

FAIR Principles and Big Data: A Software Reference Architecture for Open Science

João P. C. Castro^{1,2}^a, Lucas M. F. Romero¹^b, Anderson C. Carniel³^c and Cristina D. Aguiar¹^d

¹Department of Computer Science, University of São Paulo, Brazil

²Information Technology Board, Federal University of Minas Gerais, Brazil

³Department of Computer Science, Federal University of São Carlos, Brazil

Keywords: Open Science, FAIR Principles, Big Data Analytics, Software Reference Architecture.

Abstract: Open Science pursues the assurance of free availability and usability of every digital outcome originated from scientific research, such as scientific publications, data, and methodologies. It motivated the emergence of the FAIR Principles, which introduce a set of requirements that contemporary data sharing repositories must adopt to provide findability, accessibility, interoperability, and reusability. However, implementing a FAIR-compliant repository has become a core problem due to two main factors. First, there is a significant complexity related to fulfilling the requirements since they demand the management of research data and metadata. Second, the repository must be designed to support the inherent big data complexity of volume, variety, and velocity. In this paper, we propose a novel FAIR-compliant software reference architecture to store, process, and query massive volumes of scientific data and metadata. We also introduce a generic metadata warehouse model to handle the repository metadata and support analytical query processing, providing different perspectives of data insights. We show the applicability of the architecture through a case study in the context of a real-world dataset of COVID-19 Brazilian patients, detailing different types of queries and highlighting their importance to big data analytics.

1 INTRODUCTION


In the era of big data analytics, data is constantly being collected on an unprecedented scale. This occurs mostly due to advances in cloud computing and parallel and distributed data processing, which are responsible for reducing challenges regarding storing and querying massive datasets. These advances also motivate the sharing of datasets and their derived analyses.


In this context, a significant opportunity has emerged for the scientific community: boosting scientific data sharing to increase the collaboration between researchers across the globe. The concept of Open Science is an answer to this opportunity. Its objective is to ensure that every digital output of research objects is made available and usable free of charge (Medeiros et al., 2020). These outputs can include, but are not limited to: (i) research publications;


(ii) research data; and (iii) research methodologies, encompassing any algorithm employed in the process of generating research data.


Such magnitude of scientific data sharing demands a dedicated infrastructure, which must be implemented in a standardized manner to avoid any type of incompatibilities. Therefore, a set of standards referred to as the FAIR Principles have been proposed (Wilkinson et al., 2016). FAIR stands for Findability, Accessibility, Interoperability, and Reusability of digital datasets. These principles describe several requirements that contemporary data sharing repositories must adopt to support manual and automated deposition, exploration, sharing, and reuse. Satisfying these requirements involves handling scientific data and their associated metadata, which can result in a significant complexity depending on their volume, variety, and velocity (Chen et al., 2014).

However, the FAIR Principles alone may not be enough to guide a data engineer towards the implementation of a repository capable of addressing challenges inherent to the context of Open Science. Their proximity to the user level and the complexity intrinsic

^a  <https://orcid.org/0000-0003-3566-0415>

^b  <https://orcid.org/0000-0002-2155-8076>

^c  <https://orcid.org/0000-0002-8297-9894>

^d  <https://orcid.org/0000-0002-7618-1405>

sis to big data environments requires the adoption of a Software Reference Architecture (SRA) to assist data engineers in overcoming this gap.

According to Nakagawa et al. (2017), an SRA is “an architecture that encompasses the knowledge about how to design concrete architectures of systems of a given application or technological domain”. That is, SRAs are employed as a basis to derive architectures adapted to the requirements of specific contexts (Angelov et al., 2012). Regarding Open Science, SRAs can work as a bridge between the FAIR Principles and the repository being implemented.

Due to the close relationship between Open Science and big data analytics, an SRA designed to support the FAIR Principles must include components to store considerable volumes of data and metadata, which can be later efficiently retrieved by analytical queries. The use of a data warehouse and a data lake in this context enhances the features provided by the SRA. A data warehouse is an integrated, subject oriented, historical, and non-volatile database that is usually built by a multidimensional model (Kimball and Ross, 2011; Vaisman and Zimányi, 2014). A data lake is a considerably large raw data storage that deals with any data format, i.e., structured, semi-structured, and unstructured data (Couto et al., 2019; Sawadogo and Darmont, 2021). Besides their importance in enabling the repository compliance with the FAIR Principles, these components also contribute to the generation of data insights, an essential characteristic in the decision-making process of big data analytics.

Despite the considerable importance behind adopting an SRA to support Open Science, solutions available in the literature introduce limitations, as detailed in Section 2 and summarized as follows.

There are studies that propose SRAs for generic big data systems but are unaware of the intrinsic characteristics of the FAIR Principles. There are also implementations of the FAIR principles in repositories that are driven to specific contexts. However, these implementations do not propose an architecture generic enough to fit the concept of an SRA, negatively impacting on reusability. These limitations motivate the development of our work.

We introduce the following contributions:

- Proposal of an SRA to implement a data sharing repository that is compliant with the FAIR Principles and is able to store, process, and query massive volumes of scientific data and metadata.
- Specification of a generic multidimensional model to implement a metadata warehouse to handle the repository metadata.
- Demonstration of the applicability of our architec-

ture through a case study that manipulates a real-world dataset.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents the proposed architecture and highlights its compliance with the FAIR Principles. Section 4 details the design of the metadata warehouse. Section 5 describes the case study that instantiates the architecture and shows the execution of analytical queries, discussing their usefulness in the decision-making process. Finally, Section 6 concludes the paper.

2 RELATED WORK

In this section, we analyze studies available in the literature by dividing them in two groups. Group 1, named big data architectures, consists of general purpose big data SRAs, i.e., big data architectures that were not developed with the objective of complying with the FAIR Principles. The state-of-the-art architectures are (Davoudian and Liu, 2020): (i) traditional business intelligence (Vaisman and Zimányi, 2014); (ii) kappa (Kreps, 2014); (iii) lambda (Warren and Marz, 2015); (iv) liquid (Fernandez et al., 2015); (v) solid (Martínez-Prieto et al., 2015); and (vi) bolster (Nadal et al., 2017).

The main objective behind these architectures is the generation of knowledge from big data to assist users in the decision-making process. However, due to their concern in providing real time analytics, most of these SRAs are not designed for collecting and managing data provenance and other types of metadata. Since this is an essential characteristic for a repository to be compliant with the FAIR Principles, these architectures can be deemed inadequate for the context of Open Science.

