewriting process. Currently, the query rewriter supports class and property equivalence, subclasses, as well as `owl:sameAs` statements.

```
R1: <Resource1> a <http://example.com/onto/ClassA> .
    <Resource1> <http://example.com/onto/propA> "value1" .

T1: @prefix owl: <http://www.w3.org/2002/07/owl#> .
    @prefix onto: <http://example.com/onto/> .
    @prefix anotherOnt: <http://example.com/anotherOnt/> .
    onto:propA a owl:DatatypeProperty .
    onto:propA owl:equivalentProperty anotherOnt:propX

Q1: SELECT ?resource  WHERE {
      ?resource anotherOnt:propX "value1" .
    }

Q2: SELECT ?resource  WHERE {
      { ?resource onto:propA "value1" } UNION
      { ?resource anotherOnt:propX "value1" }
    }
```

Fig. 3: Query rewriting example

C. Resource Structure Optimization

Optimizing the structure of data is not a trivial task. It requires qualified specialists that have deep knowledge on the specific domain and software tools to assist them in this process. Though, even for a specialist, restructuring data in such a way that information would be better represented through linked resources may be something difficult to accomplish. By using data mining techniques, sdd-µs provides a means to optimize the initial RDF materialization in order to achieve higher levels of data reuse.

An important part of the optimization process is the discovery of association rules, which consists in determining relationships between sets of items in a very large database. Agrawal and Srikant [23] state this problem as follows. Let $I = \{i_1, i_2, ..., i_m\}$ be a set of m items. Let $D = \{t_1, t_2, ..., t_n\}$ be a set of n transactions, each one identified by a unique transaction id (TID). Each transaction t consists of a set of items from I and an itemset I is contained in a transaction $t \in D$ if $I \subseteq t$. The support of an itemset I is the percentage of transactions in D containing I. Association rules are of the form $r : I_1 \xrightarrow{c} I_2$, with $I_1, I_2 \subset I$ and $I_1 \cap I_2 = \phi$. Given the user defined minimum support $(minsup)$ threshold, the problem of mining association rules can be divided in two sub-problems: (i) find all itemsets in D with support greater or equal to $minsup$ and (ii) for each itemset found, generate all association rules $I_2 \xrightarrow{c} I_1 - I_2$, where $I_2 \subset I_1$. That been stated, sdd-µs adopts the A-Close algorithm, proposed by Pasquier et al. [24], as implemented in the Open-Source Data Mining Library [25].

To properly explain the structure optimization process, consider the following example. Table I presents three records, each one containing seven properties in which their values share some level of association. The process starts with converting each record into a transaction, which implies converting its values into a set of items as an ordered numerical

[1] http://www.w3.org/TR/sparql11-protocol
[2] https://jena.apache.org/

TABLE I: Example of records about employees

Property	R1	R2	R3
employeeName	Alice	John	Bob
employeeSecNumber	123	456	789
employeeBirthDate	01/01/1952	01/02/1954	01/03/1950
companyName	Void Corp	Void Corp	Acme Corp
companyLocation	São Paulo	São Paulo	New York
salary	2.000	1.000	1.000
admissionDate	01/01/2018	01/01/2018	01/01/2018

