

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Yan Mendes Ferreira

**Polyflow: a Polystore-compliant Mechanism to Provide Interoperability to
Heterogeneous Provenance Graphs**

Juiz de Fora

2020

Yan Mendes Ferreira

**Polyflow: a Polystore-compliant Mechanism to Provide Interoperability to
Heterogeneous Provenance Graphs**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. D.Sc. Victor Ströele

Coorientador: Prof. D.Sc. Daniel de Oliveira

Juiz de Fora

2020

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Ferreira, Yan Mendes.

Polyflow : a Polystore-compliant Mechanism to Provide Interoperability
to Heterogeneous Provenance Graphs / Yan Mendes Ferreira. – 2020.
66 f. : il.

Orientador: Prof. D.Sc. Victor Ströele

Coorientador: Prof. D.Sc. Daniel de Oliveira

Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação, 2020.

1. polystore. 2. syntactic interoperability. 3. semantic interoperability.
I. Ströele, Victor, orient. II. de Oliveira, Daniel, coorient. III. Título.

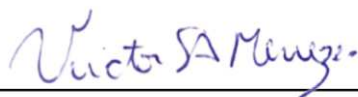
Yan Mendes Ferreira

**Polyflow: a Polystore-compliant Mechanism to Provide Interoperability to
Heterogeneous Provenance Graphs**

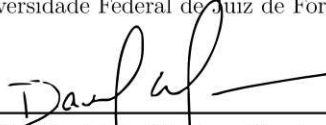
Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Aprovada em: 13 de Novembro de 2020

BANCA EXAMINADORA



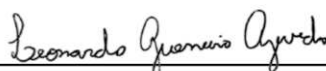
Prof. D.Sc. Victor Ströele - Orientador
Universidade Federal de Juiz de Fora



Prof. D.Sc. Daniel de Oliveira - Coorientador
Universidade Federal Fluminense



Prof. D.Sc. Regina Maria Maciel Braga Villela
Universidade Federal de Juiz de Fora



Prof. D.Sc. Leonardo Azevedo
IBM Research

ACKNOWLEDGEMENTS

Aos meus pais, pelo sustento, encorajamento e apoio.

Aos professores Victor e Daniel pela orientação, amizade e principalmente, pela paciência, sem a qual este trabalho não se realizaria.

Aos professores do Programa de Pós Graduação em Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

“The more I study, the more insatiable do I feel my genius for it to be.”
(Ada Lovelace)

RESUMO

Muitos experimentos científicos são modelados como *workflows* (fluxos de trabalho). *Workflows* produzem comumente um grande volume de dados. De forma a garantir a reprodutibilidade desses *workflows*, estes geralmente são orquestrados por Sistemas de Gerência de *Workflows* (SGWfs), garantindo que dados de proveniência sejam capturados. Dados de proveniência representam o histórico de derivação de um dado ao longo da execução do *workflow*. Assim, o histórico de derivação dos dados pode ser representado por meio de um grafo de proveniência. Este grafo possibilita aos cientistas analisarem e avaliarem resultados produzidos por um *workflow*. Todavia, cada SGWf tem seu formato proprietário de representação para dados de proveniência, e os armazenam em diferentes granularidades. Consequentemente, em cenários mais complexos em que um cientista precisa analisar de forma integrada grafos de proveniência gerados por múltiplos *workflows*, isso se torna desafiador. Primeiramente, para entender o campo de pesquisa, realizamos um Mapeamento Sistemático da Literatura, avaliando soluções existentes sob diferentes lentes. Com uma compreensão mais clara do atual estado da arte, propomos uma ferramenta chamada **Polyflow**, inspirada em conceitos de sistemas *Polystore*, possibilitando a integração de várias bases de dados heterogêneas por meio de uma interface de consulta única que utiliza o ProvONE como *schema* global. **Polyflow** permite que cientistas submetam consultas em múltiplos grafos de proveniência de maneira integrada. **Polyflow** foi avaliado em conjunto com especialistas usando dados de proveniência coletados de *workflows* reais que apoiam o estudo de geração de árvores filogenéticas. O resultado da avaliação mostrou a viabilidade do **Polyflow** para interoperar semanticamente dados de proveniência gerado por distintos SGWfs, tanto do ponto de vista de desempenho quanto de usabilidade.

Palavras-chave: *Polystore*, Interoperabilidade sintática, Interoperabilidade semântica.

ABSTRACT

Many scientific experiments are modeled as workflows. Workflows usually output massive amounts of data. To guarantee the reproducibility of workflows, they are usually orchestrated by Workflow Management Systems (WfMS), that capture provenance data. Provenance represents the lineage of a data fragment throughout its transformations by activities in a *workflow*. Provenance traces are usually represented as graphs. These graphs allows scientists to analyze and evaluate results produced by a workflow. However, each WfMS has a proprietary format for provenance and do it in different granularity levels. Therefore, in more complex scenarios in which the scientist needs to interpret provenance graphs generated by multiple WfMSs and workflows, a challenge arises. To first understand the research landscape, we conduct a Systematic Literature Mapping, assessing existing solutions under several different lenses. With a clearer understanding of the state of the art, we propose a tool called **Polyflow**, which is based on the concept of Polystore systems, integrating several databases of heterogeneous origin by adopting a global ProvONE schema. **Polyflow** allows scientists to query multiple provenance graphs in an integrated way. **Polyflow** was evaluated by experts using provenance data collected from real experiments that generate phylogenetic trees through workflows. The experiment results suggest that **Polyflow** is a viable solution for interoperating heterogeneous provenance data generated by different WfMSs, from both a usability and performance standpoint.

Keywords: Polystore, Syntactic interoperability, Semantic interoperability.

LIST OF FIGURES

Figure 1 – ProvONE model adapted from (VÍCTOR LUDÄSCHER BERTRAM, 2016)	17
Figure 2 – BigDAWG’s architectural design. Inspired in (DUGGAN et al., 2015) .	18
Figure 3 – (a) Physical approach to database integration. (b) Logical approach to database integration. Both figures adapted from (COULOURIS; DOLLIMORE; KINDBERG, 2005)	19
Figure 4 – Timeline of publications. Edges connecting two papers represent a citation. Works in red are the product of snowballing; the ones in blue were found in the knowledge bases and the yellow ones were derived from the ad-hoc complimentary search.	29
Figure 5 – Vehicle type in which the papers were published.	30
Figure 6 – Number of citations each paper has on July, 2020.	30
Figure 7 – (a) Mapping Swift/T schema to ProvONE; (b) <i>Mapper</i> of the <i>APP_EXEC</i> entity to <i>Execution</i> entity.	34
Figure 8 – Polyflow architecture.	36
Figure 9 – Polyflow relational schema	37
Figure 10 – (a) Entity Mapper for the Execution ProvONE entity in Swift/T’s database; (b) Expansion for the query over all ProvONE Executions in Swift/T’s database.	38
Figure 11 – Querying the Execution ProvONE entity in Swift/T’s database.	38
Figure 12 – (a) Entity Mapper for the Program ProvONE entity in Kepler’s database; (b) Expansion for the query over all ProvONE Programs in Kepler’s database.	39
Figure 13 – Illustration of a phylogenetic tree. Credit: < https://ksuweb.kennesaw.edu/\protect\unhbox\voidb@x\penalty\@M\{\}jdirnber/Bio2108/Lecture/LecPhylogeny/LecPhylogeny.html >	41
Figure 14 – Specification of the phylogenetic analysis experiment (represented using Apache Taverna notation)	42
Figure 15 – Query classes for two provenance graphs. Inspired by (OLIVEIRA et al., 2016))	43
Figure 16 – Illustration of subject issuing <i>Q1</i> to Polyflow	47
Figure 17 – Illustration of participant issuing <i>Q6</i> to Polyflow	49

ACRONYMS

API	Application Programming Interface
CCM	Conceptual Canonical Model
CGS	Conceptual Global Schema
DBMS	Database Management System
DNA	Deoxyribonucleic Acid
EC	Exclusion Criteria
ETL	Extraction-Transformation-Loading
IC	Inclusion Criteria
JSON	JavaScript Object Notation
LSs	Local Schemas
MQ_n	Mapping Question n
OPM	Open Provenance Model
PC3	Third Provenance Challenge
p-prov	Prospective Provenance
QLP	Query Language for Provenance
r-prov	Retrospective Provenance
Rabicó	Rede Avançada em Biologia Computacional
RDBMS	Relational Database Management System
RDF	Resource Description Framework
RNA	Ribonucleic Acid
RPQ	Regular Path Queries
SLM	Systematic Literature Mapping
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Standard Query Language
SWPDM	Scientific Workflow Provenance Data Model

UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
Wf	Workflow
WfMS	Workflow Management System
XML	Extensible Markup Language

CONTENTS

1	INTRODUCTION	12
2	BACKGROUND	15
2.1	WORKFLOWS AND PROVENANCE	15
2.2	POLYSTORE SYSTEMS	16
2.3	FINAL REMARKS	20
3	RELATED WORK	21
3.1	SYSTEMATIC LITERATURE MAPPING	21
3.1.1	Planning	21
3.1.2	Execution	24
3.1.3	Results	25
3.2	<i>AD-HOC</i> COMPLEMENTARY LITERATURE REVIEW	28
3.3	DISCUSSION	29
3.4	THREATS TO VALIDITY	32
3.5	FINAL REMARKS	32
4	POLYFLOW: INTEGRATING HETEROGENEOUS PROVENANCE	
	GRAPHS	33
4.1	FORMALISM	33
4.2	MAPPING LOCAL SCHEMAS TO PROVONE	33
4.3	ARCHITECTURE OVERVIEW	34
4.4	POLYFLOW IN ACTION	36
4.5	FINAL REMARKS	39
5	EVALUATION AND RESULTS	40
5.1	EVALUATION CONTEXT	40
5.2	FEASIBILITY STUDY	41
5.3	USABILITY EVALUATION	45
5.3.1	Domain Expert Execution and Results	47
5.3.2	Domain Knowledge Execution and Results	48
5.3.3	Discussion	50
5.4	FINAL REMARKS	50
6	FINAL REMARKS	52
7	APPENDIX	54

7.1	APPENDIX A - PAPERS REPORTED IN SLM	54
7.2	APPENDIX B - QUESTIONNAIRE USED USABILITY AS- SESSMENT	56
	REFERENCES	59

1 INTRODUCTION

Over the last decade, the (big) data-driven science paradigm became a reality (HEY et al., 2009; OLIVEIRA; LIU; PACITTI, 2019). As discussed by Abbasi, Sarker and Chiang (2016) and Jagadish et al. (2014), there has been an increasing number of efforts by the industry to create more efficient and accurate applications and information systems (*e.g.*, new view on Business Intelligence and Analytics) to adhere to this new paradigm (ABADI et al., 2016; ABADI et al., 2019). However, some challenges must still be overcome, such as the dependency on data quality (HAZEN et al., 2014) and the lack of reproducibility of the results (PENG, 2015; SCHWAB; KARRENBACH; CLAERBOUT, 2000; FREIRE; CHIRIGATI, 2018; CHIRIGATI; FREIRE, 2018). In order to foster reproducibility (OLIVEIRA; OLIVEIRA; MATTOSO, 2017), historical information such as all generated data, the used software and the settings of the execution environment must be made available to different researchers. This metadata is called *Provenance* (FREIRE et al., 2008). Provenance can be classified as *Prospective* (or simply *p-prov*), which is associated to the specification of an experiment, and *Retrospective* (or simply *r-prov*), which is associated to the execution of an experiment. Since provenance represents generated data and the processes that transformed it, provenance can be represented in a graph, called a *provenance graph*, whose nodes represent the artifacts/influences and whose edges their relations with one another (HUYNH et al., 2018).

The lack of provenance data can be especially hindering for researchers that use computational models to conduct experiments (MATTOSO et al., 2010). The use of computational simulations to support experiments in various fields of science has become a reality in the last 10 years (MATTOSO et al., 2010; ATKINSON et al., 2017; DEELMAN et al., 2018). These experiments follow a well-defined life cycle (with composition, execution, and analysis steps (MATTOSO et al., 2010)), and they are commonly composed by the invocation of several applications in a specific order, according to their production and consumption of data, thus creating a *Scientific Workflow* (henceforth named as *Workflow*) (ATKINSON et al., 2017). There are several information systems that already support the composition and execution of such workflows, *e.g.*, Workflow Management Systems (WfMS) and Science Gateways. There is a plethora of WfMSs, such as Kepler (ALTINTAS et al., 2004), Taverna (WOLSTENCROFT et al., 2013), Chiron (OGASAWARA et al., 2013), Swift/T (WOZNIAK et al., 2013), Pegasus (DEELMAN et al., 2015), eScience Central (WATSON; HIDEN; WOODMAN, 2010), VisTrails (BAVOIL et al., 2005), and SciCumulus (OLIVEIRA et al., 2010; OLIVEIRA et al., 2013). Each of these WfMS manage the execution of the workflow and capture provenance data from the workflow execution automatically.

Science Gateways (GESING et al., 2018) are complex information systems that aim at integrating several existing approaches that support the composition and execution of

workflows in distributed environments, such as clouds, grids, and clusters, by integrating many existing WfMSs (GLATARD et al., 2017). Although such gateways represent a step forward, they follow a tight integration, where the underlying WfMSs share software components with the gateway, but not their provenance databases and repositories. In this way, it is extremely difficult to query all provenance databases in an integrated form since most solutions adopted proprietary data models. Thus, interoperating provenance data captured by underlying WfMSs remains an issue (OLIVEIRA et al., 2016).

In theory, it should be possible for scientists to query both provenance databases transparently. However, the heterogeneity in the data models and implementations makes it difficult to query them in an integrated way, requiring researchers to be aware of both database schemas and their associations. Ultimately, this lessens the role of provenance in fostering interoperability. This way, one can reduce this scenario to two fundamental issues: (i) the heterogeneity of storage methods and (ii) the heterogeneity of data models. These issues lead to two types of interoperability issues in provenance databases: (i) syntactic and (ii) semantic (LITWIN; ABDELLATIF, 1986).

Recently, new approaches such as Polystore systems (BEGOLI; KISTLER; BATES, 2016; HAMADOU; GALLINUCCI; GOLFARELLI, 2019; KHAN et al., 2019b) started being discussed very intensively across the research community as a solution for integrating multiple heterogeneous databases. According to (DUGGAN et al., 2015), a Polystore system is built on top of multiple, heterogeneous, integrated storage engines. Differently from their predecessors *Federated Databases*, they integrate several heterogeneous databases engines while accessing them separately through their own query engine (DUGGAN et al., 2015). Polystore databases support multiple query languages and data models, as opposed to traditional federated systems that support a single one. The rationale behind Polystore systems suits well for the problem of querying heterogeneous provenance databases.

Nevertheless, the semantic interoperability problem is still an open issue since one needs to be aware of the underlying provenance database schemas. This way, in this dissertation, we propose **Polyflow**, a Polystore-compliant mechanism that provides *semantic interoperability* for heterogeneous provenance graphs. By assuming that all data models are associated with the same domain (*i.e.*, provenance), we can leverage a Conceptual Canonical Model (CCM) (*e.g.*, ProvONE (CUEVAS-VICENTTÍN et al., 2015)) that can represent all concepts involved, even if they present different granularity. **Polyflow** uses a *mediation approach*, *i.e.*, when connected to a *Data Source* (*e.g.*, a provenance database), users can create *Entity Mappers* between elements in the CCM and elements in the original data models (COULOURIS; DOLLIMORE; KINDBERG, 2005) that are used to rewrite, at runtime, the submitted queries. Thus, queries submitted to **Polyflow** refer to ProvONE entities and then are rewritten to be submitted to the underlying provenance databases with their respective local schemas. **Polyflow** supports distributed

environments, empowering geographically scattered researchers, since different researchers commonly use different WfMSs (and consequently different provenance databases).

The contributions of this dissertation are summarized as follows:

- An overview of the research topic using a Systematic Literature Mapping (SLM). The objective of the SLM is comparing state-of-the-art approaches, identifying the current limitations of these solutions.
- With a better an understanding the research landscape, we propose `Polyflow`, an approach to handle the syntactic and semantic interoperability in provenance databases using polystore paradigm.
- A series of experiments to evaluate `Polyflow` with domain experts.

The remainder of this dissertation is organized as follows. Chapter 2 briefly provides the background knowledge that supports this work. Chapter 3 showcases related work, granting an overview of the research topic. Chapter 4 details the proposed approach. Chapter 5 showcases two evaluations of `Polyflow`. Finally, Chapter 6 concludes the dissertation.

2 BACKGROUND

In this chapter, we introduce the main concepts and techniques adopted in this dissertation. Firstly, we formalize scientific workflows and provenance concepts. Next, we discuss Polystore systems and databases.

2.1 WORKFLOWS AND PROVENANCE

Over the last few years, workflows have become a *de facto* standard to represent scientific experiments based on computational simulations (OLIVEIRA; LIU; PACITTI, 2019). A workflow is an abstraction capable of representing a logical sequence of programs and/or services invocations (*i.e.*, *activities*) and their data dependencies (MATTOSSO et al., 2010). Thus, a workflow can be formally defined as an “automation of scientific processes in which tasks are structured based on their control and data dependencies” (YU; BUYYA, 2005). In other words, it can be seen as the formalization of a pipeline of computational tasks with their respective inputs and outputs. According to Oliveira et al. (2012), a workflow can be formalized as follows: a graph $W(A, \phi)$, where A is the set of activities in W and ϕ is the set of data dependencies. Thus, $A = \{a_1, a_2, \dots, a_n\}$ and each activity a_i can be represented as $a_i(I, P)$, $a_i : \{I, P\} \rightarrow O$, where I is the input dataset, P the parameters and O the data generated by activity a_i . Hence, $I = \{i_1, i_2, \dots, i_d\}$, where i_d is an input file for activity a_i , $O = \{o_1, o_2, \dots, o_k\}$, where each o_k is an output file for a_i and $P = \{p_1, p_2, \dots, p_m\}$, where each p_m is a parameter of activity a_i . Each execution of a_i is associated to a tuple of m parameters $\langle p_1, p_2, \dots, p_m \rangle$, where the value v_m of each parameter p_m is defined by a function $\zeta_m(p_m) = v_m$. Thereafter, we can describe a data dependency set as $\phi = \{\varphi_{1,2}, \dots, \varphi_{i,j}\}$, where each $\varphi_{i,j} = \langle i_d, a_i, a_j \rangle$, $input(a_i) \in I$, $i_d \neq \emptyset$ and $output(a_i) \in O$. Thus, $\varphi_{i,j} \leftrightarrow \exists o_k \in input(a_j) | O_k \in output(a_i)$.