The exceptions are the traditional business intelligence and bolster architectures. Besides being able to provide real time analytics, these architectures employ a centralized metadata repository for the collection and maintenance of different types of metadata. However, they do not comply with several requirements imposed by the FAIR Principles. For instance, these SRAs do not support the retrieval of source data objects based on their metadata. They are also not capable of guaranteeing that metadata will be kept alive even when their corresponding data objects no longer exist. Furthermore, these architectures are not concerned with ad-hoc data anonymization or the employment of knowledge mapping data structures to comply with domain-relevant community standards.

Group 2, named FAIR implementations, consists of implementations of the FAIR Principles in data

Table 1: Comparing the key characteristics of the proposed SRA with related work.

| Key Characteristics | Big Data SRAs | FAIR Implementations | Our Architecture |
|---|---------------|---|------------------|
| Complies with the FAIR Principles | ✗ | Not clear if all requirements are fulfilled | ✓ |
| Describes the association between the architecture's layers and the FAIR Principles | ✗ | ✗ | ✓ |
| Can fit the concept of a SRA | ✓ | ✗ | ✓ |
| Retrieves source data objects through its metadata | ✗ | ✓ | ✓ |
| Guarantees the existence of the metadata even when the related data object does not exist | ✗ | ✓ | ✓ |
| Enables the generation of big data insights | ✓ | ✗ | ✓ |
| Provides ad-hoc data anonymization | ✗ | ✗ | ✓ |
| Employs knowledge mapping data structures | ✗ | ✓ | ✓ |

sharing repositories that are driven to specific contexts. The work of Pommier et al. (2019) describes a workflow for the plant phenotypic data management, detailing the data models and technologies used to comply with the FAIR Principles. Devarakonda et al. (2019) introduce a workflow to enable the adoption of the FAIR Principles in the Atmospheric Radiation Measurement Data Center, specifying the dataflow and the employed components. In Lannom et al. (2020), the context of biodiversity science and geoscience is addressed through the proposal of a Digital Object Architecture in an attempt to satisfy the FAIR Principles. Sinaci et al. (2020) propose a workflow to implement the FAIR Principles in the context of health data, along with an architecture for this specific domain. Finally, in Delgado and Llorente (2021), a modular architecture that handles genomic information and supports the FAIR Principles is proposed, and the technologies employed in each service are specified.

Although studies in Group 2 are concerned with the intrinsic characteristics of the FAIR Principles, they face several limitations. First, they do not propose an architecture that is generic enough to fit the concept of an SRA. This is mostly due to the fact that the proposed solutions are domain-specific, overburdening data engineers in the process of adapting their solutions to different contexts. Second, some of the studies are described at the implementation level, such as Pommier et al. (2019) and Devarakonda et al. (2019). This negatively impacts the reuse of the proposed solutions, a FAIR principle of significant importance.

Third, none of the studies clarify which requirements imposed by the FAIR Principles are satisfied by their solutions. This fact raises two significant concerns: (i) which parts of the solution are responsible

for implementing a specific requirement; and (ii) if all requirements are being completely fulfilled. Fourth, the studies are not optimized to provide big data insights to data consumers. Although this is not a requirement imposed by the FAIR Principles, it is important to support the decision-making process. Fifth, ad-hoc data anonymization is not addressed by the studies, raising an imbroglio regarding data security and the compliance of the repository with domain-relevant community standards.

In this paper, we overcome the limitations of the studies analyzed in this section, as summarized in Table 1. We propose an SRA to implement a data and metadata sharing repository according to the FAIR Principles. We highlight the relationship between these principles and each of the architecture layers. Our solution can retrieve source data objects by using their metadata and also generate multiple insights to assist users in the decision-making process of big data analytics. Furthermore, due to the employment of a metadata warehouse, our architecture guarantees the persistence of metadata even when the related data objects no longer exist. Finally, we employ ad-hoc data anonymization and knowledge mapping data structures to comply with domain-relevant community standards.

3 ARCHITECTURE

In this section, we describe a novel SRA for implementing a data sharing repository compliant with the FAIR Principles. Section 3.1 introduces the layers of this architecture. Section 3.2 describes how these layers comply with the FAIR Principles.