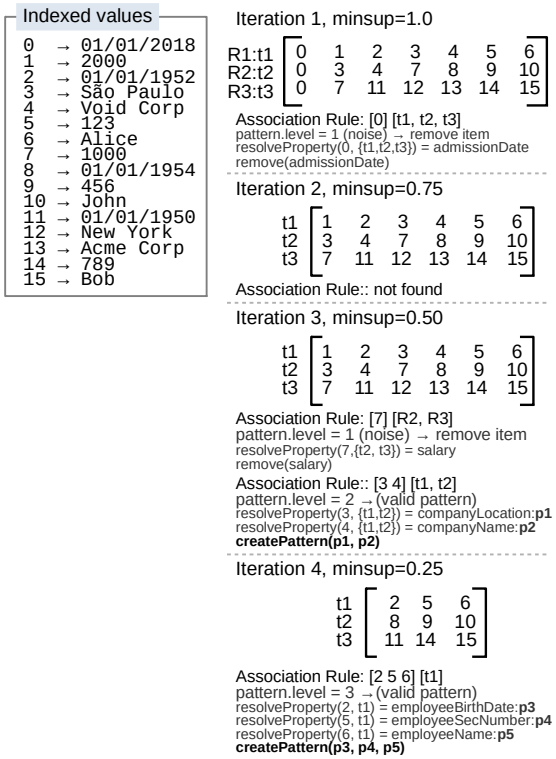


Fig. 4: Mining association rules example

vector. This step consists in creating an index that associates each literal value with a unique integer number, as shown in Fig. 4. These vectors are organized into a single matrix used as input to the A-Close algorithm.

It is necessary to perform several iterations in order to recognize frequent closed itemsets, and then generate association rules in different support thresholds. Iterations start from the maximum support and will be decreased according to a configurable parameter. In this example, the decrease rate was set to `0.25`. The first iteration is setup with minimal support to 1.00, which results in finding the association rule `[0]` in `t1, t2, t3`, with level 1. The level represents the number of items in a given itemset. Only association rules with level equal or greater than 2 are eligible to be part of a pattern, otherwise they are considered noise. The next step is to find the properties associated with the resulting itemset. In this case, the itemset `[0]` corresponds to the literal value `01/01/2018`, which in its turn is associated with the property *admissionDate* for all transactions. Finally, this item is removed from all itemsets of the initial matrix. It

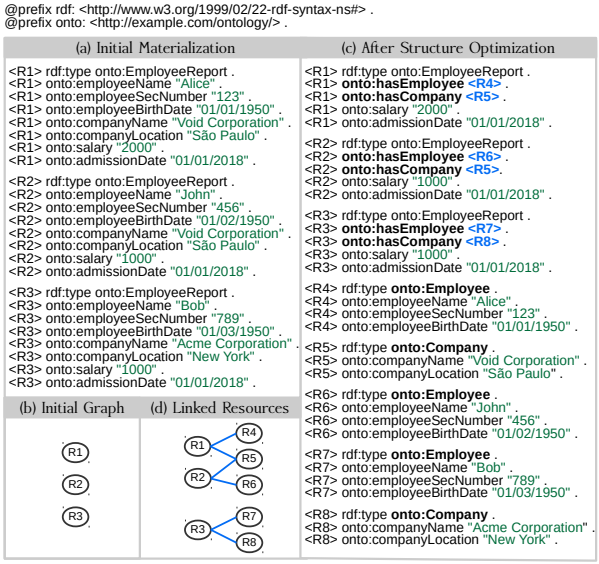


Fig. 5: Example of a resource structure optimization result

is important to mention that removing previously found closed itemsets is essential to properly recognize further association rules using lower support thresholds. Iteration 2 is setup with `minsup=0.75`, however, there is no association rule that is identified using this threshold. Iteration 3 results in two association rules. The first one is considered noise, which results in removing all items associated with the property `salary`. The second one is considered valid and its items are resolved to properties to be part of a pattern. In this case, a new semantic class is created and properties `companyLocation` and `companyName` will be restructured in an independent resource. In iteration 4, properties `employeeBirthDate`, `employeeSecNumber` and `employeeName` are also combined to be part of an independent resource.

The aforementioned steps are performed to recognize association rules and then generate patterns that essentially create new semantic classes and `objectProperties`. Based on these new concepts, a new RDF materialization is performed in order to update the dataset with the resulting structure optimization. Fig. 5 (a) describes the initial RDF materialization, which basically translates the records of Fig. I in resources. This literal translation results in a disconnected graph, as shows Fig. 5 (b). However, as Fig. 5 (c) shows, the new RDF materialization contains new resources that aggregate properties according to the resulting patterns. These new resources are then connected with existing ones, resulting in a connected graph of linked resources, as describes Fig. 5 (d). Despite the fact that, for this example, the resulting optimized materialization required more triples to represent the same information, a very large dataset with a significant level of data overlap would result in a smaller optimized dataset. Moreover, datasets in which there is no clear separation of concepts or whose records have no apparent relation will also benefit from the resource structure optimization.