In order to evaluate and reproduce workflows, provenance data must be gathered, stored and analyzed. *Provenance* or data lineage is a metadata associated with a data product that describes its derivation path. More formally, it describes a data fragment, and all processes and transformations applied to it (BUNEMAN; KHANNA; WANG-CHIEW, 2001). These metadata bring transparency to a data product, enabling its reuse (SIMMHAN; PLALE; GANNON, 2005). Moreover, it also helps data interpretability and audition (GROTH; MOREAU, 2009). There are two types of provenance: prospective (*p-prov*) and retrospective (*r-prov*) (DAVIDSON; FREIRE, 2008). P-prov aims at capturing the specification of a computational task (*e.g.*, a workflow), expressing the steps that need to be followed to generate a set of data products. On the other hand, r-prov captures the steps executed and information about the execution derives a data product. Although evolution provenance is not formally defined in the literature, it is present in several works (PRABHUNE et al., 2018a; PIMENTEL et al., 2016; PRABHUNE et al.,

2018). It aims to capture the evolution of the workflow definition throughout its versions (CUEVAS-VICENTTÍN et al., 2015).

In the context of this dissertation, we use ProvONE (CUEVAS-VICENTTÍN et al., 2015) as a CCM capable of integrating provenance graphs of multiple workflows and produced by different WfMSs. ProvONE extends the W3C PROV recommendation (MOREAU et al., 2015) and is composed of several classes and relationships. The reasons why we chose ProvONE are two-fold: (i) it is currently being evaluated as a W3C, to become a recommendation for workflow provenance representation possibly; (ii) it has been widely adopted by related literature, as we will show in Chapter 3. You can refer for more information in (PRABHUNE et al., 2016).

ProvONE’s data model is represented in Fig. 1. In ProvONE, a p-prov graph is described by the following entities: *Programs* represents an activity that consumes and generates data through its *Ports*. *Program* instances can be atomic or composite, *i.e.*, they can have *SubPrograms*. A *Port* enables a *Program* to send and receive data and parameters. Data dependencies across programs are identified by the class *Channels* that connect two or more *Programs* through their *Ports*. The class *Workflow* represents a special kind of *Program* that is the root of a recursive composition of *Programs*. Finally, the class *Entity* represents the information units consumed or generated by a *Program*.

On the other hand, a r-prov graph G_p is generated by the *Execution* of W . It is defined as $G_p = (V_p, E_p, A_p, T_p)$, where each vertex V_p represents either programs or data artifacts and the edges E_p represent its lineage. An attribute $type \in A_p$ should exist for every vertex and edge. If a node v_{pi} represents a program associated with an activity a_i , then $Value(v_{pi}, type) = program$. On the other hand, if it represents a data artifact, $Value(v_{pi}, type) = data$. The set T_p represents edge types that can be: *WasGeneratedBy*, *Used*, *WasInformedBy*, *WasDerivedFrom*, *WasAttributedTo*, *WasAssociatedWith*, *ActedOnBehalfOf*, according to the PROV recommendation (MOREAU et al., 2015) and ProvONE data model (Fig 1). The *Users* entity and its relations, identify workflow’s invocations. For a complete understanding of the ProvONE, refer to their complete specification¹

2.2 POLYSTORE SYSTEMS

According to Wegner (1996), the term “Interoperability” can be defined as “the ability of two or more software components to cooperate despite differences in language, interface, and execution platform”. Although this can be considered an outdated definition for interoperability (TOLK; MUGUIRA, 2003), it represents what is known as *syntactic interoperability*. However, syntactic interoperability is just one existing type

¹ <<http://jenkins-1.dataone.org/jenkins/view/Documentation%20Projects/job/ProvONE-Documentation-trunk/ws/provenance/ProvONE/v1/provone.html>>

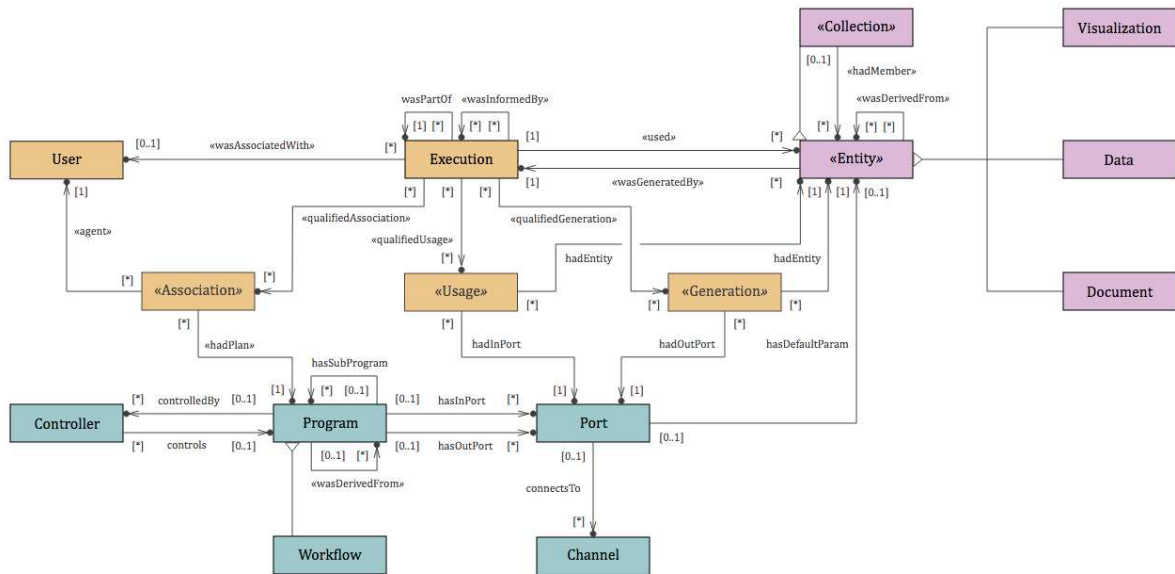


Figure 1 – ProvONE model adapted from (VICTOR LUDÄSCHER BERTRAM, 2016)

of interoperability. In this dissertation’s context, the main challenge is how to provide semantic interoperability of provenance databases. We assume that software components are the heterogeneous sources (*e.g.*, provenance databases managed by existing Database Management Systems (DBMSs)) that provide access to provenance data generated by WfMSs. As hinted in Chapter 1, the approach proposed in this dissertation is built on top of a *Polystore* system. Polystore systems can be seen as a new type of data federation, *i.e.*, a meta-database management system that provides a centralized and transparent interface to underlying database engines (GADEPALLY et al., 2016).

Differently from their predecessors (Federated Systems) that support a single query language and data model, Polystore systems support multiple query languages and data models. Polystore systems aim at mitigating usability issues by providing users a wide array of storage solutions and query languages. This new paradigm offers an alternative to the traditional “*one size fits all*” approach (STONEBRAKER, 2015), storing and processing fragments of the dataset in the engine that provides the best performance to operation at hand (*e.g.*, insertion, queries) (GADEPALLY et al., 2016). BigDAWG² (GADEPALLY et al., 2016) is a pioneer Polystore system and is the engine responsible for the syntactic interoperability layer of the proposed solution. BigDAWG’s architecture is presented in Fig. 2.

The *middleware* layer is responsible for orchestrating incoming user queries, plan its submission to the underlying islands, and integrating intermediate responses. In other words, the *middleware* is responsible for identifying which islands a query target, break it down into island-specific queries, and then integrating partial responses that are returned

² <<https://bigdawg.mit.edu>>

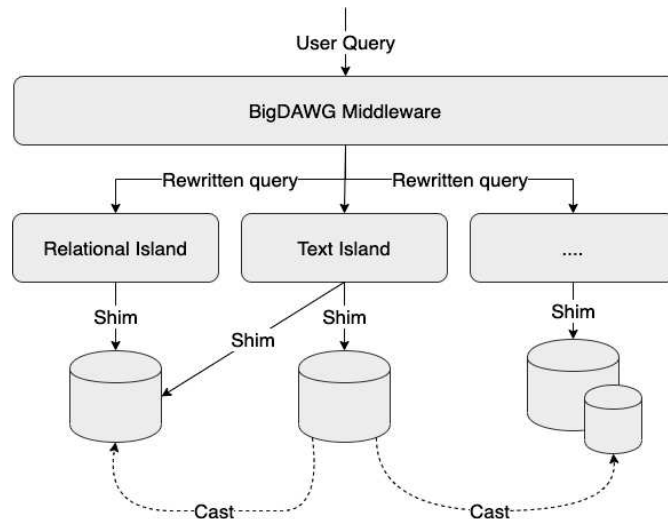


Figure 2 – BigDAWG’s architectural design. Inspired in (DUGGAN et al., 2015)

by each DBMS. The *middleware* also maintains a record of previous query performance for optimization’s purpose.

An *island* is the definition of a *data model* and a *query language* that represents a *data type*. For instance, you would use SQL to submit queries to the *relational island* because it is the *query language* defined by it. Since the *data model* or *query language* defined by an island might be different from the one implemented in an underlying DBMS, Gadepally et al. (2016) introduces the concept of *shims*.

The *shim* operator is responsible for translating the data model and query constructs defined by an island to the model and constructs supported by the underlying DBMS. Furthermore, shims may navigate across different islands, *i.e.*, users may recover data using query constructs from different islands than the database that belongs.

Finally, the *cast* operator is responsible for migrating data between engines. It is very useful for boosting the performance of queries. For example, suppose the users would like to analyze data stored in a relational database that is very computationally expensive to conduct on an RDBMS. In that case, they can move it to a different engine that better supports the operation. You can picture it as an Extraction-Transformation-Loading (ETL) process across database engines.

We chose to adopt BigDAWG as the Polystore layer of this work because, to the best of the authors’ knowledge, at the time in which this work began to unfold, it was the only existing Polystore implementation that was open source and had a license that allowed us to use it for free. It is important to highlight that `Polyflow` does not follow a tight integration, and the underlying polystore system can be easily changed by creating a new querying interface.

As discussed by Tolk and Muguira (2003), interoperability goes beyond the imple-

mentation itself. The last layer of conceptual interoperability proposed by the authors is named “Harmonized data”, where “applications can comprehend the data, both structurally and semantically”. In this dissertation, semantic interoperability is the ability to bridge granularity and representation differences in provenance data captured and stored by WfMSs with different data models.. It comes down to the database integration problem with a bottom-up approach (COULOURIS; DOLLIMORE; KINDBERG, 2005), *i.e.*, seamlessly integrating several different databases into one.

The first step of the process proposed in Coulouris, Dollimore and Kindberg (2005) is to define *conceptual schemas*. *Local Schemas* (LSs) are schemas that describe data from a specific database. Since we aim to integrate several heterogeneous data sources into a single one, a *Conceptual Global Schema* (CGS) is required. The second step is to define the integration strategy to allow data to flow between the schemas. It can be either *physically* or *logically* (JHINGRAN; MATTOS; PIRAHESH, 2002).

The *physical* approach (Fig. 3(a)) materializes the results of the mapping using the CGS, which speeds up querying. However, this approach has an *availability* problem, since it requires constant extraction of data from underlying databases to keep data up to date. On the other hand, the *logical* approach (Fig. 3(b)) transforms, at runtime, queries using entities and relations of CGS to constructs of underlying databases. This approach does not have any *availability* issues since the original data is being queried. However, speed may become an issue for two reasons: (i) the overhead added by the translation operation; and (ii) queries may not be as optimal because there is a layer that is abstracted from the end-user.

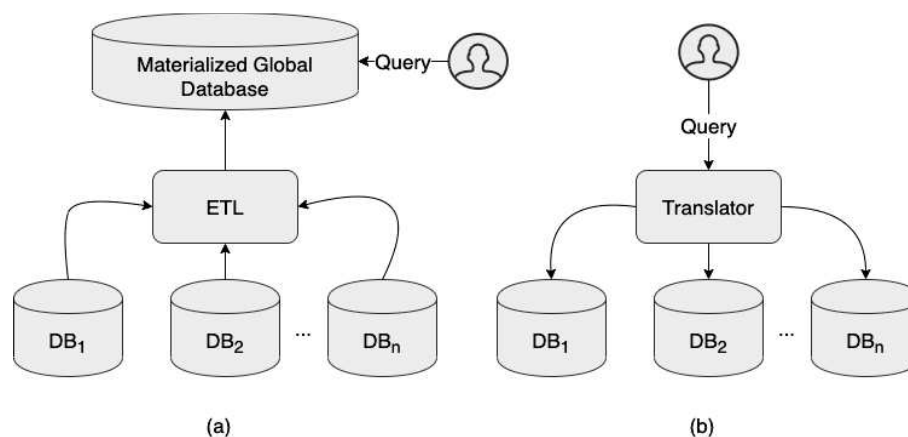


Figure 3 – (a) Physical approach to database integration. (b) Logical approach to database integration. Both figures adapted from (COULOURIS; DOLLIMORE; KINDBERG, 2005)

2.3 FINAL REMARKS

In this chapter, the background knowledge required to understand this work was laid out. Firstly, the domain in which this work was built around, *workflows and provenance*, are introduced. Then, we go over *syntactic and semantic interoperability*, the two research problems we aim to provide support for. We leverage the usage of *Polystore systems*, a novel approach to federated systems, to provide support to syntactic interoperability. In section 2.2, we explain its architecture and fundamentals. Finally, we highlight the two approaches to provide support to *semantic interoperability*, namely *physical* and *logic*, explaining their trade-offs and justifying our choice to go for a *logic* integration approach.

3 RELATED WORK

In this Chapter, we report a Systematic Literature Mapping (SLM) conducted in August, 2018 and an *ad hoc* complementary study to discuss works published between August, 2018 and October, 2020.

3.1 SYSTEMATIC LITERATURE MAPPING

The goal of the SLM is to identify and discuss interoperability solutions for heterogeneous provenance data, qualitatively assessing them by the following perspectives: *(i) Completeness*: the ability to capture p-prov, r-prov and evolutionary provenance, *(ii) User Adaptability*: how big is the learning curve for new users and *(iii) Extensibility*: how hard it is to extend the proposed solution. Thus, the contributions are twofold: (i) it provides an overview of the research area, identifying the most used provenance models and query languages; (ii) each work is individually presented and discussed, providing qualitative analysis of these solutions based on aforementioned metrics, guiding future research efforts.

As discussed by Pérez, Rubio and Sáenz-Adán (2018), there are numerous surveys regarding provenance and WfMSs (*e.g.*, (SIMMHAN; PLALE; GANNON, 2005; DAVIDSON; FREIRE, 2008; BOSE; FREW, 2005)). However, to the best of the authors' knowledge, there are no secondary studies or surveys that tackle interoperability of provenance data generated by WfMSs, thus justifying this study. This SLM is structured based on the guidelines on established by Budgen et al. (2008) and Kitchenham (2004). Section 3.1.1 discusses the planning of this SLM; Section 3.1.2 presents the execution procedure; Section 3.1.3 reports the findings; showcasing and discussing related literature. Finally, Section 3.4 exposes threats to this study's validity.

3.1.1 Planning

We identified the goals during the planning process and defined a protocol, following the guidelines described in Kitchenham (2004). The protocol specifies the method to be used in the SLM in order to reduce researcher bias (STEINMACHER; CHAVES; GEROSA, 2013). Moreover, an SLM must be reproducible and the protocol is the document that empowers it. The main goal of this study is the **identification of the current state-of-the-art in provenance data interoperability**. As a secondary goal, we aim at evaluating these solutions under three different aspects: completeness, usability, and extensibility, identifying possible improvements to those. We hereby define *provenance data interoperability* as any effort that allows users to store and query heterogeneous provenance traces of workflows, *i.e.*, provenance data described by different formats and/or generated by different WfMSs. Finally, we restrict this dissertation's scope to assess only

papers that propose a solution that supports interoperability to provenance data generated by WfMSs.

Regarding the *Protocol definition*, the first step is to define the *Population*, *Intervention*, *Comparison*, *Outcome* and *Context*, according to the PICOC procedure (KITCHENHAM, 2004). This is done to formalize the scope of the study as following: (i) Population: Heterogeneous provenance data, (ii) Intervention: Support interoperability, (iii) Comparison: -, (iv) Outcome: Solutions (frameworks, tools, models, architectures, etc), and (v) Context: Scientific experimentation.

Since the terms used in the PICOC may have multiple synonyms, we have to define keywords and synonyms (Table 1) that will compose the search string. The second step is to define Mapping Questions (MQ) that the SLM looks for answers to, namely: (MQ1) What is the current state of the art of heterogeneous provenance data interoperability? (MQ2) What query languages are the most used in solutions that support interoperability across heterogeneous provenance graphs? (MQ3) What models are the most used to represent provenance data?, and (MQ4) What are the existing gaps that justify the improvement of the current state of the art?

Table 1 – Keywords and synonyms derived from PICOC.

Keyword	Synonyms	Related to
Heterogeneous provenance data	Heterogeneous lineage data Heterogeneous pedigree data Heterogeneous provenance graphs Heterogeneous tracking data Provenance graphs	Population
Interoperability	Interoperable Interoperate	Intervention
Scientific experimentation	e-Science Scientific Workflows	Context
Collaborative experiments	Collaborative research Collaborative workflows	Context

In order to evaluate the latter more objectively, three aspects of the solution are considered: completeness, usability, and extensibility. To assess completeness, we analyze its ability to capture p-prov, r-prov, and evolution provenance. We also evaluate the querying interface, more specifically, if it supports query languages that their prospective users are already familiar with. Finally, to assess extensibility, we consider the used model - if it is already defined in the literature or if it is a novel approach. All aspects mentioned above are evaluated using the following annotation:

(i) \equiv - The solution fully satisfies the assessed metric,

(ii) \simeq - The solution partly satisfies the assessed metric, and

(iii)□ - The solution does not satisfy the assessed metric.

Due to the vast number of papers returned during the search process, we had to define a *Filtering Process* to guarantee the reproducibility of the SLM. The first filtering was performed by reading the title of the paper. At this stage, only the papers in which the title clearly indicated that the paper did not propose a solution towards supporting interoperability between multiple scientific workflows (*e.g.* A scripting approach for integrating software packages and geoprocessing services into scientific workflows) or that indicated that the paper was not a primary study (*e.g.* A systematic review of provenance systems) were removed. Since a large volume of papers were evaluated in this first phase, to minimize human bias and error, this process was revisited on two distinct occasions.