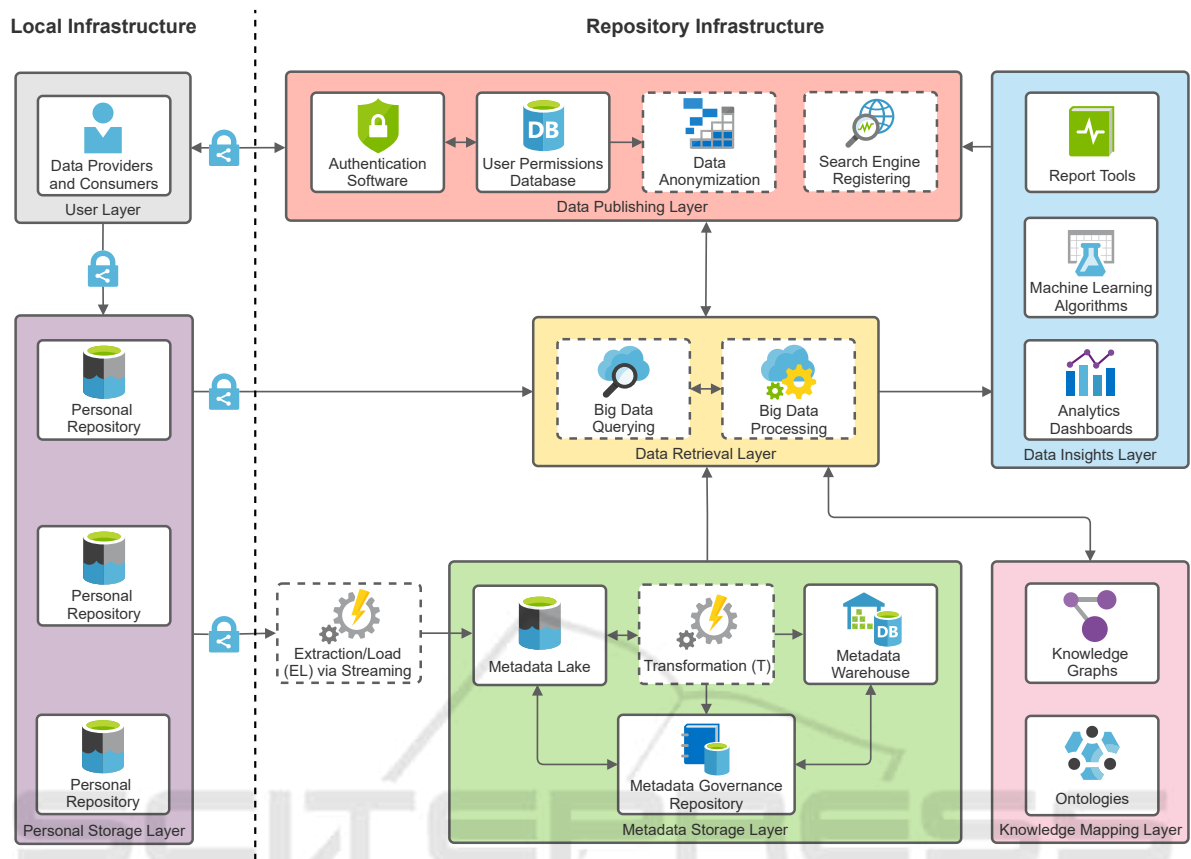


Figure 1: The proposed architecture for implementing a data sharing repository compliant with the FAIR Principles.

3.1 Layers

Figure 1 depicts our proposed architecture, which consists of the following layers: (i) User; (ii) Personal Storage; (iii) Metadata Storage; (iv) Data Retrieval; (v) Knowledge Mapping; (vi) Data Insights; and (vii) Data Publishing. In these layers, components are represented by boxes with a solid border, whereas processes are represented by boxes with a dashed border. Furthermore, directional arrows represent the flow of data through the different layers, components, and processes. Whenever the depiction of these arrows include a padlock, the flow of data must be end-to-end encrypted. This is due to the fact that our architecture employs ad-hoc data anonymization based on user permissions; thus, it is important to guarantee that non anonymized data travels the network with enhanced security (Puthal et al., 2017).

User Layer. Encompasses the users that interact with the data repository and their respective environment. Users can assume one or both of the following roles: (i) data providers, which are responsible for loading their own personal repository with their research data and respective metadata, if available; and (ii) data

consumers, which can consume different types of data from the repository. Data providers interact only with the Personal Storage Layer, where their personal repositories are located. On the other hand, data consumers interact only with the Data Publishing Layer, providing data requests and their credentials for authentication, depending on the nature of the request.

Personal Storage Layer. Comprises a set of repositories built to store research data objects along with their respective metadata. Each personal repository is owned by a different data provider and can be implemented by using any available technology (e.g. a data lake). Thus, the repositories can be autonomous, geographically distributed, and heterogeneous. Our architecture requires every personal repository to be connected to a data streaming application programming interface (API) responsible to send every novel metadata entry to the metadata storage layer in real time (e.g., Apache Kafka (Le Noac’h et al., 2017)). This is an important characteristic of the architecture since it enables analyses involving researches that are constantly generating data in short intervals. It is also imperative for the personal repositories to have an interface that allows the Data Retrieval Layer to capture

research data objects when requested by data consumers.

Metadata Storage Layer. Stores the metadata constantly extracted from the Personal Storage Layer. To this end, we employ two types of repositories: Metadata Warehouse and Metadata Lake. Due to its historical and non-volatile characteristics, the Metadata Warehouse keeps the metadata alive even when their related data objects no longer exist. It is also designed to support batch metadata querying to provide different types of big data insights, as detailed in Section 4. The Metadata Lake is responsible for instantly storing the raw metadata obtained via streaming from the Personal Storage Layer until it is completely transformed and loaded into the Metadata Warehouse. The Metadata Lake supports real time metadata querying and also serves as a data staging area, storing the intermediate results of the metadata transformations to enhance performance. All metadata provenance, such as data about the personal repositories that provide the metadata and details about the extraction (E), load (L), and transformation (T) processes, are maintained in the Metadata Governance Repository. This repository also contains metadata about the metadata extracted from the Personal Storage Layer, i.e., a set of data that describes and gives information about the metadata.

Data Retrieval Layer. Provides resources to processing and querying tasks, and is considered the core of the architecture. It uses the repository big data infrastructure along with a parallel and distributed framework (e.g., Apache Spark (Zaharia et al., 2010)) to perform several tasks, such as: (i) retrieving source research data objects from the Personal Storage Layer based on the content of the metadata stored in the Metadata Storage Layer; (ii) performing ad-hoc data anonymization (Bazai et al., 2021) in the original research data objects and in their associated metadata according to the specifications sent by the Data Publishing Layer; (iii) using the ontologies and knowledge graphs from the Knowledge Mapping Layer to translate user data requests to the data model adopted by the Metadata Storage Layer; and (iv) accessing the content of every other layer to generate different types of intelligence for the Data Insights Layer.

Knowledge Mapping Layer. We employ knowledge mapping data structures, such as ontologies (Staab et al., 2001) and knowledge graphs (Ehrlinger and Wöß, 2016) for different purposes. For instance, these structures can serve as a mapping between a generic data model known by data consumers and the data model implemented by the Metadata Storage Layer. This is a very important requirement to data con-

sumers since it enables data retrieval by using well-known standards. Further, the models stored in the Knowledge Mapping Layer can also be applied to delineate different data relationships, enabling the generation of several types of big data insights.

Data Insights Layer. Enables the generation of descriptive, predictive, and prescriptive analyses (Lepeniotti et al., 2020) by using the content stored in the Metadata Storage, Personal Storage, and Knowledge Mapping layers. The insights created by this layer to support big data analytics can include, but are not limited to: (i) public dashboards that enhance the publicity of the scientific data stored in the repository, making it more findable; (ii) private dashboards used by managers and directors to monitor the repository; (iii) machine learning models, which can be employed to perform predictive and prescriptive analyses; and (iv) report tools that generate predefined reports from the stored data. The created insights are accessed only through the Data Publishing Layer to guarantee security and anonymization.

Data Publishing Layer. Serves as a single access point for data consumers to obtain any type of data from the repository. It is responsible for authenticating data consumers, receiving their data requests, and, based on their permissions, sending the requests to the Data Retrieval Layer along with data anonymization specifications. It also returns the metadata requested to data consumers as soon as they are processed by the Data Retrieval Layer or the Data Insights Layer. Further, the Data Publishing Layer supports mechanisms for indexing the metadata stored in the repository in a search engine to provide increased findability.