IV. Related Work

As pointed out by Marjit et al. [26], the majority of the data available on the Web is stored in formats other than RDF, which are called legacy data. There have been several research works aimed at publishing legacy data as linked data on the Web. Ontobroker [5] is a semantic middleware that works as an Integration Engine. It exposes data from information sources – such as relational database systems, Web Services, and Excel sheets – through a SPARQL endpoint. Instead of materializing RDF, Ontobroker provides data wrappers to map data in the original format. However, reasoning capabilities are not supported.

DataOps [6] proposes the Anything-to-RDF data integration, a toolkit that supports the integration of both semantic and non-semantic data. It provides means to explore and visualize the resulting data based on a SPARQL endpoint and proprietary formats. DataOps materializes RDF based on a mapper, which consolidates new data with existing data instances by establishing `owl:sameAs` links. Despite being able to create links among materialized resources, DataOps does not support inference over the data.

Another approach to provide linked data based on non-semantic sources is augmenting legacy Web Services with semantic capabilities, which include semantic descriptions as well as semantic data representations. This approach mainly aims at improving the understanding of service operations and therefore compositions.

Linked REST APIs (LRA) [7] is a conceptual framework for REST service integration based on linked data models. It introduces a vocabulary that enables the semantic representation of REST services, including authentication mechanisms, quality, and relationships between inputs and outputs. LRA provides means to answer SPARQL queries through a fully automatic process. Reasoning capabilities are also supported, allowing to perform inferences for `owl:sameAs` and `owl:equivalentClass`. Authors argue that the reason for the limited adoption of semantic Web techniques is related to the fear developers have to use it. However, SPARQL is not widely known among Web Service developers, therefore it may not be the most suitable interface to expose data.

OntoGenesis [8] is an architecture for semantically enriching representations provided by data services. Its goal is to semantically enrich syntactic representations by automatically associating concepts defined in domain ontologies with representations. Its architecture has two main components: the semantic adapter and the OntoGenesis Web API. The former should be used along with legacy Web Services to automatically send requests containing the representations required by the user to the OntoGenesis Web API. The latter implements an engine to semantically enrich these representations. The enrichment process is performed by a property matching mechanism, which reuses well-known concepts defined by external sources and public ontologies. Despite being able to semantically enrich data representations, Ontogenesis does not support inference capabilities.

Moreover, some patterns have emerged to address the development of Web services in the context of data services. Database-is-the-Service [27] is a pattern that considers the database as a service by itself. Despite the fact that this pattern does not separate the business layer from the data access layer in distinct services, it promotes data as the most valuable asset.

The microservice architecture has also been used along with data science techniques. Thiele et al. [28] adopt microservices to expose entities generated by machine learning-based processes over a network. Zhang et al. [29] propose an analytics-focused API design for data services, which provides context information about the origin, scope, and historical manipulations on a certain dataset, allowing to share and reuse historical data exploration process and derived data. Those patterns are aimed at providing solutions for specific domains, in contrast to this proposal that may be applied to a general data publishing problem.

V. Evaluation

In this section we describe our experimental methodology and analyze the obtained results. The objective of this evaluation is to measure the efficiency of the resource structure optimization process regarding data reuse and its cost in terms of processing time. In addition, this evaluation aims at identifying the impact of performing reasoning, by comparing two different inference approaches: Jena OWL Rule Engine and SPARQL rewriting. In order to allow the replication of experimental results, source code and instructions for setting up this evaluation are available in a public repository[3].

The evaluation used real data from two distinct data providers. The first dataset is provided by the Public Security Secretariat of the state of São Paulo (SSP-SP)[4] - Brazil. The SSP/SP system publishes police reports that describe suspicious death, intentional homicide, robbery followed by murder, car theft, among others. Information about the report such as location, police station and date of the incident are available. For this evaluation, only police reports that describe car theft were considered. A total of 175 CSV files have been downloaded, containing reports from 2003 to 2017. The second dataset is provided by The NYC Open Data portal[5], which publishes a variety of datasets, including data about business, health, education, government, environment, among others. For this evaluation, was considered the Parking Violations (NYC- PVI) dataset[6] - Fiscal Year 2019, which describes information such as vehicle details, data and location in which the violation took place, type of violation, among others. This dataset is provided as a single CSV file, which contains violations from January 1, 2018 to September 10, 2018, and is monthly updated.