The second filtering was performed by assessing the papers' title, abstract, introduction, and conclusion when they were available. Finally, the last filtering was performed by reading the entire paper. To mitigate threats to this study regarding papers that share the same scope as this SLM but were either not in any of the knowledge bases used in this research, or not captured by the search string, we conducted one iteration of backward and forward snowballing (WOHLIN, 2014). We chose the following *Knowledge Bases* according to criteria proposed by Costa and Murta (2013): (i) They are capable of using logical expressions or a similar mechanism; (ii) They allow full-length searches or searches only in specific fields of the works; (iii) They are available in the researcher's institution; and (iv) They cover the research area of interest in this mapping: computer science.

This way, the search was done using the following knowledge bases: (i) ACM Digital Library¹, (ii) El Compendex², (iii) IEEE Digital Library³, (iv) Scopus⁴, and (v) Springer Link⁵. Even though Springer does not provide a refined search (*e.g.*, search indexed metadata terms) as other knowledge bases, it is important to the present context since many papers in the area, *e.g.*, (OLIVEIRA et al., 2016; PRABHUNE et al., 2016; PRABHUNE et al., 2018b), are only available in this knowledge base.

To create the *Search String*, we used the terms defined in PICOC. Iteratively, the knowledge bases were queried, and some papers were read, enriching the Keywords and synonyms (Table 1). This process culminated in the following search string, validated by co-authors and researchers in this domain: (“*Collaborative experiments*” OR “*Collaborative research*” OR “*Collaborative workflows*” OR “*Heterogeneous provenance data*” OR “*heterogeneous lineage data*” OR “*heterogeneous pedigree data*” OR “*heterogeneous provenance graphs*” OR “*heterogeneous tracking data*” OR “*provenance graphs*” OR “*scientific workflow*” OR “*e-science*”) AND (“*interoperability*” OR “*interoperable*” OR “*interoperate*”)

¹ <<http://portal.acm.org>>

² <<http://www.engineeringvillage.com>>

³ <<http://ieeexplore.ieee.org>>

⁴ <<http://www.scopus.com>>

⁵ <<http://link.springer.com>>

The decision on whether or not to discard a paper is based on the following *Inclusion Criteria (IC)* and *Exclusion Criteria (EC)*: (IC1) The work proposes an interoperability solution on storing and querying heterogeneous provenance metadata; (EC1) Deprecated, *i.e.*, a more recent follow-up study was found; (EC2) Duplicated, *i.e.*, the work was already recovered in another knowledge base; (EC3) The paper does not have an abstract; (EC4) The paper is not a primary study; (EC5) The paper is not available to download using the university’s credentials; (EC6) The study was published as a short paper; (EC7) The study is not written in English; (EC8) The study was not published in a conference or journal related to Computer Science; (EC9) The study was not published in a peer-review vehicle; (EC10) The study was published before 2008; (EC11) The study does not propose a solution that supports interoperability across heterogeneous provenance datasets; and (EC12) The proposed solution is not able to capture generic provenance metadata - *i.e.*, it was designed for a domain-specific solution.

Exclusion criteria from 1 through 9 are self-explanatory and exist to guarantee the quality of the papers assessed in this SLM. EC10 was included because 2008 is when the first standard provenance recommendation - Open Provenance Model (OPM) (MOREAU et al., 2008) - was published. EC11 and EC12 were taken into account to filter papers that did not attend this SLM scope but were returned by the knowledge bases’ query engines. It is worth mentioning that some of these criteria (EC7, EC8, EC10) were applied during the search process in the knowledge bases.

3.1.2 Execution

We executed the search string in the knowledge bases previously presented. The results (available as BibTeX entries at <<https://bit.ly/37ROdvv>>) gathered from all knowledge bases were organized in Parsifal⁶. Table 2 synthesizes the results of each filtering strategy based on ICs and ECs previously discussed. We then conducted one iteration of backward and forward snowballing on this set of papers, following the guidelines established by Wohlin (2014), applying the same inclusion and exclusion criteria defined before.

Knowledge Base	# papers returned	After duplicates removal	After reading title	After reading abstract, introduction and conclusion	After reading full paper
ACM	95	75	8	0	0
Compendex	144	82	31	5	4
IEEE	81	48	12	5	3
Scopus	168	108	26	2	1
Springer	747	727	58	7	3
Total	1235	1040	135	19	11

Table 2 – SLM filtering processes and the results of each step.

⁶ <<http://parsif.al>>

3.1.3 Results

In this Section, we briefly discuss the proposed solutions and assess them under the qualitative attributes defined in Section 3.1.1. Finally, we compare the works and synthesize their evaluation regarding this SLM’s questions.

Ellqvist et al. (2009) propose a mediator-based architecture that is able to interoperate provenance data derived from different data sources. The architecture has two main components: a global schema that is general and able to represent provenance information expressed by other models and a system-independent query API that is able to retrieve answers from distinct data sources. The authors propose a data model, the Scientific Workflow Provenance Data Model (SWPDM), to represent their global schema, since the reference provenance data model available at the time (OPM (MOREAU et al., 2008)) is unable to capture p-prov. However, it is unable to capture evolution provenance. The querying layer is implemented as independent APIs for each implemented wrapper. To evaluate their proposal, the authors conducted a case study with three different WfMSs, creating wrappers that transform data generated by them to SWDPM, querying the integrated knowledge base in a illustrative fashion.

Chebotko et al. (2010) propose an integration approach that takes advantage of provenance models described by ontologies. They designed a storing solution, RDFProv, that transparently works as an RDF store, even though they use a relational storage. The authors used domain-specific constraints, *e.g.*, low frequency on update and deletion operations, to boost performance. The proposed architecture is organized in three layers: firstly, the provenance model layer is responsible for managing provenance ontologies and execute inferences rules on the knowledge-base to generate new triples, enriching the dataset. The mapping layer provides three functionalities: (i) Schema Mapping: responsible for generating a relational schema based on the ontology; (ii) Data Mapping: responsible for mapping RDF triples to relational tuples and (iii) Query Mapping: responsible for translating SPARQL queries to SQL. Finally, the relational model layer is responsible for storing the data in a relational fashion. It requires domain-specialists to interoperate datasets represented in different models. Moreover, the solution is restricted to models that have an ontology representation, which several WfMSs do not. RDFProv is model-dependent, hence, the ability to capture evolution, p-prov and r-prov provenance cannot be assessed in a general manner. Users are restricted to SPARQL queries when using RDFProv.

Missier et al. (2010), along with integrating heterogeneous provenance data, also propose a solution to keep the provenance trace alive between workflows executions, *i.e.*, identify that a resource generated by a given workflow execution is the same as consumed by another. For the integration solution, the authors propose an extension of OPM (MOREAU et al., 2008), since it is unable to capture p-prov in its original

form, as discussed previously. The proposed model, however, lacks support to evolution provenance. As a proof of concept, the authors implement mapping algorithms that transforms provenance data from two different WfMSs, namely Taverna and Kepler, into the proposed model. (MISSIER et al., 2010) use a relational format to store data and the solution only supports SQL constructs. However, it can be a challenge to users of WfMSs that do not support this querying format. For the tracing problem, the authors define the $copy(r, S, S')$ operation where: r is a reference to the resource - *e.g.* an Uniform Resource Identifier (URI) -; S is the origin storage (*i.e.*, where r is being copied from); and S' is the destination storage (*i.e.*, where r is being copied to), thus guaranteeing the provenance trace connectivity between workflows.

Anand et al. (2010) propose a tool that supports users in the access and exploration of provenance metadata by providing a browsing and querying interface. To achieve that, they use a novel provenance model that is able to express provenance traces, however, it is unable to describe p-prov and capture the workflow's evolution. They show that the model has a correspondence to the OPM and that the proposed architecture can accommodate the latter. Finally, they introduce the Query Language for Provenance (QLP), a query language based on path expressions, having a similar syntax to graph-based languages. Even though in their implementation the authors use a relational database, conceptually, the proposed architecture can support different storage solutions by interfacing them to the proposed query language, namely, QLP.

In (ALTINTAS et al., 2010), an extension of the QLP (ANAND et al., 2009) is proposed to capture implicit user collaborations, *i.e.*, relations between agents that, directly or indirectly, influenced a data product generation. Moreover, the authors also establish a mapping between QLP and OPM (MOREAU et al., 2008), enabling their solution to be utilized on any data described by the latter. The proposed architecture consists of heterogeneous data stores (*e.g.*, Relational Database Management Systems (RDBMS), XML and RDF files), a query engine that is responsible for translating QLP queries to the respective storage query language and a query interface that works as the user's endpoint. From a usability stand point, this is an advancement when compared to the previous analyzed works, since the solution exempts the user's need to understand the storage solution, providing a single query interface that is able retrieve information stored in different formats.

In (LIM et al., 2011), the authors formally define a relational OPM-compliant (MOREAU et al., 2008) model, named OPMPProv. To evaluate it, the authors use the relational model to answer a set of queries defined in the Third Provenance Challenge (PC3) (THE. . . , 2009), an effort to promote studies regarding WfMSs interoperability. (GASPAR; BRAGA; CAMPOS, 2011) propose a generic architecture that can be coupled to WfMSs and is responsible for collecting and managing provenance information generated by the

workflow execution. The authors use OPM as a canonical model to represent data and a relational storage solution to persist it. To foster the semantic approach of the proposal, the authors also provide an SPARQL querying interface through the representation of the provenance graph in RDF format associated with the OPM ontology, enabling the usage of reasoners to make inferences in these knowledge bases. To illustrate their proposal, the authors conduct some case studies to evaluate their architecture suitability to collect and manage provenance metadata.

Ding et al. (2011) assessed shortcomings of solutions proposed at the PC3, eliciting requirements and developing a solution with web-semantic components. It extends the OPM ontology specification and, given a OPM-compliant RDF provenance trace, users can query the knowledge base via SPARQL constructs. The proposed data model lacks p-prov and evolution provenance support. Moreover, their architecture only supports RDF stores and users are restricted to SPARQL constructs. Regarding extensibility, a mapping between the proposed model or OPM ontology is required to support data described by different formats.

In (CUEVAS-VICENTTIN et al., 2012), the authors propose an extension of the OPM (MOREAU et al., 2008), namely D-OPM, that is able to capture p-prov, r-prov and evolution provenance. To evaluate their model, the authors implement it in a RDBMS and provide a querying mechanism based on Regular Path Queries (RPQ): graph paths expressed as regular expressions, to ease querying since provenance trace resembles graphs. Finally, they conduct a performance evaluation of their architecture utilizing a generic graph testbed dataset. Since their proposal provide a standard canonical data representation and storage solution, to integrate data derived from different WfMSs, all one must do is create wrappers to transform the data and store it.

Oliveira et al. (2016) propose an integration architecture that has two layers: the first (namely, Cartridges - a wrapper abstraction -) is responsible for transforming WfMSs' traces into Prolog facts described by the ProvONE model; the second, a shared knowledge base where the facts are stored and can be accessed via Prolog queries. The authors evaluate their approach using real workflow traces generated by two research groups that share the same research domain. From an extensibility standpoint, since all data is represented by the ProvONE model and integrated in the shared knowledge base layer, all one must do is implement a new Cartridge to support a new WfMS. On the other hand, to access the knowledge base, users must write Prolog queries (a query language that is not supported by most WfMSs), which may hamper the usability of the approach.

Jabal and Bertino (2016) propose a data model focused on access control over provenance data that is unable to capture both p-prov and evolution provenance. Moreover, they also implement algorithms that map data described in their model to OPM (MOREAU et al., 2008) and PROV (PROV-OVERVIEW, 2013). The solution supports RDBMS and

a graph store (Neo4j⁷) as storage solutions for their framework.

Differently from Jabal and Bertino (2016), Prabhune et al. (2018) proposes a more generic framework that aims to support any kind of metadata, not only provenance. They accomplish that by providing support to multiple data models and tools to handle metadata from different domains. As for provenance metadata, they support two models, namely ProvONE a PREMIS (LI; SUGIMOTO, 2014), storing data in a RDF solution (Apache Jena TDB⁸). Along with providing a SPARQL endpoint, the authors also implement an API with various query patterns already implemented that retrieve useful provenance information.

Prabhune et al. (2018b) propose a framework that aids researchers in analyzing heterogeneous provenance metadata. To accomplish that, they use a similar approach to (PRABHUNE et al., 2018) to handle provenance: a RDF storage solution where data is represented by the ProvONE model. In this work, they implement three mapping algorithms that transforms data described by other formats into ProvONE-compliant data. To evaluate their framework, the authors illustrate its functionality by interoperating data generated by different WfMSs.

3.2 AD-HOC COMPLEMENTARY LITERATURE REVIEW

Since some time has elapsed since the SLM execution and elaborating this dissertation, we need to complement it with published works since then. Since the SLM process is time-consuming, we have opted to cover works published in the last 2 years with an *ad-hoc* search. We used Google Scholar as the search engine and issued the following query: “*Heterogeneous "provenance data" interoperability*”. From the results, we filtered the papers based on their titles and abstracts. We present the results following.

Parciak et al. (2019) consider different research groups, in the medical domain, sharing research topics that have difficulty assessing concluding results created by heterogeneous software. The authors propose an implementation roadmap of a system that captures provenance, using the PROV data model, so these heterogeneous processes are comparable. Souza et al. (2019) propose a distributed system that can capture heterogeneous workflows provenance data at runtime with small overhead. Even though the authors do not propose a solution that integrates heterogeneous provenance data, they still propose a workflow-agnostic solution that could be plugged into any WfMS. They an extension of PROV as their *lingua franca* to capture provenance from the external workflows. Khan et al. (2019a) collect a comprehensive summary of recommendations by the community regarding workflow design and resource sharing and leverage that knowledge to define a hierarchical provenance framework. The goal is to achieve homogeneity in the granularity

⁷ <<https://neo4j.com/>>

⁸ <https://jena.apache.org/documentation/tdb>

of the information shared, with each level addressing specific provenance recommendations. Based on that, they also define a standardized format, namely CWLProv.

3.3 DISCUSSION

All the papers evaluated in this SLM propose either a storage solution and/or an architecture that supports interoperability of heterogeneous provenance graphs. A timeline of the publications is presented in Fig. 4. Fig. 5 illustrates vehicle types in which the papers have been published and Fig. 6 shows the number of citations of each paper. Following we discuss each of the questions.

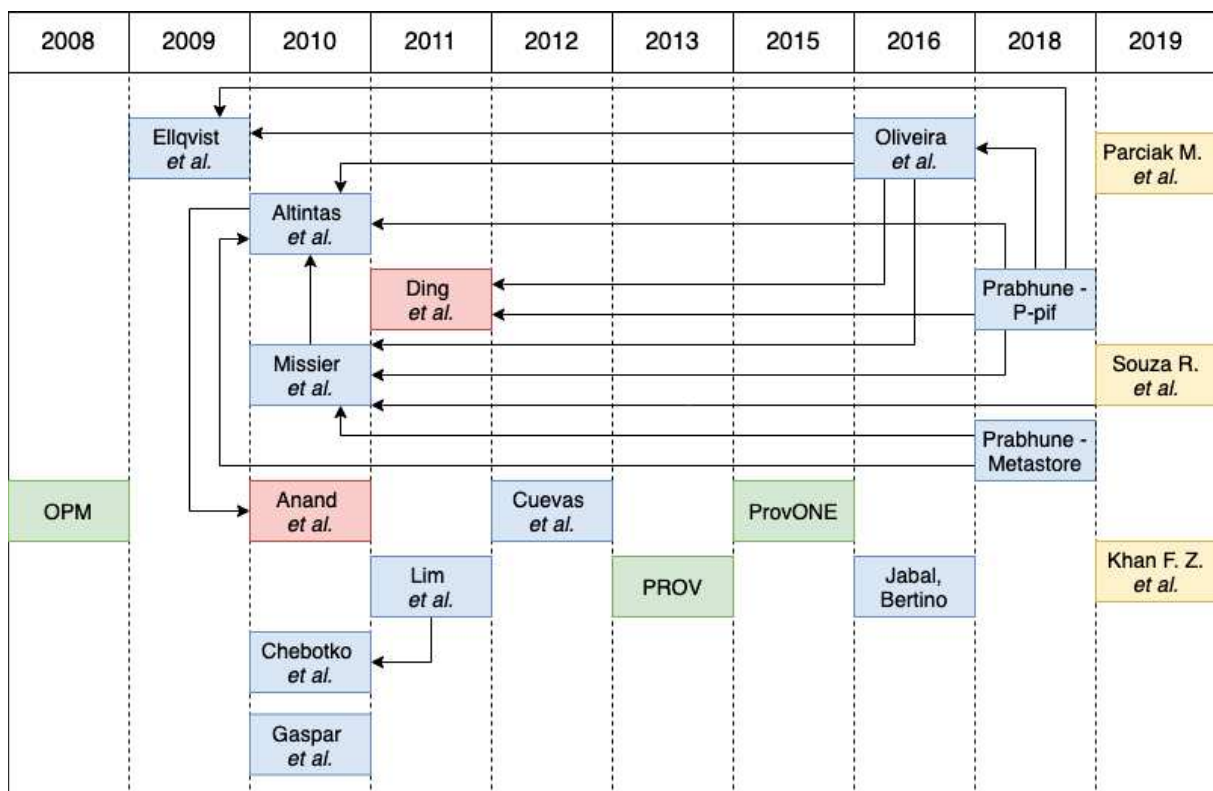


Figure 4 – Timeline of publications. Edges connecting two papers represent a citation. Works in red are the product of snowballing; the ones in blue were found in the knowledge bases and the yellow ones were derived from the ad-hoc complimentary search.

(MQ1) *What is the current state of the art of heterogeneous provenance data interoperability?*

Many solutions were assessed in this SLM. Some aim to be WfMS-agnostic and others that impose a less harsh learning curve on prospect users, using storage solutions, data models, and query languages that they are already familiar with. However, none of those fully satisfies all these criteria. All the evaluations of the solutions are synthesized in Table 3. The rows are shortened due to space restrictions and are described following: (i) p-prov: The solution is able to capture prospective provenance; (ii) r-prov: The solution is

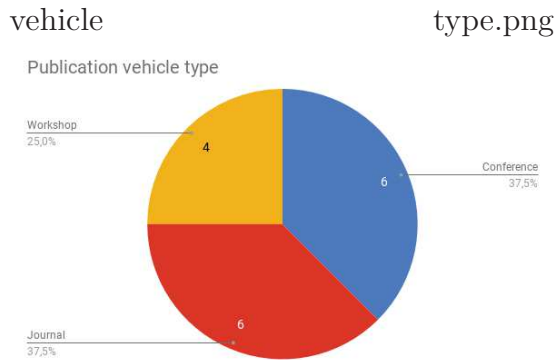


Figure 5 – Vehicle type in which the papers were published.