Because of the multiple possible contexts behind the implementation of a repository compliant with the FAIR Principles, it is not mandatory to instantiate every component in the architecture. Data engineers should choose the appropriate layers and components according to the characteristics of the environment in which the architecture is being employed.

3.2 Compliance with the FAIR Principles

Table 2 describes the requirements related to the FAIR Principles of Findability, Accessibility, Interoperability, and Reusability. It also highlights which layers of the proposed SRA (Figure 1) fulfill each requirement.

The Findability requirements are mostly satisfied by the Metadata Storage Layer. This layer contains the Metadata Warehouse, which is responsible for storing the metadata (F2) and its identifier (F1), as well as maintaining the identifier of the data instance

Table 2: Relationship between the architecture layers and the FAIR Principles. Requirements reproduced from Wilkinson et al. (2016).

| Principles | Requirements | Architecture Layers |
|------------------|--|---|
| Findability | F1. Data and metadata are assigned a globally unique and persistent identifier. | Personal Storage Metadata Storage |
| | F2. Data is described with rich metadata. | Metadata Storage |
| | F3. Metadata clearly and explicitly include the identifier of the data it describes. | Metadata Storage |
| | F4. Data and metadata are registered or indexed in a searchable resource. | Data Publishing |
| Accessibility | A1. Data and metadata are retrievable by their identifier using a standardized communications protocol. | Data Publishing Data Retrieval Metadata Storage Personal Storage |
| | A1.1. The protocol is open, free, and universally implementable. | Data Publishing |
| | A1.2. The protocol allows for an authentication and authorization procedure, where necessary. | Data Publishing |
| | A2. Metadata is accessible, even when the data is no longer available. | Metadata Storage |
| Interoperability | I1. Data and metadata use a formal, accessible, shared, and broadly applicable language for knowledge representation. | Knowledge Mapping |
| | I2. Data and metadata use vocabularies that follow FAIR Principles. | Knowledge Mapping |
| | I3. Data and metadata include qualified references to other data and metadata. | Knowledge Mapping |
| Reusability | R1. Data and metadata are richly described with a plurality of accurate and relevant attributes. | Metadata Storage |
| | R1.1. Data and metadata are released with a clear and accessible data usage license. | Metadata Storage |
| | R1.2. Data and metadata are associated with detailed provenance. | Metadata Storage |
| | R1.3. Data and metadata meet domain-relevant community standards. | Metadata Storage Knowledge Mapping |

to which the metadata refers (F3). These identifiers can be implemented as unique fields, such as primary keys in relational databases. Two other layers also enable findability. The Personal Storage Layer assigns an identifier for every data instance (F1) and the Data Publishing Layer is responsible for registering the repository in search engines (F4).

Regarding the Accessibility requirements, they fall mostly under the responsibility of the Data Publishing Layer since it handles user connection (A1, A1.1) and authentication (A1.2). The Metadata Storage Layer also plays a significant role since the Metadata Warehouse keeps the metadata alive even when the source data is no longer available (A2). Finally, the Data Retrieval, Metadata Storage, and Personal Storage layers enable data retrieval based on its unique identifier (A1).

The Knowledge Mapping Layer must be employed to enable the Interoperability requirements. This layer handles the translation between schemas in the Metadata Storage and Personal Storage layers to a well-known language (I1) and vocabulary (I2). It is also responsible for storing the relationships between different instances of metadata and data objects (I3).

Finally, the Reusability requirements are tackled as follows. In the Metadata Storage Layer, the Metadata Warehouse enables a rich description of the data objects and their metadata (R1), encompassing a dimension to deal with licensing information (R1.1) and a Metadata Governance Repository to store data provenance (R1.2). Further, the compliance with domain-relevant community standards (R1.3) is obtained by the Metadata Warehouse or through a mapping stored in the Knowledge Mapping Layer.

Our architecture goes one step forward in regards to the FAIR Principles since it also enables big data analytics. The Metadata Warehouse and the Metadata Lake store huge volumes of metadata that can be retrieved efficiently by the Data Insights Layer. The Metadata Warehouse supports the traditional batch analytical query processing, while the Metadata Lake is responsible for the streaming query processing to monitor and extract knowledge in (almost) real time. Therefore, the architecture contributes to the generation of a broad set of data insights to support data consumers in the decision-making process. We show this applicability by using real-world datasets of COVID-19 Brazilian patients, as discussed in Section 5.

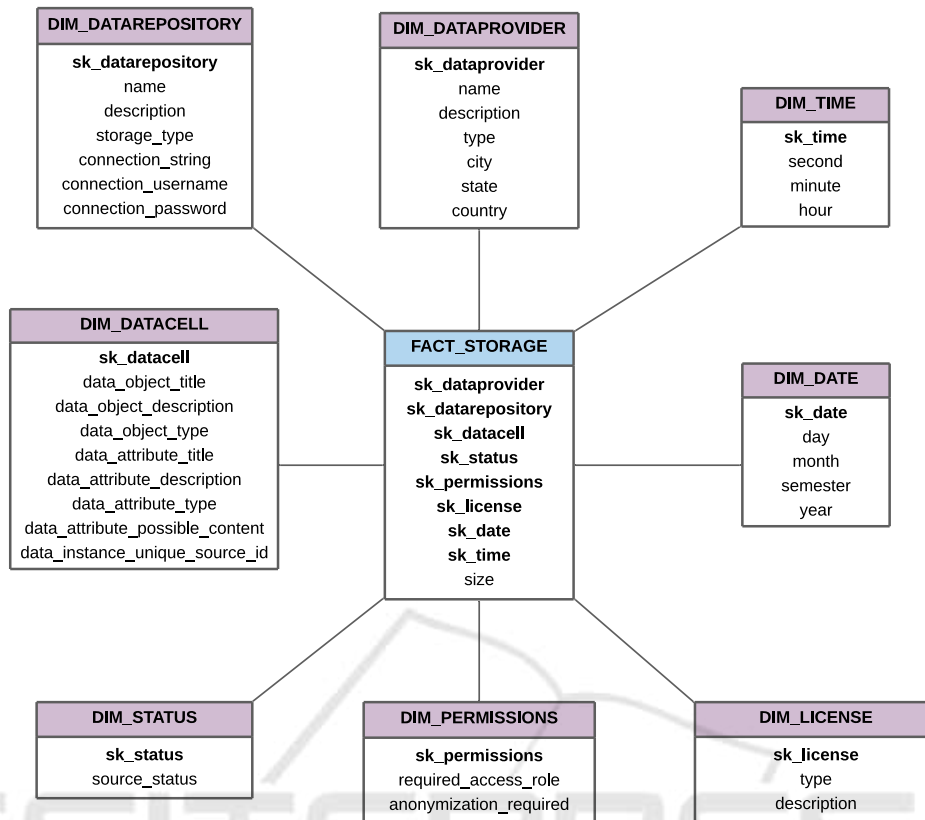


Figure 2: The proposed metadata warehouse generic model.