[3] https://salvadori.bitbucket.io/projects/sddms/
[4] http://www.ssp.sp.gov.br/transparenciassp/
[5] http://opendata.cityofnewyork.us/
[6] https://data.cityofnewyork.us/City-Government/Parking-Violations-Issued-Fiscal-Year-2019/pvqr-7yc4

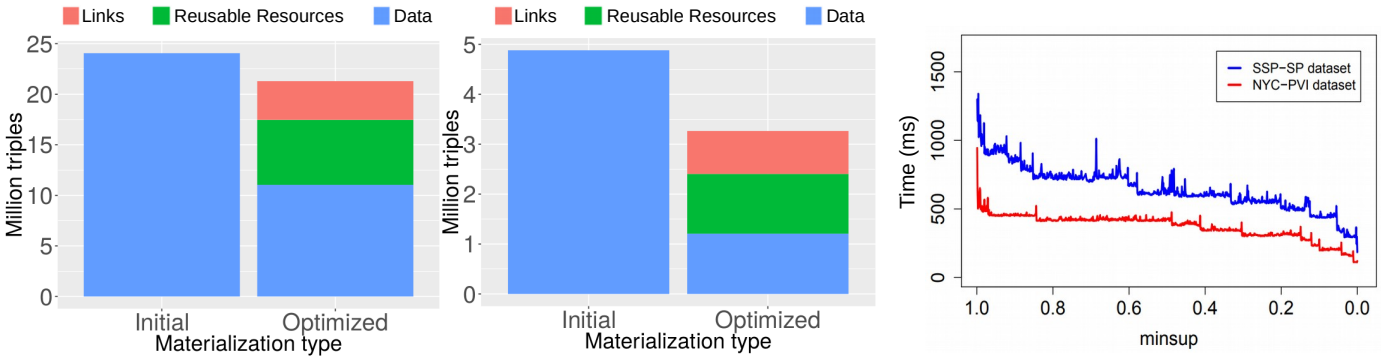


Fig. 6: Structure resource optimization results: (a) SSP-SP dataset, (b) NYC- PVI dataset and (c) processing time

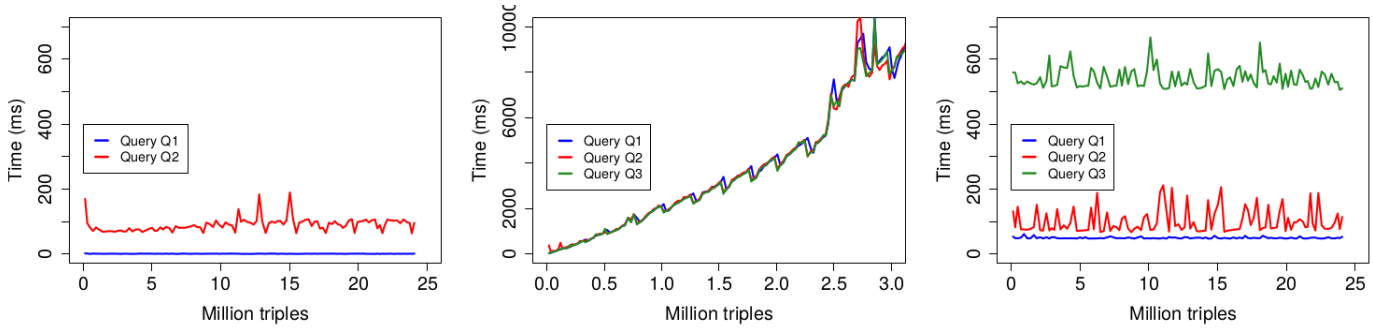


Fig. 7: Query response time: (a) no inference support, (b) Jena inference enabled and (c) sdd-μs query rewriter

The experiment was executed using the computational infrastructure provided by the Cloud Computing for Cooperation[7], with the following configuration: a dedicated server with 24 Intel Xeon X5690 processors at 3.47GHz and 148 GiB of primary memory, running the CentOS-7 operating system and openjdk version 1.8.0_141 with maximum heap size set to 2 GiB for both the structure resource optimization and sdd-ms query rewriter evaluations.