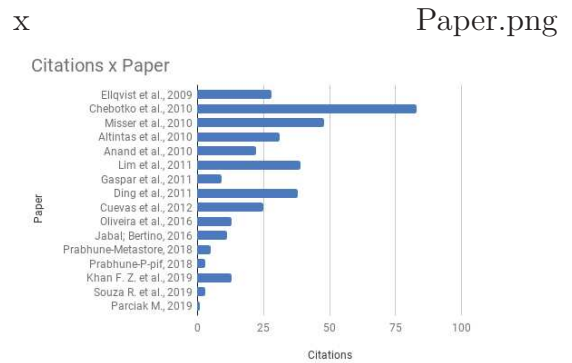


Figure 6 – Number of citations each paper has on July, 2020.

able to capture retrospective provenance; (iii) Evolution: The solution is able to capture evolution provenance; (iv) Query: The solution supports a query language that users are already familiar with. A full score is given if the authors propose at least SPARQL and SQL constructs. In case they support only one of those or propose another querying solution that aims to attenuate the users' learning curve, a half score is given; and (v) Model: The solution uses a data model already defined in the literature. In case it extends one, a half score is given.

Table 3 – Comparative table that synthesizes all aspects assessed in this SLM.

	Ellqvist et al., 2009	Chebotko et al., 2010	Misser et al., 2010	Anand et al., 2010	Altintas et al., 2010	Lim et al., 2011	Gaspar et al., 2011	Ding et al., 2011	Cuevas-Vincentin et al., 2012	Oliveira et al., 2016	Jabal, Bertino, 2016	Prabhune-Metastore, 2018	Prabhune-P-pif, 2018	Parciak M., 2019	Souza R. et al., 2019	Khan F. Z. et al., 2019
p-prov.	≡	∩	≡			≡	≡		≡	≡		≡	≡		≡	
r-prov.	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡	≡
Evolution		∩							≡	≡		≡	≡			
Query	∩	∩	∩	∩	∩	∩	≡	∩	∩		∩	∩	∩	∩		
Model			∩		≡	≡	≡	∩	∩	≡		≡	≡	≡	∩	≡

(MQ2) What query languages are the most used in solutions that support interoperability across heterogeneous provenance graphs?

This mapping question aim to assess the most used query languages in this context, guiding future research efforts. The results are synthesized in Table 4. There is a clear preference for SQL and SPARQL (query languages that most WfMSs natively support). Besides those who propose their own querying infrastructure (ELLQVIST et al., 2009; ALTINTAS et al., 2010; ANAND et al., 2010; CUEVAS-VICENTTIN et al., 2012). Oliveira

et al. (2016) use Prolog constructs to query the data and Jabal and Bertino (2016) use Cypher, Neo4j’s query language. Given the natural representation of provenance metadata as a graph, the latter may be a viable alternative solution to the more traditional relational and RDF stores.

Table 4 – Query languages supported by each solution.

	Ellqvist et al., 2009	Chebotko et al., 2010	Misser et al., 2010	Anand et al., 2010	Altintas et al., 2010	Lim et al., 2011	Gaspar et al., 2011	Ding et al., 2011	Cuevas-Vincentin et al., 2012	Oliveira et al., 2016	Jabal; Bertino, 2016	Prabhune-Metastore, 2018	Prabhune-P-pif, 2018	Parciak M., 2019	Souza R. et al., 2019	Khan F. Z. et al., 2019
SPARQL		≡					≡	≡				≡	≡		≡	
SQL			≡			≡	≡				≡					
API	≡											≡				≡
QLP				≡	≡											
RPQ									≡							
Prolog										≡						
Cypher											≡			≡		

(MQ3) *What models are the most used to represent provenance data?*

This mapping question aims to assess the most used models in this context, guiding future research efforts. The results are synthesized in Table 5. The \simeq sign indicates an extension of the model. Even though OPM and PROV are the more frequent used models, more recent works use the ProvONE data model, since it was specifically designed to attend WfMSs demands. It is worth mentioning that Chebotko et al. (2010) propose a storage solution that is not constrained by any model and, therefore, its column in Table 5 is empty.

(MQ4) *What are the existing gaps that justify the improvement of the current state of the art?*

None of the proposed approaches fully satisfy the criteria analyzed in this SLM. Even though all of those support interoperability across heterogeneous provenance metadata, very few of those regard for usability. The recent work of Souza et al. (2019) leverages a Polystore approach, using multiple storage solutions and query languages through a single interface, granting a wider array of options to users, however they opted for the PROV data model, constraining the solution to r-prov queries.

Table 5 – Provenance model used by each solution.

	Ellqvist et al., 2009	Chebotko et al., 2010	Misser et al., 2010	Anand et al., 2010	Altintas et al., 2010	Lim et al., 2011	Gaspar et al., 2011	Ding et al., 2011	Cuevas-Vincentin et al., 2012	Oliveira et al., 2016	Jabal; Bertino, 2016	Prabhune-Metastore, 2018	Prabhune-P-pif, 2018	Parciak M., 2019	Souza R. et al., 2019	Khan F. Z. et al., 2019
OPM			∩		≡	≡	≡	∩	∩							
PROV														≡	≡	≡
ProvONE										≡		≡	≡			
Original	≡			≡							≡					

3.4 THREATS TO VALIDITY

Many papers excluded in the filtering process proposed a solution to support interoperability between WfMSs executions since the scope of this SLM is defined to be a subset of this area, *i.e.*, interoperability between provenance data generated by any WfMSs. Since there is a strong intersection between these two research topics, there is the chance of existing a WfMSs interoperability solution that can also support interoperability to the provenance data generated. The inability to identify these solutions is the first threat to this work. Moreover, some papers were not available using the university’s credentials in the knowledge bases. The authors of those papers were contacted, but not all of them answered, so some papers were left unchecked. Even though other researchers evaluated the search string and keywords table, they may not contemplate all the works in the area. Furthermore, only one researcher conducted the filtering process. The bias would have been reduced if this process was done by a group. Finally, Google Scholar usage to bridge the gap between when the SLM was initially conducted and when this work was published is the final identified threat.

3.5 FINAL REMARKS

In this chapter, we report an SLM conducted to have a better understanding of the research landscape and the current state of the art, integrated analysis over heterogeneous provenance data, qualitatively assessing them by the following perspectives: *(i) Completeness*, *(ii) User Adaptability* and *(iii) Extensibility*. Its contributions are twofold: (i) it provides an overview of the research area, identifying the most used provenance models and query languages; (ii) each work is individually presented and discussed, providing qualitative analysis of these solutions based on metrics mentioned above, guiding future research efforts.

4 POLYFLOW: INTEGRATING HETEROGENEOUS PROVENANCE GRAPHS

The proposed approach, named **Polyflow**, was developed to integrate heterogeneous databases that represent heterogeneous provenance graphs of workflow executions. In this section, we present an extension of the formalism initially presented in Section 2, the data mapping process and the proposed architecture of **Polyflow**.

4.1 FORMALISM

Polyflow enables the user to submit integrated queries across multiple provenance graphs. Thus, in the context of this dissertation, it is important to formalize the concept of *Provenance Database*. A provenance database \mathfrak{S} can be defined as a set of θ provenance graphs G_p , where $\mathfrak{S} = \{G_{p_1}, G_{p_2}, \dots, G_{p_\theta}\}$. For each $G_{p_\theta} \in \mathfrak{S}$, **Polyflow** must be able to perform queries over these graphs. Thus, a *query based on parameters' values over multiple provenance graphs* can be defined as $Q_M(S, \mathfrak{S}) \Leftrightarrow \{G_{p_\theta} \in \mathfrak{S} | \exists p_m \in G_{p_\theta} \wedge \lambda(\star, \zeta_m(G_{p_\theta}.p_m), v_m) \wedge (p_m, v_m) \in S\}$, where:

(i) Q_M is a set of pairs $S = \{(p_1, v_1), (p_2, v_2), \dots, (p_m, v_m)\}$, where p_m is the queried parameter and v_m is the reference value; and

(ii) λ is the function that compares the value v_m of a parameter p_m to its value on the graph G_{p_θ} through $\zeta_m(G_{p_\theta}.p_m)$, using any relational algebra operator \star (e.g., σ , Π , \bowtie , etc.).

Consider two provenance graphs $G_{p_1}, G_{p_2} \in \mathfrak{S}$ that present semantically equivalent parameters p_m and p'_m ($p_m \equiv p'_m$), but with different names. Thus, given a set $P_\alpha = \{p_1, p_2, \dots, p_\mu\} \in G_{p_1}$ and $P_\beta = \{p_1, p_2, \dots, p_\nu\} \in G_{p_2}$ and a transformation language Υ , we should be able to find a mapping $v \in \Upsilon$ where each $p_\mu \in P_\alpha$ and $p_\nu \in P_\beta$, $\tau(p_\mu) = p_\nu$. Hence, in heterogeneous provenance databases, the most complex task is implementing queries for parameters' values $Q_M(\Gamma, \mathfrak{S})$ (where $\Gamma = S_\alpha \cup S_\beta$, $S_\alpha = \{(p_1, v_1), (p_2, v_2), \dots, (p_\mu, v_\mu)\}$ e $S_\beta = \{(p_1, v_1), (p_2, v_2), \dots, (p_\nu, v_\nu)\}$), considering the mapping τ between parameters of workflow traces represented in different graphs.

4.2 MAPPING LOCAL SCHEMAS TO PROVONE

As aforementioned in Section 4.1, a challenge to provide integrated provenance analysis is to define the mapping τ between the parameters in workflow traces represented in heterogeneous graphs. This dissertation proposes an approach to query these graphs in an integrated manner from heterogeneous databases. The adopted strategy is based on *mediation* (ÖZSU; VALDURIEZ, 2011), which is the reconciliation made to enable the translation of the data described by local schemas to a global schema. **Polyflow** adopts

the ProvONE model as CGS (or CCM). Thus, the strategy adopted in this dissertation is to map schemas from different sources to the ProvONE model.

Such mappings are used by the components of *Polyflow* (*Query Resolvers* and *Entity Mappers*, explained in Section 4.3) to execute the queries, whether in a single or multiple provenance graphs. In *Polyflow* there are two types of possible mappings: (i) 1-1: where two entities present equivalent granularity; (ii) 1-N: when an entity of the Global Schema is associated with the composition of several entities of the Local Schema, *i.e.*, the granularity level is different. An entity that is represented by a join between two tables in a RDBMS would be an example of a 1 – 2 relation. *Polyflow* supports other integration operations such as *Union* for the relational island.

Note that N-1 and N-M can be written as N 1-1 and 1-M mappings, respectively. Fig. 7(a) shows a fragment of the *Local Schema* of Swift/T (WOZNIAK et al., 2013) provenance database and its mapping to ProvONE. The dashed arrows show the mappings between the two schemas. Since the granularity between the models is different, only ProvONE entities involved in the mapping were represented. Mappings are represented by JSON objects (*i.e.*, Entity Mappers, explained in the following Section). Fig. 7(b) shows the Mapper of the *APP_EXEC* entity (Swift/T) to ProvONE’s *Execution*.

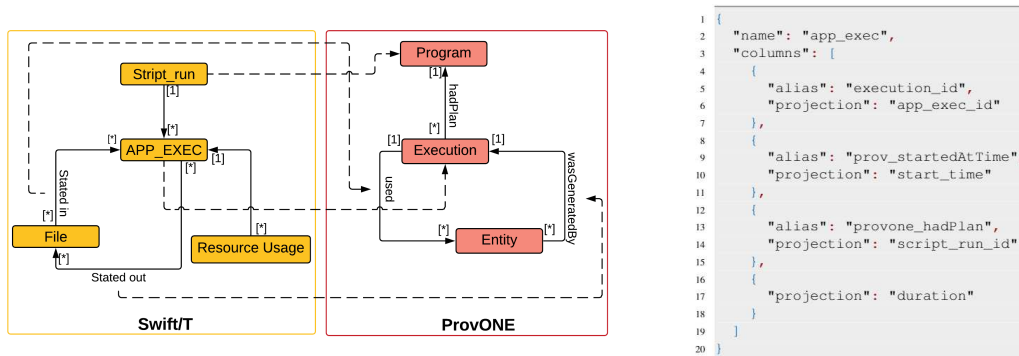


Figure 7 – (a) Mapping Swift/T schema to ProvONE; (b) *Mapper* of the *APP_EXEC* entity to *Execution* entity.

4.3 ARCHITECTURE OVERVIEW

Polyflow was designed to support provenance interoperability, based on the concept of Polystore systems that provide logical data interoperability. In this way, *Polyflow*’s architecture follows the recommendations of Polystore architectures. In a Polystore approach, the various storage mechanisms are distinct and accessed in an isolated way using their own query mechanisms and a common interface. An overview of the architecture is presented in Fig. 8. The *Polyflow* architecture is composed of four layers: (i) *Query Interface*, (ii) *Data Source*, (iii) *Mediation Layer* and (iv) *Query Processing Layer*. The

source code of the Polyflow and all the mappings performed in the experiments presented in this dissertation are available at <https://github.com/yanmendes/polyflow>.

The *Data Source* layer represents the data sources that contain the multiple provenance databases $\mathfrak{S}_1, \mathfrak{S}_2, \dots, \mathfrak{S}_f$, each of which is represented in different formats. All provenance data is collected by WfMS during (or after) a workflow execution (step [1] in Fig. 8) and is stored in a provenance database (step [2] in Fig. 8). Each provenance database is related to a *island* of the Polystore model (*e.g.*, relational, JSON, XML). Polyflow currently supports PostgreSQL, MySQL, and BigDAWG data sources. This way, a data source is a PSQL/MySQL URL or a BigDAWG endpoint. In a more general manner, it is a URI to resources across the web (*e.g.*, databases, files) that are mediated by Polyflow.

The *Mediation* layer is the core of Polyflow. In this layer, all mappings between the multiple islands are defined. The Mediation layer has three main components: (i) Global Schema, (ii) Local Schema and (iii) Entity Mappers. *Local Schemas* are data models associated with each provenance database in the data source layer. The *Global Schema* is the canonical model used to perform queries in the Polyflow. Although the architecture allows the usage of multiple Global Schemas, at the moment we only consider ProvONE. Finally, *Entity Mappers* are software modules that exploits encoded knowledge about certain sets or subsets of data to create information for a higher layer of applications (WIEDERHOLD, 1992). *Entity Mappers* perform mappings between the Local Schema and the Global Schema. Note that the Entity Mappers have to be implemented by users who are aware of the data models. Entity Mappers are consumed by *Query Resolvers* (explained next) when processing provenance queries. It is important to emphasize that implementing Entity Mappers may be an arduous task. All *Entity Mappers* are persisted in JSON format in Polyflow’s internal catalog. The JSON format was chosen for its popularity and simplicity, mitigating the learning curve that other more semantic formats (*e.g.*, RDF).

The *Query Processing* layer is responsible for receiving a query from the user and, based on the *Entity Mappers*. The *Query Resolver* is responsible for expanding queries based on a Global Schema (*i.e.*, ProvONE) into valid queries in the Local Schemas (step [5] in Fig. 8). The *Query Resolver* works similarly to the *shim* operator of the PolyStore architecture. Polyflow is also capable of resolving aggregations between pairs of entities of the Local Schema recursively, *i.e.*, a *Entity Mapper* can be composed of pairs of *Entity Mappers*. In its current version, Polyflow supports *Query Resolvers* for PostgreSQL, MySQL, and BigDAWG. Since Polyflow aims to be technology-agnostic, different data sources can be added by implementing new interfaces and **Query Resolvers**. Due to space restrictions, refer to the project’s Github repository for more details.

The *Query Interface* is the area where the user (or groups of users) submits queries

using the provided Polyflow endpoint (step [4] in Fig. 8). It is important to highlight that the queries submitted to the interface follow the Polyflow standard. The queries are SQL-like and must follow the syntax *mediator-name[entity-to-be-mediated]*. Queries are expanded at runtime by replacing mediators referenced in the query with a subquery using the elements present in the associated *Entity Mapper*.

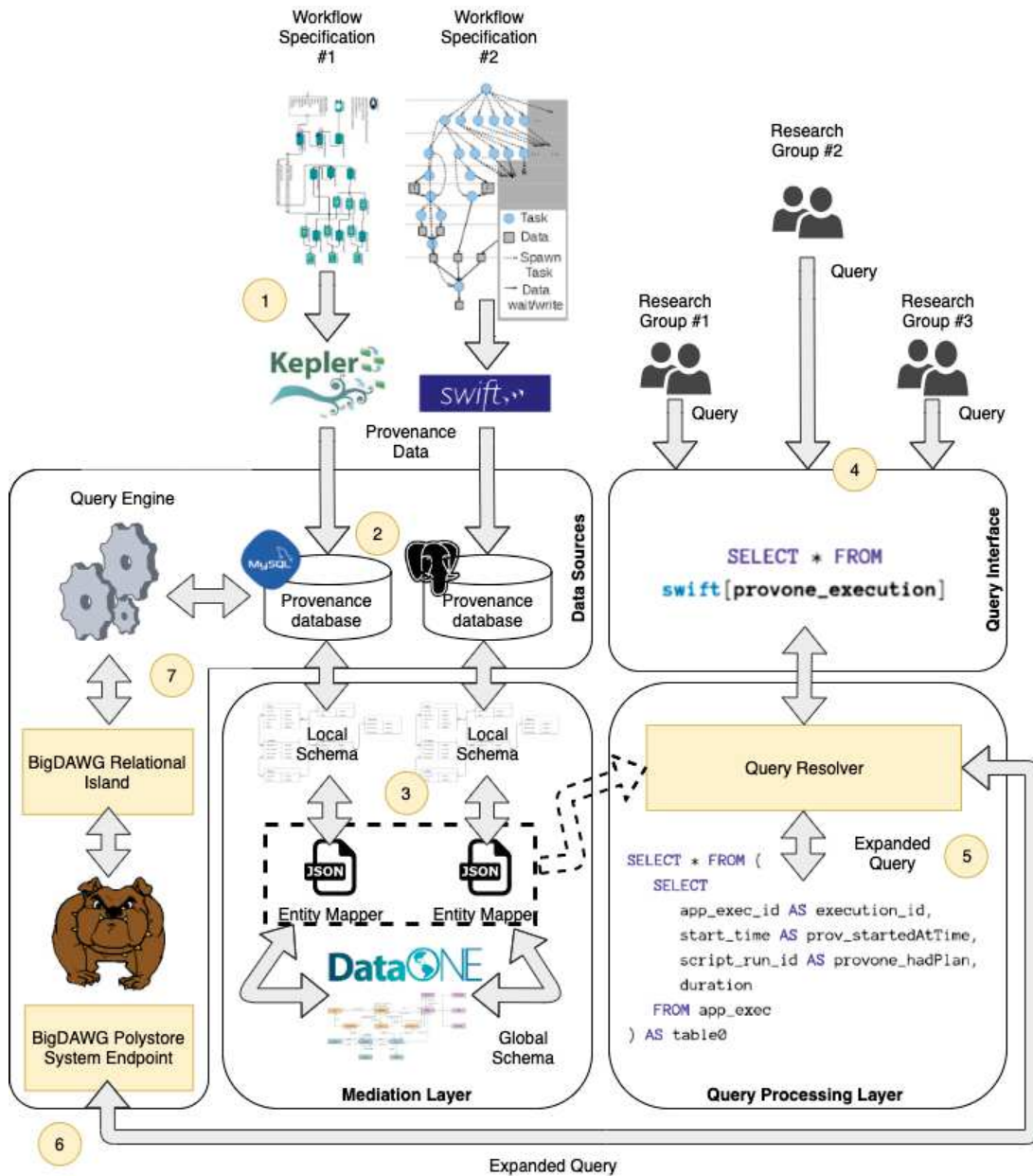


Figure 8 – Polyflow architecture.