4 METADATA WAREHOUSE GENERIC MODEL

To effectively implement a data warehouse, we must first model its numeric measures and dimensions. Numeric measures are the subjects of interest. Dimensions are described by a set of attributes and determine the context for these measures. In relational implementations, the data warehouse is designed through a star schema composed of fact and dimension tables, corresponding to the numeric measures and dimensions, respectively (Kimball and Ross, 2011).

We propose a generic model for the Metadata Warehouse that encompasses the needed metadata for a repository to be compliant with the FAIR Principles. This model is composed of eight dimension tables and one fact table, as shown in Figure 2. More dimensions or attributes can be included depending on the scope of the repository being modeled.

The fact table **FACT_STORAGE** represents the event of extracting the metadata of a data cell at a given date and time, considering the data repository, the data provider, and the associated status, permis-

sions, and license. A data cell can be defined as the intersection of an attribute and a tuple, such as the value of a column in a row for a relational table, or the value of a field in a document for a document collection. The fact table contains the surrogate keys of every dimension, enabling different perspectives of analysis. The set of these surrogate keys also composes the primary key. The fact represents the size of the data cell, which can be expressed in characters, bytes, or similar measurement units. It is an additive numeric measure, indicating that the size can be summed across all dimensions. Thus, it can be useful to support analyses regarding the growth of the repository over time. Other numeric measures can also be included in the fact table, depending on the context of the repository being implemented.

The following dimension tables are associated with the fact table: (i) **DIM_DATAPROVIDER**, storing information on the provider of the data cell, such as its name, type, and location; (ii) **DIM_DATAREPOSITORY**, containing the data necessary to connect to the repository, its description, and storage type (e.g., PostgreSQL storage); (iii) **DIM_DATACELL**, keeping the metadata related

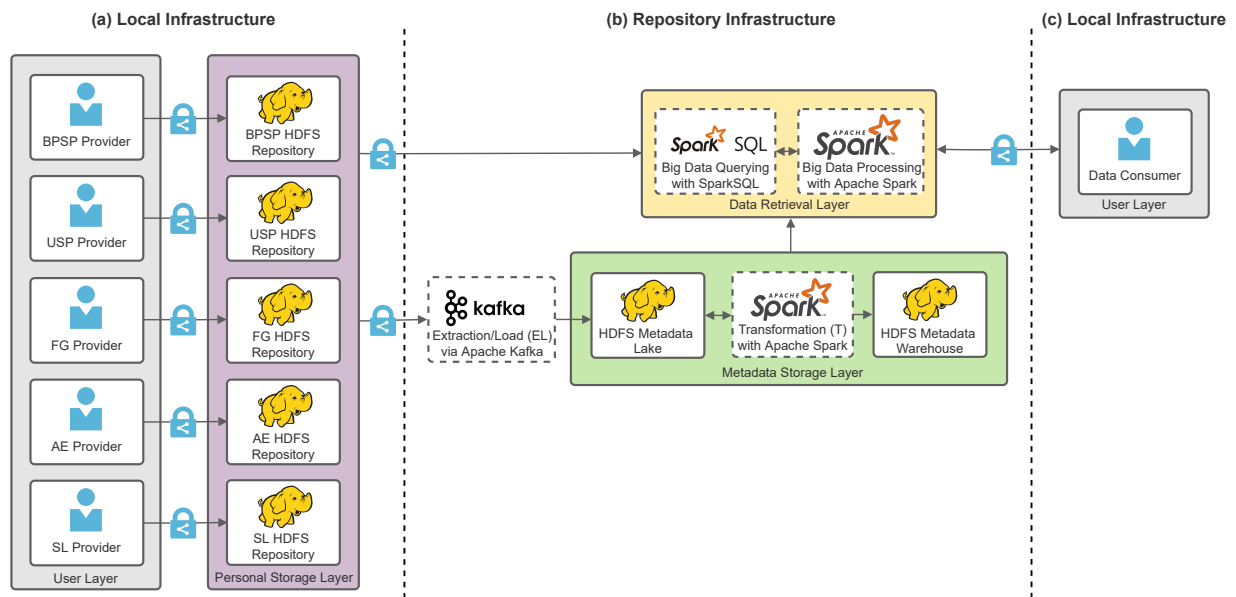


Figure 3: Architecture instantiation for the context of the COVID-19 DataSharing/BR dataset.

to the source data objects (e.g., a relational table), its attributes (e.g., a column in a relational table), and its instances (e.g., a row in a relational table); (iv) DIM_STATUS, storing the status of the data cell in the data source, specifying if it still exists; (v) DIM_PERMISSIONS, maintaining the required role to access a data cell and a boolean attribute to inform if data anonymization is required for the access; (vi) DIM_LICENSE, containing the data cell licensing information; and (vii) DIM_DATE and DIM_TIME, representing respectively the date and time in which the data cell metadata has been extracted from the Personal Storage Layer.

The use of the metadata warehouse generic model depicted in Figure 2 is very important to achieve the FAIR Principles in a data sharing repository. For instance, it enables data objects to be associated with rich metadata, keeping it persisted even when these objects no longer exist. Furthermore, due to the intrinsic characteristics of a data warehouse (i.e. subject oriented and integrated), analytical queries on the stored metadata are considerably optimized, an essential characteristic for a big data analytics environment. Analyses involving multiple perspectives are also enabled, not only due to the plurality of the dimensions incorporated in the model but also due to the fact that the data size is stored in the lowest possible granularity level. Finally, since the model does not contain any components that are specific to a particular repository, it is generic enough to be reused in distinct contexts.