Fig. 6 shows the results regarding the resource structure optimization process. The optimization of the SSP-SP dataset is represented by Fig. 6 (a). The initial materialization required 24.057 million triples to represent the information. In the initial materialization, each CSV record was converted into a single RDF resource, which holds all mapped properties. As a result of the structure optimization, a new materialization was created with all recognized data patterns as well as the necessary links to connect the original data with the new resources. Those patterns are represented by resources, which are instances of new concepts. The optimized materialization required 21.292 million triples to represent the same information, resulting in a reduction of 11.7%. However, the most important result is that it was able to properly reorganize the information among more reusable resources. Three patterns were recognized for this dataset. The first one aggregates properties that describe the stolen vehicle. The second one aggregates properties that describe the police station responsible for the report. The last one represents the common information about the location where the incident took place, which includes region, street and city name.

For the NYC-PVI dataset, the optimization resulted in a significant reduction of the materialized dataset, as shows Fig. 6 (b). Two patterns were recognized for this dataset. The first one aggregates properties that describe the vehicle. The second one aggregates properties that describe the violation type along with the street name. While this optimization resulted in a significant reduction of the resulting dataset, it came up with a pattern that does not necessary represent a common sense reorganization. The reason for that is the lack of data to properly recognize the pattern, however, it represents the real nature of the data. In other words, the resulting patterns are result of the data, and do not necessarily follow a given domain logic.

[7] http://www.c3lab.tk.jku.at

```
@prefix onto: <http://www.public-security-ontology/> .
@prefix anotherOnto: <http://www.anotherOntology.com/> .
Q1: SELECT  ?p ?o { <{resource_URI}> ?p ?o}

Q2: SELECT ?resource  WHERE {
      ?resource a onto:TheftAutoReport .
      ?resource onto:timeOfDay "EVENING" .
    } limit 100 offset 0;

Q3: SELECT ?resource  WHERE {
      ?resource a onto:CriminalReport .
      ?resource anotherOnto:periodOfDay "EVENING" .
    } limit 100 offset 0;
```

Fig. 8: SPARQL queries

Fig. 6 (c) shows the processing time to perform the optimization for each dataset. The size of the dataset has an important effect on the processing time. The spikes represent the points where vectors of transactions were recreated due to a found pattern or a noise. It also explains the decreasing processing time behavior, in which the main factor is the size of the matrix rather than the minsup.

This evaluation also takes into consideration the Processor module, comparing the Apache Jena OWL Rule Engine with the sdd-μs SPARQL rewriter. In order to compare these two inference approaches, three queries, shown by Fig. 8, were evaluated. These three queries were issued every time a new Web resource was inserted into the RDF dataset throughout the materialization process for the SSP-SP dataset. Query Q1 only retrieves the latest inserted Web resource, Q2 filters a collection of resources based on property `"timeOfDay"`, and Q3 produces the same result of Q2, however it requires reasoning for inferring the equivalence between `"TheftAutoReport"` and `"CriminalReport"`.

Fig. 7 shows the results of this comparison. In Fig. 7 (a), Q1, Q2 and Q3 were executed without inference support. One can see that Q3 produces no results as it requires reasoning features. However, considering all the 24.057 million triples according to the initial materialization[8], the execution time for Q1 is close to zero milliseconds, while for Q2 it is more than 100 milliseconds. However, when the Jena OWL Rule Engine is enabled, the time required for Q3 to produce the expected results increases sharply, as can be seen in Fig. 7 (b). It is worth noticing that, regardless of the query, the execution time was similar. Approximately 10 seconds were required to execute each query over 3 million triples. Finally, Fig. 7 (c) shows the result for the sdd-μs Query Rewriter. In this approach, the execution times for Q1 and Q2 were similar when compared to the execution without inference support. Moreover, the required time to execute Q3 was in the interval between 500 and 600 milliseconds.