4.4 POLYFLOW IN ACTION

In this section, we briefly expand formalism described in this chapter into use scenarios to illustrate the solution's proposal more clearly.

`Polyflow` was implemented in Node.js for its innate capability of dealing with I/O and HTTP requests asynchronously and popularity. This feature enables a looser coupling between `Polyflow` and the underlying data sources, making it possible to return partial results as queries finish processing. We've implemented a distributed-compliant software by providing a network interface as its only entry-point. This layer was implemented using GraphQL, an open-source interface that works on top of TCP/UDP that describes itself as "a query language for APIs." The main feature that compelled us to make this choice is its self-documenting capabilities, making consumption and discoverability of endpoints easier. It also provides a playground where users can explore the schema we've been using as our User Interface (UI). `Polyflow` has an internal catalog powered by a RDBS where *Data Sources*, *Mediators* and *Entity Mappers* are stored. Its schema is illustrated in Fig 9. The *slug* dimension present in all tables are the keys used in submitted queries to `Polyflow` that identify the *Mediator* and *Entity* that are being used (e.g. *swift* and *provone_execution* in the example showcased in Fig. 10). Slugs are normalized strings that don't contain special characters and spaces. All transactions are negotiated with `Polyflow` through the GraphQL API - readers can refer to our Github repository¹ for the exact signatures of our APIs).

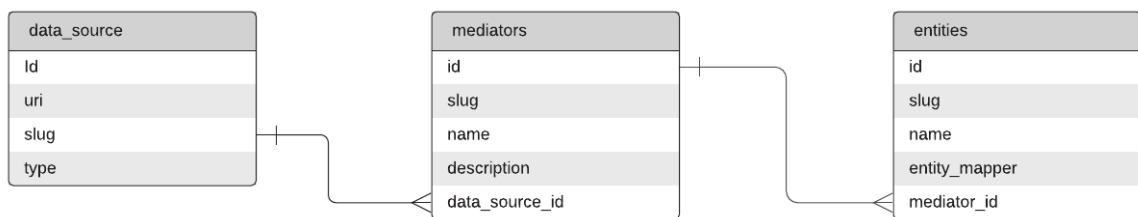


Figure 9 – `Polyflow` relational schema

Assuming an established connection with our Swift/T *Data Source* and existence of a *Mediator*, also named Swift, Fig. 10(a) illustrates how the *Entity Mapper* would look like for the *Execution* ProvONE entity.

When querying for all *Executions*, users submit the query to a GraphQL endpoint, illustrated in Fig 11 (step [4] in Fig. 8). Fig. 10(b) illustrates the query expansion performed by `Polyflow` (step [5] in Fig. 8).

Fig 10 showcases a 1-1 granularity, *i.e.*, the *Execution* entity's projection on the *Data Source* accesses just one entity (*app_exec*). `Polyflow` also supports 1-N mappings, where an *Entity Mapper* is recursively composed by relations in the *Data Source*. Fig. 12(a) illustrates how ProvONE's *Program Entity Mapper* for Kepler's *Data Source* would be defined:

¹ <<https://github.com/yanmendes/polyflow>>

```

{
  "name": "app_exec",
  "columns": [
    {
      "alias": "execution_id",
      "projection": "app_exec_id"
    },
    {
      "alias": "prov_startedAtTime",
      "projection": "start_time"
    },
    {
      "alias": "provone_hadPlan",
      "projection": "script_run_id"
    },
    {
      "projection": "duration"
    }
  ]
}

```

```

SELECT
  *
FROM swift[provone_execution]

<=>

SELECT * FROM (
  SELECT
    app_exec_id AS execution_id,
    start_time AS prov_startedAtTime,
    script_run_id AS provone_hadPlan,
    duration
  FROM app_exec
) AS table_0

```

Figure 10 – (a) Entity Mapper for the Execution ProvONE entity in Swift/T’s database; (b) Expansion for the query over all ProvONE Executions in Swift/T’s database.

```

1 query {
2   query(query:"bdrel(select * from swift[provone_execution])")
3 }

```

```

{
  "data": {
    "query": [
      {
        "swift_execution_id": "phylo-run002-2390439672:fastaNumbered-41slcp3n",
        "swift_prov_startedAtTime": "2017-02-09 12:30:14.899-0200",
        "swift_provone_hadPlan": "phylo-run002-2390439672",
        "duration": "10.226"
      },
      {
        "swift_execution_id": "phylo-run002-2390439672:fastaNumbered-pzrlcp3n",
        "swift_prov_startedAtTime": "2017-02-09 12:30:17.093-0200",
        "swift_provone_hadPlan": "phylo-run002-2390439672",
        "duration": "8.717"
      }
    ]
  }
}

```

Figure 11 – Querying the Execution ProvONE entity in Swift/T’s database.


```

{
  "entity1": {
    "name": "actor",
    "alias": "a",
    "columns": [{
      "alias": "program_id",
      "projection": "a.id"
    }]
  },
  "entity2": {
    "name": "entity",
    "alias": "e",
    "columns": [{
      "alias": "label",
      "projection": "e.name"
    }]
  },
  "columns": [
    {
      "alias": "program_id",
      "projection": "a.id"
    },
    {
      "alias": "label",
      "projection": "e.name"
    }
  ],
  "type": "INNER",
  "params": ["a.id", "e.id"]
}

```

```

SELECT
  *
FROM kepler[provone_program]

<=>

SELECT * FROM (
  SELECT
    a.id AS program_id,
    e.NAME AS label
  FROM actor AS a
  INNER JOIN entity AS e
  ON a.id = e.id
) AS table_0

```

Figure 12 – (a) Entity Mapper for the Program ProvONE entity in Kepler’s database; (b) Expansion for the query over all ProvONE Programs in Kepler’s database.

Even though we have one level of recursion in this example, Polyflow supports as many as needed. Fig. 12(b) showcases how Polyflow expands a query over all Kepler’s *Programs*.

4.5 FINAL REMARKS

In this chapter, we introduce Polyflow, a tool built on top of a polystore system that aims to provide support to syntactic and semantic interoperability. Polyflow mediates incoming queries described by a Global Schema (*e.g.* ProvONE), and rewrites them, during run time, making them processable by underlying engines that have data described by other schemas in the same domain (*e.g.* provenance).

5 EVALUATION AND RESULTS

In this Chapter, we evaluate `Polyflow` in two different ways. We firstly assess the technical implementation in two dimensions: (a) *Completeness* and (b) *Efficiency*. In terms of *Completeness*, we show that `Polyflow` can query all provenance graphs across distributed databases. This evaluation aims to showcase its feasibility, *i.e.*, that the architecture proposed in this work is a good solution for the problem. In terms of *Efficiency*, we investigate if `Polyflow` adds an acceptable overhead to query the provenance databases, *i.e.*, that the architecture could be used in a production environment.

Later, we conduct a usability evaluation with researchers that are either experts or are knowledgeable in provenance. In the experiment, subjects are asked to write provenance queries using `Polyflow` and compare with the current state of practice, *i.e.*, integrating the data and querying the integrated database. This study's objective is to assess whether or not the solution is practical enough to be a viable state of practice.

Both of these assessments are made using real workflows used by researchers from the Rede Avançada em Biologia Computacional, a Brazilian network of geographically distributed researchers that work on similar topics and want to compare their results.

5.1 EVALUATION CONTEXT

In the experimental evaluation, we used the *Rede Avançada em Biologia Computacional (Rabico)*¹, a Brazilian network whose primary goal is to develop the computational apparatus to support data analysis of biological models. The network has members of multiple institutions and universities such as LNCC, COPPE/UFRJ, and UFRGS. In this project, two or more geographically distributed teams may work on similar research topics such as phylogenetic analysis (OCAÑA et al., 2011). Each team adopts slightly different approaches, resulting in different workflows that can be potentially managed by different WfMSs. However, they maintain common goals and, therefore, their results should be comparable.

In order to evaluate `Polyflow`, we have chosen the phylogenetic analysis experiment. A phylogenetic analysis experiment receives as input a dataset composed of DNA, RNA, and protein sequences to generate a phylogenetic tree that represents the evolutionary relationship between the organisms (illustrated in Fig. 13). It has seven well-defined steps (Fig. 14): (i) *Import Datasets*: data is collected from different biological repositories, such as RefSeq²; (ii) *Enumerate Sequences*: The obtained sequences are identified and enumerated (it is an optional step); (iii) *Multiple Sequence Alignment*: sequences are aligned (by programs such as MAFFT, Kalign, ClustalW, Muscle or ProbCons), *i.e.*,

¹ <<https://www.labinfo.lncc.br/rabico/>>

² <<https://www.ncbi.nlm.nih.gov/refseq/>>

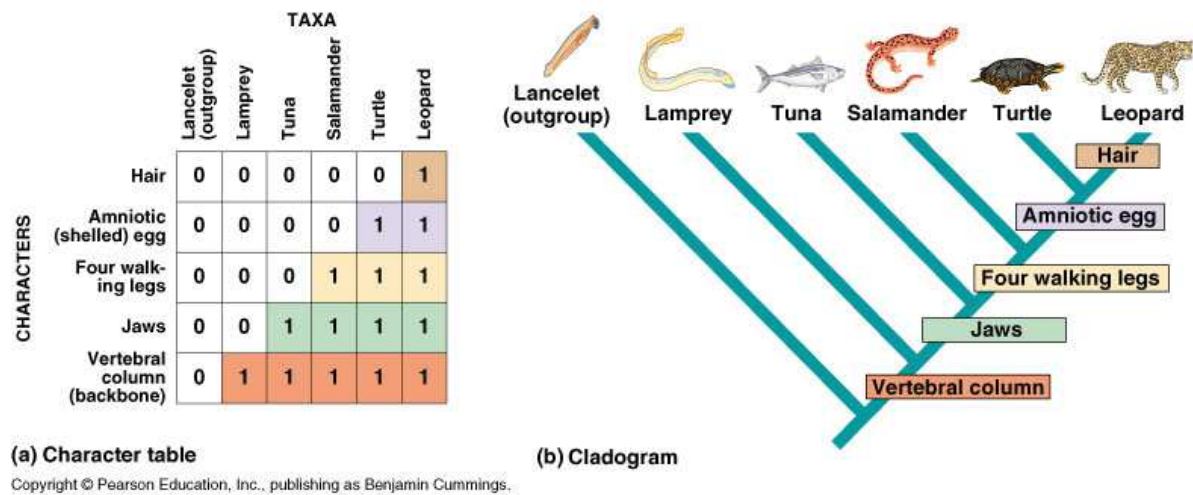


Figure 13 – Illustration of a phylogenetic tree. Credit: <<https://ksuweb.kennesaw.edu/~jdirnber/Bio2108/Lecture/LecPhylogeny/LecPhylogeny.html>>

identify similar regions that may be consequence of functional, structural or evolutionary relations; (iv) *Sequence Conversion*: the aligned sequences are converted to the PHYLIP format; (v): *Elect Evolutionary Model*: the best evolutionary model for the aligned sequences is elected (this is the most compute-intensive step of the experiment); (vi): *Filter Sequences*: sequences can be filtered or trimmed to reduce the complexity of the generated phylogenetic tree (it is an optional step); and (vii): *Phylogenetic Tree Generation*: the phylogenetic tree is finally generated, representing evolutionary relations between the organisms.

Although the Phylogenetic Analysis experiment can be implemented in many ways, in this dissertation, we consider two implementations named SciPhy (OCAÑA et al., 2011) and SwiftPhylo (MONDELLI et al., 2018). Both workflows are variations of the same experiment in which results must be analyzed jointly. SwiftPhylo implements all activities, except for *Import Datasets* because it assumes that data is already available for processing. On the other hand, SciPhy does not implement any optional activity, *i.e.*, it has five steps. Moreover, SwiftPhylo uses MAFFT as the alignment program while SciPhy executes all aforementioned sequence alignment programs and chooses the one that produces results with the best quality. Finally, SciPhy was implemented in Kepler WfMS while SwiftPhylo in Swift/T WfMS.

5.2 FEASIBILITY STUDY

Considering the scenarios shown in the previous Section, we assume that a feasibility study's expected outcome would be to allow researchers to *query integrated provenance data* generated by SciPhy and SwiftPhylo workflows. We use ProvONE as Global Schema because:

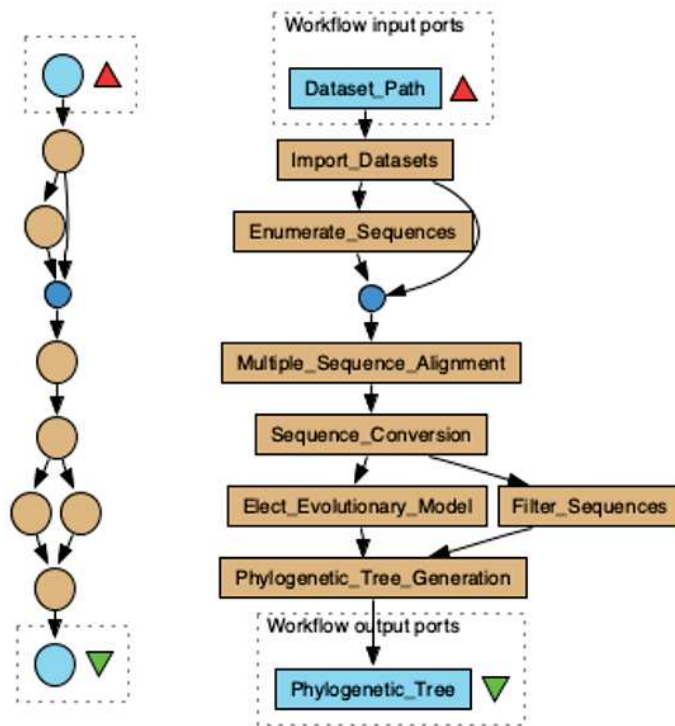


Figure 14 – Specification of the phylogenetic analysis experiment (represented using Apache Taverna notation)

- it is able to capture p-prov, r-prov and evolution provenance;
- it is an extension of Prov-DM W3C standard for r-prov representation; and
- it is becoming a *de facto* standard for representing workflow provenance, as we show in the next section.

In the work of (OLIVEIRA et al., 2016), the authors use a Venn Diagram (illustrated in Fig. 15) to showcase every possible querying scenario using two provenance graphs.

Next, we showcase one query for each class Q to demonstrate the feasibility of Polyflow. You can refer to our Github repository³ on instructions how to run them and reproduce the experiments we run in this Chapter.

- Q1 - Retrieve all ports in Sciphy's Provenance Graph (connection between programs and their input/output parameters)

```
bdrel(select * from kepler[provone_port])
```

- Q2 - Retrieve all activity executions with their generated data for SwiftPhylo's provenance graph:

³ <<https://github.com/yanmendes/Polyflow/tree/master/examples/BigDAWG>>

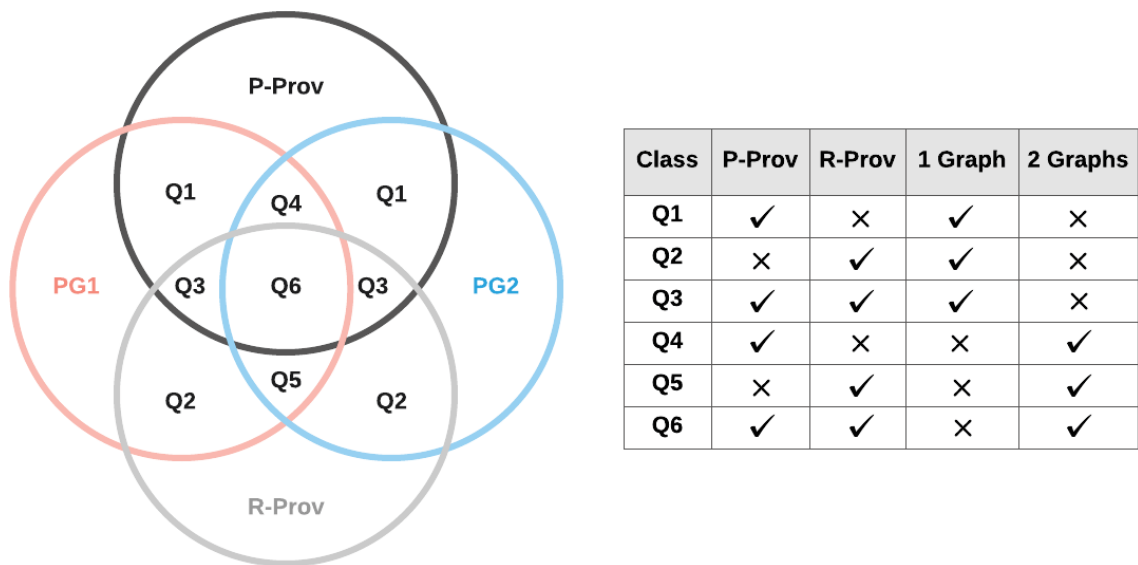


Figure 15 – Query classes for two provenance graphs. Inspired by (OLIVEIRA et al., 2016))

```
bdrel(select * from swift[provone_execution] as e
      join swift[prov_wasGeneratedBy] as wgb
      on e.swift_execution_id = wgb.swift_execution_id)
```

- Q3 - Retrieve data used in an workflow enactment and their corresponding port for Sciphy's workflow.

```
bdrel(select * from kepler[prov_usage] as u
      inner join kepler[provone_port] as p
      on u.kepler_provone_hadInPort = p.kepler_port_id)
```

- Q4- Connect Sciphy's ports to SwiftPhylo's programs.

```
bdrel(select * from kepler[provone_port]
      left join (select * from swift[provone_program]) on 1=1)
```

- Q5- Fetch all program's execution duration from both datasets*

```
bdrel(select * from
      (select duration from swift[provone_execution]) as t1
      left join (select EXTRACT(SECOND FROM
      kepler_prov_endedAtTime - kepler_prov_startedAtTime
      ) from kepler[provone_execution]) as t2 on 1=1)
```

- Q6- Fetch all programs and executions from both workflows*

```

bdrel(select * from kepler[provone_execution] as e
inner join kepler[provone_program] as p
      on p.kepler_program_id = e.kepler_provone_hadPlan
left join swift[provone_program] as sp on 1=1
left join swift[provone_execution] as se
      on se.swift_provone_hadPlan = sp.swift_program_id)

```

One point worth mentioning for clarity’s sake is the addition of the token *bdrel* to the submitted queries. As we mentioned throughout this dissertation, BigDAWG utilizes the concept of islands that defines a data model and query language. BigDAWG’s query interpreter (step [6] in 8) needs the island of destination of a query to properly assign them. Therefore, all queries are wrapped around *bdrel* so BigDAWG targets them to the relational island.