5 CASE STUDY

In this section, we present a case study to show how our architecture and the metadata warehouse generic model can be deployed to enable scientific data sharing according to the FAIR Principles in a real-world context. Our goal is to not conduct performance evaluations since it goes beyond the scope of this paper. Section 5.1 discusses how to instantiate the architecture to the given context. Section 5.2 describes different analytical queries that data consumers can execute on top of this instantiation.

5.1 Architecture Instantiation

We employ a real-world dataset of COVID-19 Brazilian patients. The dataset is available in the COVID-19 DataSharing/BR repository (FAPESP, 2020), which is a FAIR-compliant Open Science repository developed by the State of São Paulo Research Foundation (FAPESP). Every data object available in this repository is accompanied by its metadata, such as a data dictionary that describes the data type and the meaning of every one of its attributes.

There are five distinct data providers, all references in the field of medical diagnosis in Brazil: (i) Beneficência Portuguesa de São Paulo (BPSP); (ii) the University of São Paulo clinics hospital (USP); (iii) the Fleury Group clinics (FG); (iv) the Albert Einstein hospital (AE); and (v) the Syrian-Lebanese hospital (SL). Each data provider contributed with

three different data objects: (i) patients data, including their unique identification (ID), sex, birth date, country, state, city, and zip code; (ii) medical exams, including the patient’s unique ID, consultation ID, collection date, collection venue, analyte description, exam description, result, measurement unit, and reference value; and (iii) outcome, whose fields are the patient’s unique ID, consultation ID, consultation date, consultation type, clinic ID, clinic description, outcome date, and outcome description. The content of these data objects were translated from Portuguese to English to provide readability. Considering all data providers, there is a total of 862,571 patients, 54,763,675 exams, and 307,928 outcomes.

Figure 3 depicts the layers and respective components that are required to instantiate the architecture shown in Figure 1 according to the characteristics of the case study. The aforementioned five data providers are represented in the User Layer drawn on the left (Figure 3a). The COVID-19 DataSharing/BR dataset is composed of comma-separated values (CSV) and Microsoft Excel sheet (XLSX) files. Therefore, in the Personal Storage Layer (Figure 3a), we choose to store each data provider using a different Hadoop Distributed File System (HDFS) (Shvachko et al., 2010) environment so that repositories owned by different data providers are properly represented.

The Metadata Lake and Warehouse of the Metadata Storage Layer (Figure 3b) are also implemented by using the HDFS. We employ Apache Kafka ((Le Noac’h et al., 2017)) to extract every new instance of metadata inserted into the Personal Storage Layer (Figure 3a) and to load it into the Metadata Lake. Furthermore, we employ Apache Spark to transform the content of the Metadata Lake and load it into the Metadata Warehouse for further analyses. The design of the Metadata Warehouse follows the model proposed in Section 4. Thus, every fact and dimension depicted in Figure 2 is implemented.

Data consumers, represented in the User Layer drawn on the right (Figure 3c), can issue different types of requests. Three examples of analytical queries involving the stored metadata and the source data objects to which they refer are described in Section 5.2. Data consumers issue queries using the Structured Query Language (SQL). The queries are then executed by the Data Retrieval Layer (Figure 3b) in a parallel and distributed manner through the use of SparkSQL (Armbrust et al., 2015). Because of the characteristics of this interaction, the case study does not require the instantiation of any data structure from the Knowledge Mapping Layer.

Finally, since the COVID-19 DataSharing/BR dataset has its personal and sensitive data already

anonymized, the Data Retrieval Layer (Figure 3b) does not need to be concerned with ad-hoc data anonymization. We also consider that data consumers have all the needed permissions to query the repository data, which renders the instantiation of the Data Publishing Layer unnecessary.

5.2 Analytical Queries

We describe three analytical queries that data consumers can execute on top of the instantiated architecture outlined in Section 5.1. The motivation behind these queries is to validate the communication between multiple layers of the proposed architecture. The validation encompasses the task of integrating the source data objects stored in the Personal Storage Layer) with their respective metadata stored in the Metadata Storage layer. We propose different types of queries, i.e., queries that analyze different aspects in the decision-making process:

- Query 1. Involves only metadata stored in the Metadata Warehouse.
- Query 2. Encompasses only source data objects.
- Query 3. Includes both the stored metadata and the source data objects.

These types of queries are generic and can be applied to different contexts. We present specific examples of these queries in our scenario as follows.

Query 1. Analyzing Data Size Grouped by Year, Month, and Data Provider Type. This type of query allows data consumers to verify which type of data provider bestows the majority of the data to the repository over time. Hence, it can be useful to analyze the growth of the repository. Since this analysis involves only the stored metadata, the Data Retrieval Layer can perform it by executing the following SparkSQL query against the Metadata Warehouse:

```
SELECT DIM_DATE.year,
       DIM_DATE.month,
       DIM_DATAPROVIDER.type,
       SUM(FACT.size) AS size
FROM FACT_STORAGE
  INNER JOIN DIM_DATE
    ON (FACT.sk_date =
        DIM_DATE.sk_date)
  INNER JOIN DIM_DATAPROVIDER
    ON (FACT.sk_dataprovider =
        DIM_DATAPROVIDER.sk_dataprovider)
GROUP BY DIM_DATE.year, DIM_DATE.month,
         DIM_DATAPROVIDER.type
```

The results of Query 1 are depicted in Figure 4. Through the interpretation of the results, data consumers can verify that most of the data in the repository has been provided by laboratories in 2020. It is

also possible to identify that no laboratory has provided new data in February and April 2021. With this information, data consumers can work on prospecting new laboratory data providers, as well as on requesting more data from the current ones.

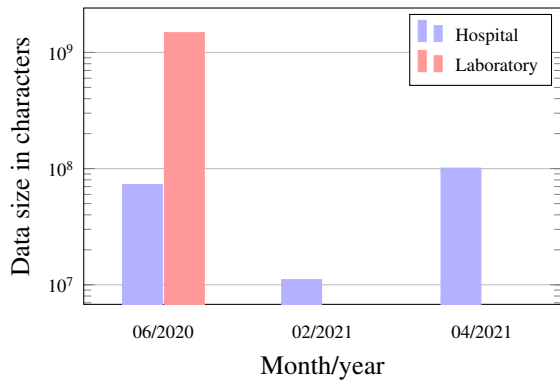


Figure 4: Results of Query 1, which represent data size grouped by month, year, and data provider type. A logarithmic scale is employed to improve data visualization.