This evaluation shows that the adoption of the proposed solution results in a significant data reuse. The optimization process was able to convert the datasets, originally represented by a collection of isolated resources into connected graphs. By adopting the proposed inference support, the time to perform inferences was drastically reduced when compared to Apache Jena. It is worth to mention that despite the inference times seem to be constant as shows Fig. 7 (c), actually they are linear when considered larger datasets.

[8] There were no significantly different results in the initial and the optimized materialization for executing the queries used in this evaluation.

VI. Conclusions and Future Work

This paper presented the semantic data-driven microservice, a service capable of providing linked data based on non-semantic data sources. By using sdd-μs there is no need to implement new Web Services to expose data on the Web. The sdd-μs provides an efficient inference support for issuing queries on large semantic datasets. Moreover, it is able to maximize the reuse of data by performing a resource structure optimization process to identify data patterns. Based on these data patterns, resource properties are combined in order to create new semantic concepts. Then, all resources with those properties will be restructured, resulting in a connected graph of linked resources.

Evaluation experiments showed the efficiency of the proposed optimization process, through which data patterns were recognized in real datasets, resulting in a significant reuse of data. In addition, the evaluation showed that Jena's Rule Engine is not suitable for performing even simple inferences. By rewriting the SPARQL queries, sdd-μs query rewriter dramatically reduced the execution time, allowing the use of inference, even for a small set of axioms.

Assuming that data can also be modeled as a monolith, microservices may be adopted as a suitable solution for distributing data across several small service providers. However, some important characteristics associated with microservices, such as disposability and native cloud orientation, may conflict with linked data principles. Disposability means that a given microservice can be removed from the infrastructure if it is no longer useful. Native cloud-oriented applications refer to software designed to be deployed in a cloud computing infrastructure. It often implies elasticity, which makes use of transient computing resources. On the other hand, linked data assumes that URLs that connect Web resources are permanent. That said, the adoption of microservices for publishing linked data requires special attention to these details, otherwise, the quality of linked data publishing may be drastically affected.

In future work we intend to adapt the Reusability Module to support other mining algorithms. The current implementation adopts the A-Close algorithm for mining association rules. However, a variety of other algorithms with a similar purpose are available and may result in better results depending on characteristics of the dataset to be optimized. Additionally, adjustments in the algorithm parameters may yield better results in certain cases. Therefore, sdd-μs could apply the most suitable algorithm according to the characteristics of the dataset. New features are to be implemented, such as considering OWL restrictions in the Inference Module and linking Web resources managed by several sdd-μs instances running in a service container.

ACKNOWLEDGMENT

This work was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and Programa de Doutorado Sanduíche no Exterior (PDSE) 88881.131816/2016-01.

REFERENCES

[1] T. Berners-Lee, "Long Live The Web," *Scientific American*, vol. 303, no. 6, pp. 80–85, dec 2010.

[2] O. Zimmermann, "Microservices tenets: Agile approach to service development and deployment," *Computer Science - Research and Development*, nov 2016.

[3] S. Newman, *Building Microservices*, 1st ed. O'Reilly Media, Inc., 2015.

[4] C. Pautasso and O. Zimmermann, "The Web as a Software Connector: Integration Resting on Linked Resources," *IEEE Software*, vol. 35, no. 1, pp. 93–98, jan 2018.

[5] J. Angele, "OntoBroker: Mature and approved semantic middleware," *Semantic Web*, vol. 5, no. 3, pp. 221–235, 2014.

[6] C. Pinkel, A. Schwarte, J. Trame, A. Nikolov, A. S. Bastinos, and T. Zeuch, "DataOps: Seamless End-to-End Anything-to-RDF Data Integration," in *The Semantic Web: ESWC 2015 Satellite Events*. Springer, 2015, pp. 123–127.

[7] D. Serrano, E. Stroulia, D. Lau, and T. Ng, "Linked REST APIs: A Middleware for Semantic REST API Integration," in *2017 IEEE International Conference on Web Services (ICWS)*. IEEE, jun 2017, pp. 138–145.