In terms of *Completeness*, we have assessed the results for each query in every configuration: (Swift/T) using only Swift/T provenance database in the underlying DBMS, (ScyPhy) using only Kepler provenance database in the underlying DBMS, (BigDAWG) using only BigDAWG and using *Polyflow*. The results are synthesized in Table 6, where the number of returned tuples for each query and configuration are presented. Moreover, we have showcased the ability to perform the queries described in the work of (OLIVEIRA et al., 2016).

Table 6 – Tuple count at every touch point through the architecture.

Query	Swift/T	ScyPhy	BigDAWG	Polyflow
Q1	0	11	11	11
Q2	3000	0	3000	3000
Q3	0	5	5	5
Q4	2	11	22	22
Q5	1200	6	7200	7200
Q6	1200	6	7200	7200

As explained in Chapter 4, when a query is submitted, *Polyflow* retrieves the entity mappers and performs a query expansion using entities and relationships from the LS. These operations (*e.g.*, Entity Mapper fetch and query expansion) add an overhead. To evaluate the added overhead, we calculate the total request time and the query execution time in the underlying DBMS. All implementation is containerized and connected via a Docker network, hosted by a 2,4 GHz Quad-Core Intel Core i5, 16GB RAM macOS Catalina 10.15.4 machine.

Each query was performed 10 times, and the first execution was discarded (cold start). All requests were sequentially submitted. One can note in Table 7 that most of the request processing time is consumed by BigDAWG, with a negligible overhead added

by `Polyflow` (3.8% in the worst case). Another tendency that one can see is that the more complex the query (*i.e.*, more resource-consuming on BigDAWG’s side), the more negligible is the overhead added by `Polyflow` (3.8% for a simple projection *versus* a 1.2% for a cross-database query Q3). It’s also important to highlight that in a distributed environment, network traffic would add a delay between `Polyflow` and the data sources, further decreasing the processing overhead relevance in the total request duration. In an ideal scenario, `Polyflow` should be physically located as near as possible to the data sources.

It is important to emphasize that these results were obtained from common consultations in the data provenance area, as defined by (OLIVEIRA et al., 2016). However, more complex queries involving domain-specific data, and not just provenance data, can be considered, which would increase the complexity of `Polyflow`. It is also important to mention that BigDAWG (which is why we represented queries followed by * in that way) presents some limitations that impairs some use cases, such as being unable to process queries that have projections, inconsistent results for queries that use the union operator and an inability to distinguish join types. A complete list of issues can be tracked in their Github repository⁴. `Polyflow` also has two limitations: the inability to process table and column aliases without the `AS` keyword and the lack of support of group by and order by statements in entity mappers. `Polyflow` also has two limitations: the inability to process table and column aliases without the `AS` keyword and the lack of support of group by and order by statements in *Entity Mappers*.

Table 7 – `Polyflow` Overhead Analysis

	Request Time (ms)				Query Execution Time (ms)				Overhead (ms)				
	min	max	avg	stdev	min	max	avg	stdev	min	max	avg	stdev	avg%
Q1	109.0	387.0	182.0	254.3	103.0	380.0	176.9	251.6	3.0	7.0	5.1	4.1	2.9
Q2	618.0	1496.0	1001.0	673.7	608.0	1484.0	982.9	670.0	7.0	36.0	18.1	28.4	1.8
Q3	130.0	384.0	215.6	241.6	127.0	370.0	209.3	233.4	2.0	14.0	6.2	11.3	3.0
Q4	319.0	461.0	359.2	118.3	314.0	457.0	354.4	118.8	3.0	10.0	4.8	6.6	1.4
Q5	386.0	731.0	571.8	366.4	380.0	684.0	560.9	347.1	4.0	47.0	10.9	38.7	1.9
Q6	1091.0	3668.0	1889.9	2171.5	1076.0	3569.0	1854.7	2114.3	5.0	99.0	32.5	92.5	1.9

5.3 USABILITY EVALUATION

In this experiment, we aimed to assess, primarily, the *usability* of the proposed solution. Since usability is a very broad term that contemplates everything under "user-friendliness" (BEVANA; KIRAKOWSKIB; MAISSELA, 1991), we hereby restrict its scope to **autonomy**, **clarity** and **ease of use**. For the first two, we want to make sure that prospective users of `Polyflow` can understand the documentation and start using it themselves. The overall experiment’s steps are described below.

⁴ <<https://github.com/bigdawg-istc/bigdawg/issues>>

Firstly, we presented the documentation on the problem and the proposed solution described in our Github repository⁵. Once participants understood the problem and the solution, we moved on to the installation of `Polyflow` and the setup of the BigDAWG integration that consists of three steps: (i) Installation of `Polyflow` itself - described on the link above; (ii) Installation of BigDAWG and (iii) Insertion of Kepler and Swift/T entity mappers on `Polyflow`'s catalog. Instructions to both of these actions are described on the examples folder of the same repository⁶.

On the pilot version, we had fewer queries as we wanted to validate the process. In the experiment itself, participants were asked to assemble queries Q1-Q6 (described in Section 5.2) from their specifications. The responsible for the experiment conducting informed them of the relations and dimensions necessary for creating the query. Then, participants were asked to compare queries submitted to `Polyflow` to the expanded ones submitted to BigDAWG. Finally, the participants were asked to fill a questionnaire (available in the appendix) to review `Polyflow` under a few different optics: (i) problem statement and documentation clarity; (ii) `Polyflow`'s functionality. We used a mix of ranking (1-5) and multiple-choice questions with open ones so participants could dive in into any issues they faced. The list of closed questions were:

- EQ1) *After reading the documentation, what's your level of clarity on the problem statement and `Polyflow` as a solution?*
- EQ2) *How hard did you find the installation process?*
- EQ3) *Given the complexity of creating and executing queries on the source databases, do you think `Polyflow` facilitates the process?*
- EQ4) *Given the problem we aim to solve and your experience during the experiment, do you think `Polyflow` is a viable solution?*
- EQ5) *From a usability standpoint, how do you rank `Polyflow`? - Added on the second iteration.*

Questions 1-2 hope to assess the level autonomy that the documentation would provide to prospect users of `Polyflow`. Questions 3-5 aims to assess, objectively, the perception of the respondent regarding `Polyflow`, *i.e.*, how do they evaluate the experience when compared to the current state of practice.

The usability evaluation was carried out with an expert and with people knowledgeable in the application domain. The execution and results are described in the next sections.

⁵ <<https://github.com/yanmendes/polyflow>>

⁶ <<https://github.com/yanmendes/polyflow/tree/master/examples/BigDAWG>>

5.3.1 Domain Expert Execution and Results

Besides the feasibility study, we also carried out an evaluation with an expert in the bioinformatics domain, named here as *Expert Researcher*. This pilot experiment aimed to assess the viability of Polyflow from the perspective of an expert user, *i.e.*, researchers.

The Expert Researcher is a member of RABICÓ project and a researcher at LNCC. She was asked to read the documentation, install Polyflow by herself, and execute the queries $Q1 - Q6$ by herself. The participant struggled to figure out by herself if she had a working version of Polyflow running because of error logs that were false alarms. Once we stepped in and clarified that those didn't affect Polyflow, she was able to perform all the queries mentioned above. Fig. 16 illustrates the subject's experience when issuing Query 1.

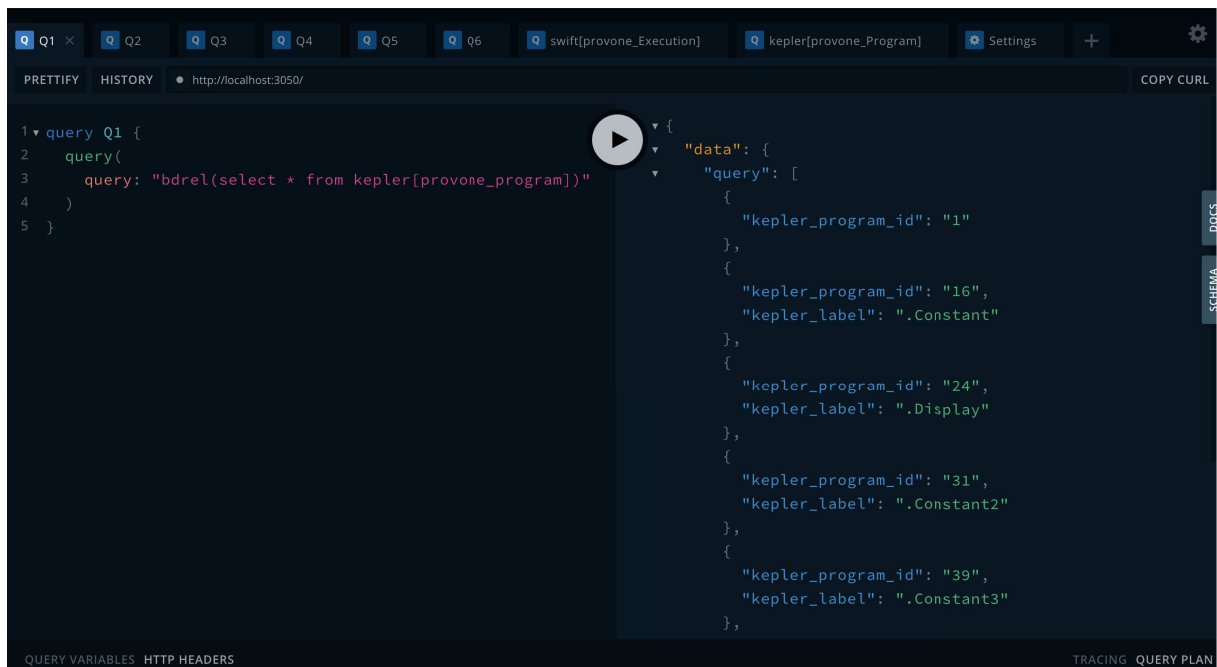


Figure 16 – Illustration of subject issuing $Q1$ to Polyflow

The Expert Researcher answered a questionnaire on March 12th, 2020, to collect her point of view, through which it was possible to identify issues that needed to be improved in Polyflow. All answers are available, in Portuguese, at this link⁷. Below, we highlight some of them.

Problem statement and documentation clarity

EQ1: The Expert Researcher assessed the documentation and problem statement as *Clear*. As we will discuss ahead, the installation process had several convoluted log messages and the documentation did not prepare her for it;

⁷ <<https://bit.ly/2OEo4bw>>

EQ2: The participant rated the difficulty of installation as *Neutral*. At this point, we had a several levels of logs being exposed to the end user, which made things unnecessarily confusing. Even though the participant had a working version of `Polyflow` and `BigDAWG` running, she did not realize it because of it.

Polyflow’s functionality

EQ3: The subject expressed a positive perception over `Polyflow`’s core functionality, *i.e.*, using a mediator to handle complex queries instead of manually aggregating and querying over the integrated data.

EQ4: The Expert Researcher agrees that `Polyflow` is a viable solution for the problem we’re proposing to solve.

All issues reported by the subject were resolved, and we developed a more robust version of the experiment with more participants, which is described in Section 5.3.2.

5.3.2 Domain Knowledge Execution and Results

Since this work happened in a particular domain (provenance) that required some degree of knowledge on relational database querying syntax (SQL), it was difficult to recruit participants that attended to both these criteria. We managed to get three volunteers who shared the same profile: postgraduate (Ph.D. and MSc) students and researchers in the provenance domain with some degree of knowledge of SQL syntax.

Differently from the pilot version, this was a fully supervised experiment, *i.e.* the authors were present during all sections of the experiment. Firstly, participants were asked to read the documentation and install `Polyflow`. The authors tried to help them only when absolutely necessary to gauge prospective users’ autonomy. Once they had a working version of `Polyflow` and `BigDAWG` running, the authors helped them build Q1. From this point on, participants were given the specification of a query written in plain English (e.g., *Retrieve all ports in Kepler’s Provenance Graph*) and the name of the table and columns pertinent to it. With that information, they were asked to assemble the query by themselves, and, again, the authors tried to intervene only when necessary. Finally, participants were shown the expanded version of Q1 – Q6 and asked to compare, in terms of complexity, to the query they submitted to `Polyflow`. Fig. 17 illustrates a participant issuing Q6 to `Polyflow`.

Due to a closer proximity to the participants this time around, we could interact with them and write notes based on observation. Therefore, the discussion we showcase is primarily based on the questionnaire’s results, but they are not limited to it. The questionnaire and its results can be found here⁸.

⁸ <<https://bit.ly/2GH68fP>>

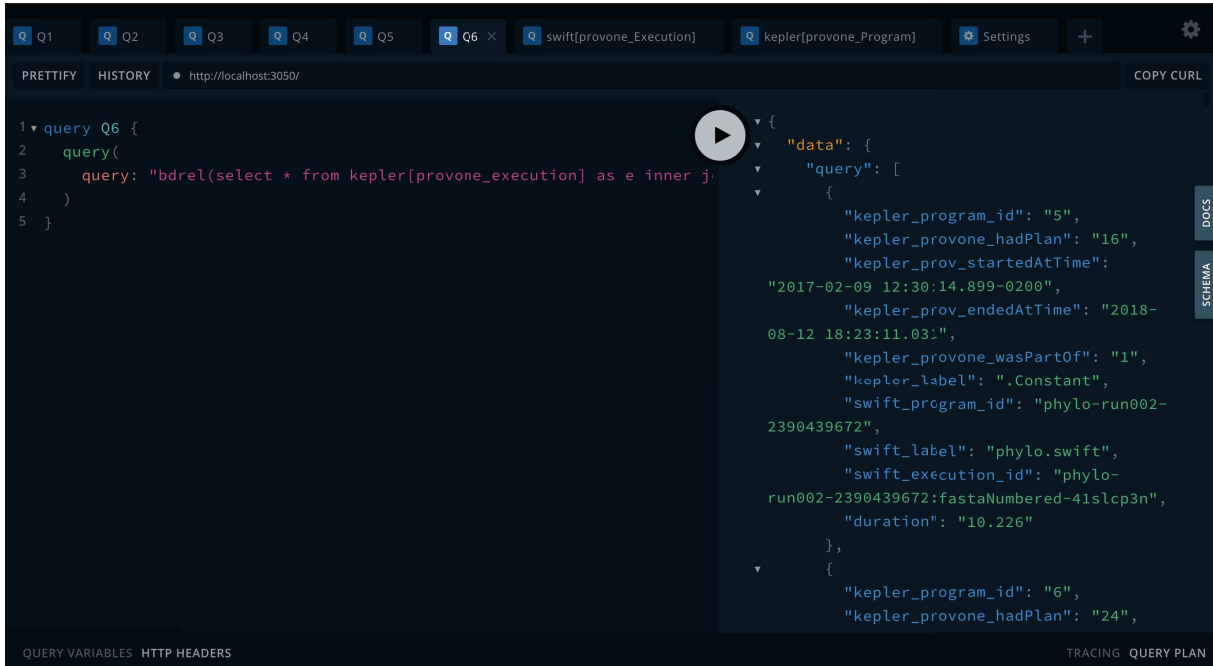


Figure 17 – Illustration of participant issuing `Q6` to Polyflow

Moreover, after revisiting the experiment’s process and objective, we understood that we were only gauging usability indirectly throughout multiple questions, which was useful to identify problems throughout different steps of the process (awareness of the problem, solution, installation, and execution). However, we also wanted to measure how they rate Polyflow, from a usability standpoint, as a whole, and not from individual components and therefore added **EQ5** (described below).

Problem statement and documentation clarity

EQ1: All participants ranked the documentation and the problem statement as *Very clear*. One constructive criticism pointed by a participant is that could "sell" Polyflow a little more by comparing simple queries mediated by Polyflow to the ones it expands.

EQ2: Two participants rated it as *Easy* while the other one rated as *Very hard*. The three installation steps are indeed confusing and scattered around two repositories. We already leverage the usage of a containerized infrastructure for these three pieces of software. An actionable that would improve the user’s perception on this front would be to orchestrate all of these services and jobs through Docker compose.

Polyflow’s functionality

EQ3: All participants expressed a positive perception over Polyflow’s core functionality, *i.e.*, using a mediator to handle complex queries instead of manually aggregating and querying over the integrated data.

EQ4: All users agree that Polyflow is a viable solution for the problem we’re proposing to solve, which is an incredibly rewarding result that validates, to an extent,

our work.

EQ5: From a usability standpoint, results were scattered between neutral and very positive. We are using Apollo GraphQL’s playground⁹ as our interface, which isn’t very user-friendly. A clear limitation is the inability of using multiple lines in a query. Creating our own interface should also facilitate user’s interaction.