Query 2. Analyzing the Amount of Patients That Were Tested for Calcium Grouped by Sex using the Dataset Bestowed by the USP Provider. This type of query inspects if there is any relationship between the patient sex and the types of exams performed. Even though this investigation encompasses only source data objects, the Data Retrieval Layer must first access the Metadata Warehouse to obtain the connection information to the Personal Storage Layer. Once this information is retrieved and used as a parameter to load the source data objects, data consumers can run the following SparkSQL query:

```
SELECT USP_PATIENTS.ic_sex,
       COUNT(DISTINCT
             USP_PATIENTS.id_patient) AS amount
FROM USP_PATIENTS
INNER JOIN USP_EXAMS
ON (USP_PATIENTS.id_patient =
    USP_EXAMS.id_patient)
WHERE USP_EXAMS.de_exam
      LIKE '%CALCIUM%'
GROUP BY USP_PATIENTS.ic_sex
```

By analyzing the results of Query 2 depicted in Figure 5, it becomes clear that the majority of patients tested for calcium in the USP dataset are male. With this insight, data consumers can perform further analyses to confirm if there really is a correlation between the patient sex and the exams performed. For instance, it is needed to verify the proportion of men and women in the dataset, as well as if this behavior remains unchanged in other data providers' datasets.

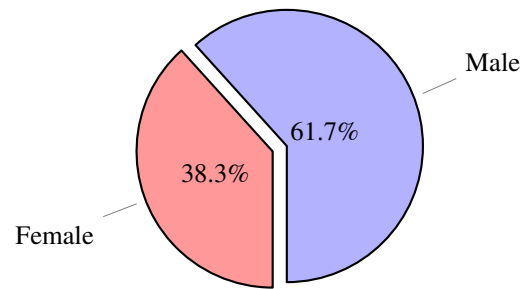


Figure 5: Results of Query 2, which represent the amount of patients that were tested for calcium grouped by sex using the dataset bestowed by the USP provider.

Query 3. Analyzing the Five Most Voluminous Data Sizes of Outcomes Registered in Emergency Rooms, Grouped by Data Provider and Clinic Name. Data consumers can use this type of investigation to generate insights to reveal which data providers occupy the most repository space with outcomes registered in emergency rooms. It is also possible to verify in which of the data providers clinics the data size is bigger. Since this query involves stored metadata and source data objects, the Data Retrieval Layer must access the Metadata Warehouse twice: first for retrieving the connection information to the Personal Storage Layer, and then for joining the source data objects with their respective metadata. Once the data objects are loaded, the following SparkSQL query is executed:

```
SELECT DIM_DATAPROVIDER.name,
       OUTCOMES.clinic,
       SUM(FACT.size) AS size
FROM FACT_STORAGE
INNER JOIN DIM_DATAPROVIDER
ON (FACT.sk_dataprovider =
    DIM_DATAPROVIDER.sk_dataprovider)
INNER JOIN DIM_DATACELL
ON (FACT.sk_datacell =
    DIM_DATACELL.sk_datacell)
INNER JOIN OUTCOMES
ON (OUTCOMES.id_patient =
    DIM_DATACELL
     .data_instance_unique_source_id_01
    AND OUTCOMES.id_consultation =
    DIM_DATACELL
     .data_instance_unique_source_id_02
    AND OUTCOMES.data_object_title =
    DIM_DATACELL.data_object_title)
WHERE OUTCOMES.examination_description
      LIKE '%Emergency%'
GROUP BY DIM_DATAPROVIDER.name,
         OUTCOMES.clinic
ORDER BY SUM(FACT.size) DESC
LIMIT 5
```

The results of Query 3 are depicted in Figure 6. By interpreting these results, data consumers can obtain different insights. For instance, they can observe

that only clinics belonging to the BPSP and the SL data providers are displayed in the query results. This can indicate that, in regards to outcomes registered in emergency rooms, there is a considerable gap between the data size of these clinics and those belonging to other data providers. Additionally, data consumers can realize that the significant majority of the data referring to these outcomes have been registered by two BPSP clinics: “U.Pa.” and “U.B.M.”. With this information, it is possible to investigate the reasons behind these outliers. For example, data consumers can verify if these specific clinics store more details regarding outcomes in emergency rooms when compared to clinics with smaller data sizes. This information can be useful to encourage other clinics to increase the level of detail in their data, enriching future analyses that encompass this context.

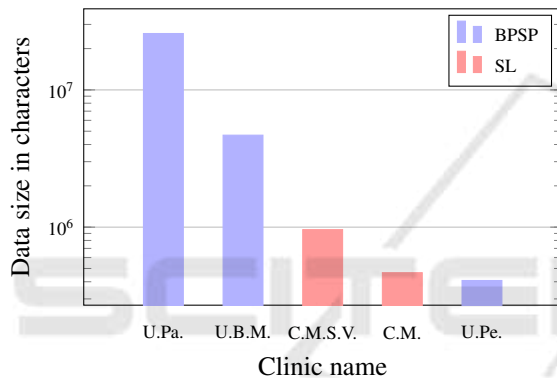


Figure 6: Results of Query 3, which represent the data size of outcomes registered in emergency rooms, grouped by data provider and clinic name. A logarithmic scale is employed to improve data visualization.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a software reference architecture to implement data sharing repositories compliant with the FAIR Principles. This architecture is composed of seven layers: (i) User Layer, representing data providers and consumers; (ii) Personal Storage Layer, encompassing the source data objects; (iii) Metadata Storage Layer, responsible for storing and maintaining the metadata extracted from the personal storage layer; (iv) Data Retrieval Layer, responsible for querying and processing data and metadata; (v) Knowledge Mapping Layer, containing associations between the repository data models and domain-relevant community standards; (vi) Data Insights Layer, aimed to generate different types of analyses from data and metadata; and (vii) Data Pub-

lishing Layer, representing a single access point for data consumers to retrieve any type of data, metadata, or insights from the repository. We detail which layer fulfills each FAIR Principles requirement.

We also propose a metadata warehouse model that can be employed by data engineers to guarantee metadata persistence, i.e., to guarantee that metadata remains alive even when their corresponding source data objects no longer exist. This model is generic and can be adapted in the design of distinct repositories, according to the data consumers’ requirements.