[8] B. C. N. Oliveira, A. Huf, I. Salvadori, and F. Siqueira, "Automatic Semantic Enrichment of Data Services," in *International Conference on Information Integration and Web-based Applications and Services - iiWAS '17*, 2017.

[9] C. Bizer, T. Heath, and T. Berners-Lee, *Linked data-the story so far*. IGI Global, 2009, pp. 205–227.

[10] T. Berners-Lee, "Linked Data – Design Issues," 2011. [Online]. Available: https://bit.ly/21MR3Zt

[11] A. Ferrara, A. Nikolov, and F. Scharffe, "Data Linking for the Semantic Web," *International Journal on Semantic Web and Information Systems*, vol. 7, no. 3, pp. 46–76, Jan. 2011.

[12] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 1, pp. 1–16, Jan. 2007.

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

[14] H. Köpcke and E. Rahm, "Frameworks for entity matching: A comparison," *Data Knowl. Eng.*, vol. 69, no. 2, pp. 197–210, Feb. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.datak.2009.10.003

[21] A. Doan, A. Halevy, and Z. Ives, *Principles of data integration*. Elsevier, 2012.

[15] M. J. Carey, N. Onose, and M. Petropoulos, "Data services," *Communications of the ACM*, vol. 55, no. 6, jun 2012.

[16] S. Speiser and A. Harth, "Integrating Linked Data and Services with Linked Data Services," in *The Semantic Web: Research and Applications*. Springer, 2011.

[17] R. Vaculín, H. Chen, R. Neruda, and K. Sycara, "Modeling and discovery of data providing services," in *Proceedings of the IEEE International Conference on Web Services, ICWS 2008*. IEEE, sep 2008, pp. 54–61.

[18] H.-y. Paik, A. L. Lemos, M. C. Barukh, B. Benatallah, and A. Natarajan, "Web Services – Data Services," in *Web Service Implementation and Composition Techniques*. Springer, 2017, pp. 93–147.

[19] A. Sill, "The Design and Architecture of Microservices," *IEEE Cloud Computing*, vol. 3, no. 5, pp. 76–80, sep 2016.

[20] C. Pautasso, O. Zimmermann, M. Amundsen, J. Lewis, and N. Josuttis, "Microservices in Practice, Part 1: Reality Check and Service Design," *IEEE Software*, vol. 34, no. 1, pp. 91–98, jan 2017.

[22] M. Lanthaler, "Creating 3rd Generation Web APIs with Hydra," in *Proceedings of the 22nd International World Wide Web Conference (WWW2013)*. International World Wide Web Conferences Steering Committee, may 2013, pp. 35–37.

[23] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499.

[24] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering frequent closed itemsets for association rules," in *Proceedings of the 7th International Conference on Database Theory*, ser. ICDT '99. London, UK, UK: Springer-Verlag, 1999, pp. 398–416.

[25] P. Fournier-Viger, J. C.-W. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z. Deng, and H. T. Lam, "The SPMF Open-Source Data Mining Library Version 2," in *Machine Learning and Knowledge Discovery in Databases*. Springer International Publishing, 2016, pp. 36–40.

[26] U. Marjit, K. Sharma, A. Sarkar, and M. Krishnamurthy, "Publishing legacy data as linked data: a state of the art survey," *Library Hi Tech*, vol. 31, no. 3, pp. 520–535, sep 2013.

[27] A. Messina, R. Rizzo, P. Storniolo, M. Tripiciano, and A. Urso, "The database-is-the-service pattern forÂ microservice architectures," in *Information Technology in Bio- and Medical Informatics*. Springer, 2016, pp. 223–233.

[28] T. Thiele, T. Sommer, S. Stiehm, S. Jeschke, and A. Richert, "Exploring Research Networks with Data Science: A Data-Driven Microservice Architecture for Synergy Detection," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. IEEE, aug 2016, pp. 246–251.

[29] Y. Zhang, L. Zhu, X. Xu, S. Chen, and A. B. Tran, "Data Service API Design for Data Analytics." Springer International Publishing, 2018, vol. 10969, pp. 87–102.