5.3.3 Discussion

After analysis and deliberation over the results, we synthesized the following summary of the results:

- *Messy and convoluted logs and warning messages:* On the pilot versio. the subject had a working version of `Polyflow` and `BigDAWG` running on her machine, but she did not realize it, because there were several unnecessary logs and warning messages (*e.g.*, `BigDAWG` failed attempt to connect to a `SciDB` instance, which was not used at all). Such messages confused and jeopardized the experience. This feedback was incorporated and the logs were hidden in the actual experiment;
- *Complicated installation process:* One subject found unnecessarily challenging the process of installation, which we agree as it is spread across different repositories and folders. We can leverage the usage of a containerized infrastructure for all services to create a single installation point through composition of containers;
- *Subjects longed for a friendlier UI:* Both on the pilot and actual experiment, subjects reported that the UI was not very friendly for long queries since Apollo’s GraphQL playground doesn’t support queries with multiple lines. `Polyflow` was designed with an API-first mentality, being able to seamlessly integrate with APIs and UIs. Unfortunately, we did not have enough time to create a friendlier UI by the time of this dissertation’s completion, but we plan on doing it as future work;
- *A viable and easy solution to perform integrated queries over provenance graphs:* This was a consistent feedback throughout both phases of the experiment, where subjects evaluated the solution as both viable and simpler than manually integrating the data.

5.4 FINAL REMARKS

In this Chapter, we evaluated `Polyflow` in two different ways. We firstly assess it in terms of *feasibility* and show that `Polyflow` is capable of interoperating p-prov and r-prov provenance queries over distributed provenance databases with negligible overhead.

⁹ <<https://www.apollographql.com/docs/apollo-server/testing/graphql-playground/>>

As for *usability*, we assess *autonomy*, *clarity* and *ease of use* by conducting an experiment with a domain expert and post-graduate candidates that research provenance. They were asked to read the documentation, install and submit several provenance queries to `Polyflow`, and asked to compare to the current state of practice*i.e.*, how they currently conduct integrated analysis over heterogeneous and distributed provenance databases. Results show that, even though improvements could be made, `Polyflow` is a viable and easy solution to perform integrated queries over provenance graphs.

6 FINAL REMARKS

This dissertation showcases **Polyflow**, a polystore-compliant approach that enables integrated analysis of heterogeneous provenance graphs distributed across distributed and heterogeneous databases. It is a problem faced continuously by distributed research teams such as the members of Rabicó, a Brazilian network of researchers that use computational simulations to investigate similar research topics and want to compare their data, workflows, and results.

The integrated analysis can be broken down to two problems: (i) syntactic interoperability and (ii) semantic interoperability. In this work’s context, the first is the ability of two or more database engines to be queried jointly despite differences in language, interface, and format. The second is the ability to integrate datasets belonging to the same domain (*e.g.* provenance) that are expressed using different models and granularity. This work’s contributions are threefold:

Firstly, the authors conducted an SLM to understand the existing gaps in the state of the art of provenance data **syntactic and semantic interoperability**. With its results, they were able to identify that no solutions in the state of art satisfy the criteria used to evaluate them, namely: (i) **Completeness**, (ii) User **Adaptability** and (iii) **Extensibility**.

With a clearer understanding of how state of the art could be improved, the authors propose a solution named **Polyflow** that provides support to **semantic interoperability** by using ProvONE as a Canonical Conceptual Model/Global Schema. It is built upon a polystore database that provides support to **syntactic interoperability**. The ProvONE data model guarantees (i) **Completeness** due to its ability to represent p-prov, r-prov and evolution provenance and (ii) **Extensibility** because it is built on top of a W3C recommendation and its recent popularity and traction in related literature.

Finally, we assess **Polyflow**’s capabilities under two perspectives (i) **Feasibility** and (ii) **Usability**. In terms of **feasibility**, we show that **Polyflow** is capable of interoperating p-prov and r-prov provenance queries over distributed provenance databases with negligible overhead. Users submit queries using ProvONE as a *lingua franca* for provenance and **Polyflow** rewrites them, during run time, to the underlying databases’ schemas. As for **usability**, we assess **autonomy**, **clarity** and **ease of use** by conducting an experiment with a domain expert and post-graduate candidates that research provenance. They were asked to read the documentation, install and submit several provenance queries to **Polyflow** and asked to compare to the current state of practice *i.e.*, how they currently conduct integrate analysis over heterogeneous and distributed provenance databases.

As for future work, creating a custom UI and simplifying the installation process are two very straight-forward improvements regarding usability. As depicted in Chapter

4, `Polyflow` is domain-agnostic and able to provide semantic interoperability based on mappings present in its catalog. With that in mind, we plan on expanding the scope of this work and show that `Polyflow` can be used to tackle the semantic interoperability problem more broadly, considering another application domain.

In this sense, we also intend to enable a more flexible global scheme definition, allowing users to define more than one global scheme and use the `Polyflow` with greater flexibility. Thus, we intend that the `Polyflow` meets the demand of different users simultaneously through the parameterization of the global scheme.

We adopted BigDAWG as the only Polystore implementation at the time. Meantime, BigDAWG impaired some querying scenarios due to its relational island inability to process some SQL statements (e.g., *union*). We also plan on providing new query interfaces that connect to other polystore systems to enable these queries. Thus, we intend to evaluate the flexibility of the `Polyflow` when integrating with other polystore systems.

In this work, we do not consider security issues in the polystore database. For the `Polyflow` to be deployed and made available, it is crucial to develop a security layer.

7 APPENDIX

7.1 APPENDIX A - PAPERS REPORTED IN SLM

- Ellqvist et al., 2009 - ELLQVIST, T. et al. Using mediation to achieve provenance interoperability. In: IEEE. *Services-I, 2009 World Conference on*. [S.l.], 2009. p. 291–298.
- Chebotko et al., 2010 - CHEBOTKO, A. et al. Rdfprov: A relational rdf store for querying and managing scientific workflow provenance. *Data & Knowledge Engineering*, Elsevier, v. 69, n. 8, p.836–865, 2010
- Misser et al., 2010 - MISSIER, P. et al. Linking multiple workflow provenance traces for interoperable collaborative science. In: IEEE. *Workflows in Support of Large-Scale Science (WORKS), 2010 5th Workshop on*. [S.l.], 2010. p. 1–8.
- Altintas et al., 2010 - ALTINTAS, I. et al. Understanding collaborative studies through interoperable workflow provenance. In: SPRINGER. *International Provenance and Annotation Workshop*. [S.l.], 2010. p. 42–58.
- Anand et al., 2010 - ANAND, M. K. et al. Approaches for exploring and querying scientific workflow provenance graphs. In: SPRINGER. *International Provenance and Annotation Workshop*. [S.l.], 2010. p. 17–26.
- Lim et al., 2011 - LIM, C. et al. Storing, reasoning, and querying opm-compliant scientific workflow provenance using relational databases. *Future Generation Computer Systems*, Elsevier, v. 27,n. 6, p. 781–789, 2011.
- Gaspar et al., 2011 - GASPAR, W.; BRAGA, R.; CAMPOS, F. Sciprov: an architecture for semantic query in provenance metadata on e-science context. In: SPRINGER. *International Conference on Information Technology in Bio-and Medical Informatics*. [S.l.], 2011. p. 68–81.
- Ding et al., 2011 - DING, L. et al. Linked provenance data: A semantic web-based approach to interoperable workflow traces. *Future Generation Computer Systems*, Elsevier, v. 27, n. 6, p. 797–805,2011.
- Cuevas et al., 2012 - CUEVAS-VICENTTIN, V. et al. Modeling and querying scientific workflow provenance in the d-opm. In: IEEE. *2012 SC Companion: High-Performance Computing, Networking, Storage and Analysis (SCC)*. [S.l.], 2012. p. 119–128.
- Oliveira et al., 2016 - OLIVEIRA, W. et al. Analyzing provenance across heterogeneous provenance graphs. In: SPRINGER. *International Provenance and Annotation Workshop*. [S.l.], 2016. p. 57–70.

- Jabal; Bertino, 2016 - JABAL, A. A.; BERTINO, E. Simp: Secure interoperable multi-granular provenance framework. In: IEEE. *e-Science (e-Science), 2016 IEEE 12th International Conference on.*[S.l.], 2016. p. 270–275.
- Prabhune-Metastore, 2018 - PRABHUNE, A. et al. Metastore: an adaptive metadata management framework for heterogeneous metadata models. *Distributed and Parallel Databases*, Springer, v. 36, n. 1,p. 153–194, 2018.
- Prabhune-P-pif, 2018 - PRABHUNE, A. et al. P-pif: a provone provenance interoperability framework for analyzing heterogeneous workflow specifications and provenance traces. *Distributed and Parallel Databases*, Springer, v. 36, n. 1, p. 219–264, 2018.

7.2 APPENDIX B - QUESTIONNAIRE USED USABILITY ASSESSMENT

10/22/2020

Clareza do problema e documentação

Clareza do problema e documentação

Formulário de avaliação da ferramenta Polyflow

***Obrigatório**

1. Após a leitura da documentação, ficou clara a proposta da ferramenta e o problema que ela se compromete a resolver? *

Marcar apenas uma oval.

	1	2	3	4	5	
Nada claro	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Muito claro

2. Caso tenha algum feedback sobre a documentação, utilize o espaço abaixo (ex: o tópico X poderia estar mais claro)

3. Qual o grau de dificuldade percebido durante a instalação da ferramenta? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muito difícil	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Muito fácil

4. Caso tenha algum feedback sobre o processo de instalação, utilize este espaço (ex: encontrei um erro inesperado na seção Y)

Solução

5. Sobre a complexidade de construção e execução de consultas, você entende que polyflow auxilia o processo? *

Marcar apenas uma oval.

- Não, prefiro executar consultas em SGBDs e integrar os esquemas manualmente
- Sim, a sintaxe SQL-like ameniza a curva de aprendizado

6. Caso possua algum feedback sobre o processo de construção e execução de consultas, descreva-o abaixo

7. Do ponto de vista de usabilidade, como você avaliaria o Polyflow? *

Marcar apenas uma oval.

	1	2	3	4	5	
Nada intuitivo	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Extremamente amigável

8. Dado o problema que Polyflow visa resolver e sua experiência no experimento, você entende que a ferramenta é uma solução viável? *

Marcar apenas uma oval.

Sim

Não

9. Caso queira dar mais algum feedback sobre o experimento, a ferramenta ou o processo, utilize o espaço abaixo

Este conteúdo não foi criado nem aprovado pelo Google.

Google Formulários

REFERENCES

- ABADI, Daniel; AGRAWAL, Rakesh; AILAMAKI, Anastasia; BALAZINSKA, Magdalena; BERNSTEIN, Philip A.; CAREY, Michael J.; CHAUDHURI, Surajit; DEAN, Jeffrey; DOAN, AnHai; FRANKLIN, Michael J.; GEHRKE, Johannes; HAAS, Laura M.; HALEVY, Alon Y.; HELLERSTEIN, Joseph M.; IOANNIDIS, Yannis E.; JAGADISH, H. V.; KOSSMANN, Donald; MADDEN, Samuel; MEHROTRA, Sharad; MILO, Tova; NAUGHTON, Jeffrey F.; RAMAKRISHNAN, Raghu; MARKL, Volker; OLSTON, Christopher; OOI, Beng Chin; RÉ, Christopher; SUCIU, Dan; STONEBRAKER, Michael; WALTER, Todd; WIDOM, Jennifer. The beckman report on database research. **Commun. ACM**, v. 59, n. 2, p. 92–99, 2016. Available from Internet: <<<https://doi.org/10.1145/2845915>>>.
- ABADI, Daniel; AILAMAKI, Anastasia; ANDERSEN, David; BAILIS, Peter; BALAZINSKA, Magdalena; BERNSTEIN, Philip A.; BONCZ, Peter A.; CHAUDHURI, Surajit; CHEUNG, Alvin; DOAN, AnHai; DONG, Luna; FRANKLIN, Michael J.; FREIRE, Juliana; HALEVY, Alon Y.; HELLERSTEIN, Joseph M.; IDREOS, Stratos; KOSSMANN, Donald; KRASKA, Tim; KRISHNAMURTHY, Sailesh; MARKL, Volker; MELNIK, Sergey; MILO, Tova; MOHAN, C.; NEUMANN, Thomas; OOI, Beng Chin; OZCAN, Fatma; PATEL, Jignesh; PAVLO, Andrew; POPA, Raluca A.; RAMAKRISHNAN, Raghu; RÉ, Christopher; STONEBRAKER, Michael; SUCIU, Dan. The seattle report on database research. **SIGMOD Rec.**, v. 48, n. 4, p. 44–53, 2019. Available from Internet: <<<https://doi.org/10.1145/3385658.3385668>>>.
- ABBASI, Ahmed; SARKER, Suprateek; CHIANG, Roger HL. Big data research in information systems: Toward an inclusive research agenda. **Journal of the Association for Information Systems**, v. 17, n. 2, 2016.
- ALTINTAS, Ilkay; ANAND, Manish Kumar; CRAWL, Daniel; BOWERS, Shawn; BELLOUM, Adam; MISSIER, Paolo; LUDÄSCHER, Bertram; GOBLE, Carole A; SLOOT, Peter MA. Understanding collaborative studies through interoperable workflow provenance. In: SPRINGER. **IPAW**. [S.l.], 2010. p. 42–58.
- ALTINTAS, Ilkay; BERKLEY, Chad; JAEGER, Efrat; JONES, Matthew B.; LUDÄSCHER, Bertram; MOCK, Steve. Kepler: An extensible system for design and execution of scientific workflows. In: **Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM 2004), 21-23 June 2004, Santorini Island, Greece**. [s.n.], 2004. p. 423–424. Available from Internet: <<<https://doi.org/10.1109/SSDM.2004.1311241>>>.
- ANAND, Manish Kumar; BOWERS, Shawn; ALTINTAS, Ilkay; LUDÄSCHER, Bertram. Approaches for exploring and querying scientific workflow provenance graphs. In: SPRINGER. **International Provenance and Annotation Workshop**. [S.l.], 2010. p. 17–26.
- ANAND, Manish Kumar; BOWERS, Shawn; MCPHILLIPS, Timothy; LUDÄSCHER, Bertram. Exploring scientific workflow provenance using hybrid queries over nested data and lineage graphs. In: SPRINGER. **SSDBM**. [S.l.], 2009. p. 237–254.
- ATKINSON, Malcolm P.; GESING, Sandra; MONTAGNAT, Johan; TAYLOR, Ian J. Scientific workflows: Past, present and future. **FGCS**, v. 75, p. 216–227, 2017. Available from Internet: <<<https://doi.org/10.1016/j.future.2017.05.041>>>.

- BAVOIL, Louis; CALLAHAN, Steven P.; SCHEIDEGGER, Carlos Eduardo; VO, Huy T.; CROSSNO, Patricia; SILVA, Cláudio T.; FREIRE, Juliana. Vistrails: Enabling interactive multiple-view visualizations. In: **16th IEEE Visualization Conference, VIS 2005, Minneapolis, MN, USA, October 23-28, 2005**. IEEE Computer Society, 2005. p. 135–142. Available from Internet: <<<https://doi.org/10.1109/VISUAL.2005.1532788>>>.
- BEGOLI, Edmon; KISTLER, Derek; BATES, Jack. Towards a heterogeneous, polystore-like data architecture for the US department of veteran affairs (VA) enterprise analytics. In: **BigData 2016**. [s.n.], 2016. p. 2550–2554. Available from Internet: <<<https://doi.org/10.1109/BigData.2016.7840896>>>.
- BEVANA, Nigel; KIRAKOWSKIB, Jurek; MAISSELA, Jonathan. What is usability. In: CITESEER. **Proceedings of the 4th International Conference on HCI**. [S.l.], 1991.
- BOSE, Rajendra; FREW, James. Lineage retrieval for scientific data processing: a survey. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 37, n. 1, p. 1–28, 2005.
- BUDGEN, David; TURNER, Mark; BRERETON, Pearl; KITCHENHAM, Barbara A. Using mapping studies in software engineering. In: **PPIG**. [S.l.: s.n.], 2008. v. 8, p. 195–204.
- BUNEMAN, Peter; KHANNA, Sanjeev; WANG-CHIEW, Tan. Why and where: A characterization of data provenance. In: SPRINGER. **ICDT**. [S.l.], 2001. p. 316–330.
- CHEBOTKO, Artem; LU, Shiyong; FEI, Xubo; FOTOUHI, Farshad. Rdfprov: A relational rdf store for querying and managing scientific workflow provenance. **Data & Knowledge Engineering**, Elsevier, v. 69, n. 8, p. 836–865, 2010.
- CHIRIGATI, Fernando; FREIRE, Juliana. Provenance and reproducibility. In: LIU, Ling; ÖZSU, M. Tamer (Ed.). **Encyclopedia of Database Systems, Second Edition**. Springer, 2018. Available from Internet: <<https://doi.org/10.1007/978-1-4614-8265-9_80747>>.
- COSTA, Catarina; MURTA, Leonardo. Version control in distributed software development: A systematic mapping study. In: IEEE. **2013 ICGSE**. [S.l.], 2013. p. 90–99.
- COULOURIS, George F; DOLLIMORE, Jean; KINDBERG, Tim. **Distributed systems: concepts and design**. [S.l.]: pearson education, 2005.
- CUEVAS-VICENTTIN, Victor; DEY, Saumen; WANG, Michael Li Yuan; SONG, Tianhong; LUDASCHER, Bertram. Modeling and querying scientific workflow provenance in the d-opm. In: IEEE. **2012 SC Companion: High-Performance Computing, Networking, Storage and Analysis (SCC)**. [S.l.], 2012. p. 119–128.
- CUEVAS-VICENTTÍN, Víctor; LUDÄSCHER, B; MISSIER, P; BELHAJJAME, K; CHIRIGATI, F; WEI, Y; LEINFELDER, B. **Provone: A prov extension data model for scientific workflow provenance**. [S.l.]: Tech. Rep. DataOne, 2015.
- DAVIDSON, Susan B; FREIRE, Juliana. Provenance and scientific workflows: challenges and opportunities. In: ACM. **2008 ACM SIGMOD**. [S.l.], 2008. p. 1345–1350.

DEELMAN, Ewa; PETERKA, Tom; ALTINTAS, Ilkay; CAROTHERS, Christopher D.; DAM, Kerstin Kleese van; MORELAND, Kenneth; PARASHAR, Manish; RAMAKRISHNAN, Lavanya; TAUFER, Michela; VETTER, Jeffrey S. The future of scientific workflows. **Int. J. High Perform. Comput. Appl.**, v. 32, n. 1, p. 159–175, 2018. Available from Internet: <<<https://doi.org/10.1177/1094342017704893>>>.