Finally, we describe a case study that instantiates the proposed architecture to the context of a real-world dataset of COVID-19 Brazilian patients, available in the COVID-19 DataSharing/BR repository. We detail three different types of queries and highlight their importance to big data analytics.

We are currently developing guidelines to assist data engineers in the process of implementing the proposed architecture. Another future work includes validating the efficiency of the architecture through performance tests that investigate several aspects, such as query response time, scalability, and memory throughput. New case studies instantiating the proposed architecture to different real-world contexts are also planned as future work.

ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

REFERENCES

- Angelov, S., Grefen, P., and Greefhorst, D. (2012). A framework for analysis and design of software reference architectures. *Inf Softw Technol*, 54(4):417–431.
- Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., Meng, X., Kaftan, T., Franklin, M. J., Ghodsi, A., and Zaharia, M. (2015). Spark SQL: Relational data processing in Spark. In *Proc. ACM SIGMOD*, pages 1383–1394.
- Bazai, S. U., Jang-Jaccard, J., and Alavizadeh, H. (2021). Scalable, high-performance, and generalized subtree data anonymization approach for Apache Spark. *Electronics*, 10(5).
- Chen, M., Mao, S., and Liu, Y. (2014). Big data: A survey. *Mob Netw Appl*, 19(2):171–209.
- Couto, J., Borges, O., Ruiz, D., Marczak, S., and Prikladnicki, R. (2019). A mapping study about data lakes: An improved definition and possible architectures. In *Proc. SEKE*, pages 453–458.

- Davoudian, A. and Liu, M. (2020). Big data systems: A software engineering perspective. *ACM Comput Surv*, 53(5):1–39.
- Delgado, J. and Llorente, S. (2021). FAIR aspects of a genomic information protection and management system. In *Proc. EFMI STC*, pages 50–54.
- Devarakonda, R., Prakash, G., Guntupally, K., and Kumar, J. (2019). Big federal data centers implementing FAIR data principles: ARM data center example. In *Proc. IEEE Big Data*, pages 6033–6036.
- Ehrlinger, L. and Wöß, W. (2016). Towards a definition of knowledge graphs. In Martin, M., Cuquet, M., and Folmer, E., editors, *Proc. Posters and Demos Track of SEMANTiCS*, volume 1695 of *CEUR Workshop Proceedings*.
- FAPESP (2020). COVID-19 Data Sharing/BR. Available at <https://repositoriodatasharingfapesp.uspdigital.usp.br>.
- Fernandez, R. C., Pietzuch, P. R., Kreps, J., Narkhede, N., Rao, J., Koshy, J., Lin, D., Riccomini, C., and Wang, G. (2015). Liquid: Unifying nearline and offline big data integration. In *Proc. CIDR*.
- Kimball, R. and Ross, M. (2011). *The data warehouse toolkit: the complete guide to dimensional modeling*. John Wiley & Sons.
- Kreps, J. (2014). Questioning the Lambda architecture. Available at <https://www.oreilly.com/radar/questioning-the-lambda-architecture/>.
- Lannom, L., Koureas, D., and Hardisty, A. R. (2020). FAIR data and services in biodiversity science and geoscience. *Data Intell*, 2(1-2):122–130.
- Le Noac’h, P., Costan, A., and Bougé, L. (2017). A performance evaluation of Apache Kafka in support of big data streaming applications. In *Proc. IEEE Big Data*, pages 4803–4806.
- Lepenioti, K., Bousdekis, A., Apostolou, D., and Mentzas, G. (2020). Prescriptive analytics: Literature review and research challenges. *Int J Inf Manage*, 50:57–70.
- Martínez-Prieto, M. A., Cuesta, C. E., Arias, M., and Fernández, J. D. (2015). The solid architecture for real-time management of big semantic data. *Future Gener Comput Syst*, 47:62–79.
- Medeiros, C. B., Darboux, B. R., Sánchez, J. A., Tenkanen, H., Meneghetti, M. L., Shinwari, Z. K., Montoya, J. C., Smith, I., McCray, A. T., and Vermeir, K. (2020). *IAP input into the UNESCO Open Science Recommendation*. Available at https://www.interacademies.org/sites/default/files/2020-07/Open_Science.0.pdf.
- Nadal, S., Herrero, V., Romero, O., Abelló, A., Franch, X., Vansummeren, S., and Valerio, D. (2017). A software reference architecture for semantic-aware big data systems. *Inf Softw Technol*, 90:75–92.
- Nakagawa, E. Y., Antonino, P. O., and Becker, M. (2011). Reference architecture and product line architecture: A subtle but critical difference. In *Proc. ECSA*, pages 207–211.
- Pommier, C., Michotey, C., Cornut, G., Roumet, P., Duchêne, E., Flores, R., Lebreton, A., Alaux, M., Durand, S., Kimmel, E., et al. (2019). Applying FAIR principles to plant phenotypic data management in GnpIS. *Plant Phenomics*.
- Puthal, D., Nepal, S., Ranjan, R., and Chen, J. (2017). A synchronized shared key generation method for maintaining end-to-end security of big data streams. In *Proc. HICSS*, pages 6011–6020.
- Sawadogo, P. and Darmont, J. (2021). On data lake architectures and metadata management. *J Intell Inf Syst*, 56(1):97–120.
- Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The Hadoop distributed file system. In *Proc. IEEE MSST*, pages 1–10.
- Sinaci, A. A., Núñez-Benjumea, F. J., Gencturk, M., Jauer, M.-L., Deserno, T., Chronaki, C., Cangiali, G., Caverio-Barca, C., Rodríguez-Pérez, J. M., Pérez-Pérez, M. M., et al. (2020). From raw data to FAIR data: the fairification workflow for health research. *Methods Inf Med*, 59(S1):21–32.
- Staab, S., Studer, R., Schnurr, H.-P., and Sure, Y. (2001). Knowledge processes and ontologies. *IEEE Intell Syst*, 16(1):26–34.
- Vaisman, A. and Zimányi, E. (2014). *Data warehouse systems*. Springer.
- Warren, J. and Marz, N. (2015). *Big data: Principles and best practices of scalable realtime data systems*. Simon and Schuster.
- Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., Santos, L. B. S., Bourne, P. E., et al. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data*, 3(1):1–9.
- Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: Cluster computing with working sets. In *Proc. USENIX HotCloud*.