DEELMAN, Ewa; VAHI, Karan; JUVE, Gideon; RYNGE, Mats; CALLAGHAN, Scott; MAECHLING, Philip; MAYANI, Rajiv; CHEN, Weiwei; SILVA, Rafael Ferreira da; LIVNY, Miron; WENGER, R. Kent. Pegasus, a workflow management system for science automation. **Future Generation Comp. Syst.**, v. 46, p. 17–35, 2015. Available from Internet: <<<https://doi.org/10.1016/j.future.2014.10.008>>>.

DING, Li; MICHAELIS, James; MCCUSKER, Jim; MCGUINNESS, Deborah L. Linked provenance data: A semantic web-based approach to interoperable workflow traces. **Future Generation Computer Systems**, Elsevier, v. 27, n. 6, p. 797–805, 2011.

DUGGAN, Jennie; ELMORE, Aaron J; STONEBRAKER, Michael; BALAZINSKA, Magda; HOWE, Bill; KEPNER, Jeremy; MADDEN, Sam; MAIER, David; MATTSON, Tim; ZDONIK, Stan. The bigdawg polystore system. **ACM Sigmod Record**, ACM, v. 44, n. 2, p. 11–16, 2015.

ELLQVIST, Tommy; KOOP, David; FREIRE, Juliana; SILVA, Cláudio; STRÖMBÄCK, Lena. Using mediation to achieve provenance interoperability. In: **IEEE Services-I, 2009 World Conference on**. [S.l.], 2009. p. 291–298.

FREIRE, Juliana; CHIRIGATI, Fernando Seabra. Provenance and the different flavors of reproducibility. **IEEE Data Eng. Bull.**, v. 41, n. 1, p. 15–26, 2018. Available from Internet: <<<http://sites.computer.org/debull/A18mar/p15.pdf>>>.

FREIRE, Juliana; KOOP, David; SANTOS, Emanuele; SILVA, Cláudio T. Provenance for computational tasks: A survey. **CS & E**, IEEE, v. 10, n. 3, 2008.

GADEPALLY, Vijay; CHEN, Peinan; DUGGAN, Jennie; ELMORE, Aaron; HAYNES, Brandon; KEPNER, Jeremy; MADDEN, Samuel; MATTSON, Tim; STONEBRAKER, Michael. The bigdawg polystore system and architecture. In: **IEEE High Performance Extreme Computing Conference (HPEC), 2016 IEEE**. [S.l.], 2016. p. 1–6.

GASPAR, Wander; BRAGA, Regina; CAMPOS, Fernanda. Sciprov: an architecture for semantic query in provenance metadata on e-science context. In: **SPRINGER International Conference on Information Technology in Bio-and Medical Informatics**. [S.l.], 2011. p. 68–81.

GESING, Sandra; DOOLEY, Rion; PIERCE, Marlon E.; KRÜGER, Jens; GRUNZKE, Richard; HERRES-PAWLIS, Sonja; HOFFMANN, Alexander. Gathering requirements for advancing simulations in HPC infrastructures via science gateways. **FGCS**, v. 82, p. 544–554, 2018. Available from Internet: <<<https://doi.org/10.1016/j.future.2017.02.042>>>.

GLATARD, Tristan; ROUSSEAU, Marc Étienne; CAMARASU-POP, Sorina; ADALAT, Reza; BECK, Natacha; DAS, Samir; SILVA, Rafael Ferreira da; KHALILI-MAHANI, Najmeh; KORKHOV, Vladimir; QUIRION, Pierre-Olivier; RIOUX, Pierre; OLABAR-RIAGA, Sílvia D.; BELLEC, Pierre; EVANS, Alan C. Software architectures to integrate workflow engines in science gateways. **Future Generation Computer Sys-**

tems, v. 75, p. 239 – 255, 2017. ISSN 0167-739X. Available from Internet: <<<http://www.sciencedirect.com/science/article/pii/S0167739X17300249>>>.

GROTH, Paul; MOREAU, Luc. Recording process documentation for provenance. **IEEE TPDS**, IEEE, v. 20, n. 9, p. 1246–1259, 2009.

HAMADOU, Hamdi Ben; GALLINUCCI, Enrico; GOLFARELLI, Matteo. Answering GPSJ queries in a polystore: A dataspace-based approach. In: LAENDER, Alberto H. F.; PERNICI, Barbara; LIM, Ee-Peng; OLIVEIRA, José Palazzo M. de (Ed.). **Conceptual Modeling - 38th International Conference, ER 2019, Salvador, Brazil, November 4-7, 2019, Proceedings**. Springer, 2019. (Lecture Notes in Computer Science, v. 11788), p. 189–203. Available from Internet: <<https://doi.org/10.1007/978-3-030-33223-5_16>>.

HAZEN, Benjamin T; BOONE, Christopher A; EZELL, Jeremy D; JONES-FARMER, L Allison. Data quality for data science, predictive analytics, and big data in supply chain management: An introduction to the problem and suggestions for research and applications. **International Journal of Production Economics**, Elsevier, v. 154, p. 72–80, 2014.

HEY, Tony; TANSLEY, Stewart; TOLLE, Kristin M et al. **The fourth paradigm: data-intensive scientific discovery**. [S.l.]: Microsoft research Redmond, WA, 2009.

HUYNH, Trung Dong; EBDEN, Mark; FISCHER, Joel E.; ROBERTS, Stephen J.; MOREAU, Luc. Provenance network analytics - an approach to data analytics using data provenance. **Data Min. Knowl. Discov.**, v. 32, n. 3, p. 708–735, 2018. Available from Internet: <<<https://doi.org/10.1007/s10618-017-0549-3>>>.

JABAL, Amani Abu; BERTINO, Elisa. Simp: Secure interoperable multi-granular provenance framework. In: IEEE. **IEEE e-Science 2016**. [S.l.], 2016. p. 270–275.

JAGADISH, H. V.; GEHRKE, Johannes; LABRINIDIS, Alexandros; PAPAKONSTANTINO, Yannis; PATEL, Jignesh M.; RAMAKRISHNAN, Raghu; SHAHABI, Cyrus. Big data and its technical challenges. **Commun. ACM**, v. 57, n. 7, p. 86–94, 2014. Available from Internet: <<<https://doi.org/10.1145/2611567>>>.

JHINGRAN, AD; MATTOS, Nelson; PIRAHESH, Hamid. Information integration: A research agenda. **IBM systems Journal**, IBM, v. 41, n. 4, p. 555–562, 2002.

KHAN, Farah Zaib; SOILAND-REYES, Stian; SINNOTT, Richard O; LONIE, Andrew; GOBLE, Carole; CRUSOE, Michael R. Sharing interoperable workflow provenance: A review of best practices and their practical application in cwlprov. **GigaScience**, Oxford University Press, v. 8, n. 11, p. giz095, 2019.

KHAN, Yasar; ZIMMERMANN, Antoine; JHA, Alok Kumar; GADEPALLY, Vijay; D'AQUIN, Mathieu; SAHAY, Ratnesh. One size does not fit all: Querying web polystores. **IEEE Access**, v. 7, p. 9598–9617, 2019. Available from Internet: <<<https://doi.org/10.1109/ACCESS.2018.2888601>>>.

KITCHENHAM, Barbara. Procedures for performing systematic reviews. **Keele, UK, Keele University**, v. 33, n. 2004, p. 1–26, 2004.

LI, Chunqiu; SUGIMOTO, Shigeo. Provenance description of metadata using prov with premis for long-term use of metadata. In: **International Conference on Dublin Core and Metadata Applications**. [S.l.: s.n.], 2014. p. 147–156.

LIM, Chunhyeok; LU, Shiyong; CHEBOTKO, Artem; FOTOUHI, Farshad. Storing, reasoning, and querying opm-compliant scientific workflow provenance using relational databases. **FGCS**, Elsevier, v. 27, n. 6, p. 781–789, 2011.

LITWIN, Witold; ABDELLATIF, Abdelaziz. Multidatabase interoperability. **Computer**, IEEE, n. 12, p. 10–18, 1986.

MATTOSO, Marta; WERNER, Cláudia; TRAVASSOS, Guilherme Horta; BRAGANHOLO, Vanessa; OGASAWARA, Eduardo S.; OLIVEIRA, Daniel de; CRUZ, Sérgio Manuel Serra da; MARTINHO, Wallace; MURTA, Leonardo. Towards supporting the life cycle of large scale scientific experiments. **IJBPI**, v. 5, n. 1, p. 79–92, 2010. Available from Internet: <<<https://doi.org/10.1504/IJBPI.2010.033176>>>.

MISSIER, Paolo; LUDÄSCHER, Bertram; BOWERS, Shawn; DEY, Saumen; SARKAR, Anandarup; SHRESTHA, Biva; ALTINTAS, Ilkay; ANAND, Manish Kumar; GOBLE, Carole. Linking multiple workflow provenance traces for interoperable collaborative science. In: IEEE. **WORKS 2010**. [S.l.], 2010. p. 1–8.

MONDELLI, Maria Luiza; MAGALHÃES, Thiago; LOSS, Guilherme; WILDE, Michael; FOSTER, Ian T.; MATTOSO, Marta; KATZ, Daniel S.; BARBOSA, Helio J. C.; VASCONCELOS, Ana Tereza Ribeiro de; OCAÑA, Kary A. C. S.; JR., Luiz M. R. Gadelha. Bioworkbench: A high-performance framework for managing and analyzing bioinformatics experiments. **CoRR**, abs/1801.03915, 2018. Available from Internet: <<<http://arxiv.org/abs/1801.03915>>>.

MOREAU, Luc; FREIRE, Juliana; FUTRELLE, Joe; MCGRATH, Robert E; MYERS, Jim; PAULSON, Patrick. The open provenance model: An overview. In: SPRINGER. **International Provenance and Annotation Workshop**. [S.l.], 2008. p. 323–326.

MOREAU, Luc; GROTH, Paul T.; CHENEY, James; LEBO, Timothy; MILES, Simon. The rationale of PROV. **J. Web Semant.**, v. 35, p. 235–257, 2015. Available from Internet: <<<https://doi.org/10.1016/j.websem.2015.04.001>>>.

OCAÑA, Kary ACS; OLIVEIRA, Daniel de; OGASAWARA, Eduardo; DÁVILA, Alberto MR; LIMA, Alexandre AB; MATTOSO, Marta. Sciphy: a cloud-based workflow for phylogenetic analysis of drug targets in protozoan genomes. In: SPRINGER. **BSB11**. [S.l.], 2011. p. 66–70.

OGASAWARA, Eduardo S.; DIAS, Jonas; SOUSA, Vítor Silva; CHIRIGATI, Fernando Seabra; OLIVEIRA, Daniel de; PORTO, Fábio; VALDURIEZ, Patrick; MATTOSO, Marta. Chiron: a parallel engine for algebraic scientific workflows. **Concurr. Comput. Pract. Exp.**, v. 25, n. 16, p. 2327–2341, 2013. Available from Internet: <<<https://doi.org/10.1002/cpe.3032>>>.

OLIVEIRA, Ary Henrique M. de; OLIVEIRA, Daniel de; MATTOSO, Marta. Clouds and reproducibility: A way to go to scientific experiments? In: ANTONOPOULOS, Nick; GILLAM, Lee (Ed.). **Cloud Computing - Principles, Systems and Applications, Second Edition**. Springer, 2017, (Computer Communications and Networks). p. 127–151. Available from Internet: <<https://doi.org/10.1007/978-3-319-54645-2_5>>.

OLIVEIRA, Daniel de; LIU, Ji; PACITTI, Esther. **Data-Intensive Workflow Management: For Clouds and Data-Intensive and Scalable Computing Environments**. Morgan & Claypool Publishers, 2019. (Synthesis Lectures on Data Management). Available from Internet: <<<https://www.morganclaypool.com/doi/10.2200/S00915ED1V01Y201904DTM060>>>.

OLIVEIRA, Daniel de; OCAÑA, Kary A. C. S.; BAIÃO, Fernanda Araujo; MATTOSO, Marta. A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds. **J. Grid Comput.**, v. 10, n. 3, p. 521–552, 2012. Available from Internet: <<<https://doi.org/10.1007/s10723-012-9227-2>>>.

OLIVEIRA, Daniel de; OCAÑA, Kary A. C. S.; OGASAWARA, Eduardo S.; DIAS, Jonas; GONÇALVES, João Carlos de A. R.; BAIÃO, Fernanda Araujo; MATTOSO, Marta. Performance evaluation of parallel strategies in public clouds: A study with phylogenomic workflows. **Future Gener. Comput. Syst.**, v. 29, n. 7, p. 1816–1825, 2013. Available from Internet: <<<https://doi.org/10.1016/j.future.2012.12.019>>>.

OLIVEIRA, Daniel de; OGASAWARA, Eduardo S.; BAIÃO, Fernanda Araujo; MATTOSO, Marta. Scicumulus: A lightweight cloud middleware to explore many task computing paradigm in scientific workflows. In: **IEEE International Conference on Cloud Computing, CLOUD 2010, Miami, FL, USA, 5-10 July, 2010**. [s.n.], 2010. p. 378–385. Available from Internet: <<<https://doi.org/10.1109/CLOUD.2010.64>>>.

OLIVEIRA, Wellington; MISSIER, Paolo; OCAÑA, Kary; OLIVEIRA, Daniel de; BRAGANHOLO, Vanessa. Analyzing provenance across heterogeneous provenance graphs. In: SPRINGER. **IPAW**. [S.l.], 2016. p. 57–70.

ÖZSU, M Tamer; VALDURIEZ, Patrick. **Principles of distributed database systems**. [S.l.]: Springer Science & Business Media, 2011.

PARCIAK, M; BAUER, C; BENDER, T; LODAHL, R; SCHREIWEIS, B; TUTE, E; SAX, U. Provenance solutions for medical research in heterogeneous it-infrastructure: An implementation roadmap. **Studies in health technology and informatics**, v. 264, p. 298–302, 2019.

PENG, Roger. The reproducibility crisis in science: A statistical counterattack. **Significance**, Wiley Online Library, v. 12, n. 3, p. 30–32, 2015.

PÉREZ, Beatriz; RUBIO, Julio; SÁENZ-ADÁN, Carlos. A systematic review of provenance systems. **Knowledge and Information Systems**, Springer, p. 1–49, 2018.

PIMENTEL, João Felipe; FREIRE, Juliana; BRAGANHOLO, Vanessa; MURTA, Leonardo. Tracking and analyzing the evolution of provenance from scripts. In: SPRINGER. **International Provenance and Annotation Workshop**. [S.l.], 2016. p. 16–28.

PRABHUNE, Ajinkya; STOTZKA, Rainer; SAKHARKAR, Vaibhav; HESSER, Jürgen; GERTZ, Michael. Metastore: an adaptive metadata management framework for heterogeneous metadata models. **DPD**, Springer, v. 36, n. 1, p. 153–194, 2018.

PRABHUNE, Ajinkya; ZWEIG, Aaron; STOTZKA, Rainer; GERTZ, Michael; HESSER, Juergen. Prov2one: an algorithm for automatically constructing provone provenance graphs. In: SPRINGER. **IPAW**. [S.l.], 2016. p. 204–208.

PRABHUNE, Ajinkya; ZWEIG, Aaron; STOTZKA, Rainer; HESSER, Jürgen; GERTZ, Michael. P-PIF: a provone provenance interoperability framework for analyzing heterogeneous workflow specifications and provenance traces. **Distributed and Parallel Databases**, v. 36, n. 1, p. 219–264, 2018. Available from Internet: <<<https://doi.org/10.1007/s10619-017-7216-y>>>.

PRABHUNE, Ajinkya; ZWEIG, Aaron; STOTZKA, Rainer; HESSER, Jürgen; GERTZ, Michael. P-pif: a provone provenance interoperability framework for analyzing heterogeneous workflow specifications and provenance traces. **DPD**, Springer, v. 36, n. 1, p. 219–264, 2018.

PROV-OVERVIEW. 2013. <<https://www.w3.org/TR/prov-overview/>>. [Online; accessed 24-August-2018].

SCHWAB, Matthias; KARRENBACH, N; CLAERBOUT, Jon. Making scientific computations reproducible. **CS & E, IEEE**, v. 2, n. 6, p. 61–67, 2000.

SIMMHAN, Yogesh L; PLALE, Beth; GANNON, Dennis. A survey of data provenance in e-science. **ACM Sigmod Record**, ACM, v. 34, n. 3, p. 31–36, 2005.

SOUZA, Renan; AZEVEDO, Leonardo; THIAGO, Raphael; SOARES, Elton; NERY, Marcelo; NETTO, Marco; BRAZIL, Emilio Vital; CERQUEIRA, Renato; VALDURIEZ, Patrick; MATTOSO, Marta. Efficient runtime capture of multiworkflow data using provenance. In: . [S.l.: s.n.], 2019.

STEINMACHER, Igor; CHAVES, Ana Paula; GEROSA, Marco Aurélio. Awareness support in distributed software development: A systematic review and mapping of the literature. **CSCW**, Springer, v. 22, n. 2-3, p. 113–158, 2013.

STONEBRAKER, Michael. The case for polystores. **ACM SIGMOD Blog**, 2015.

THE third provenance challenge. 2009. <<https://openprovenance.org/provenance-challenge/ThirdProvenanceChallenge.html>>. [Online; accessed 24-August-2018].

TOLK, Andreas; MUGUIRA, James A. The levels of conceptual interoperability model. In: CITESEER. **Proceedings of the 2003 fall simulation interoperability workshop**. [S.l.], 2003. v. 7, p. 1–11.

VÍCTOR LUDÄSCHER BERTRAM, Missier Paolo Belhajjame Khalid Chirigati Fernando Wei Yaxing Dey Saumen Kianmajd Parisa Koop David Bowers Shawn Altintas Ilkay Jones Christopher B. Jones Matthew Walker Lauren Slaughter Peter Leinfelder Ben Cao Yang Cuevas-Vicenttín. **ProvONE Data Model**. 2016. <<http://jenkins-1.dataone.org/jenkins/view/Documentation%20Projects/job/ProvONE-Documentation-trunk/ws/provenance/ProvONE/v1/provone.html>>. [Online; accessed 19-July-2020].

WATSON, Paul; HIDEN, Hugo; WOODMAN, Simon. e-science central for CARMEN: science as a service. **Concurrency and Computation: Practice and Experience**, v. 22, n. 17, p. 2369–2380, 2010. Available from Internet: <<<https://doi.org/10.1002/cpe.1611>>>.

WEGNER, Peter. Interoperability. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 28, n. 1, p. 285–287, 1996.

WIEDERHOLD, Gio. Mediators in the architecture of future information systems. **Computer**, IEEE, v. 25, n. 3, p. 38–49, 1992.

WOHLIN, Claes. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: **ACM. Proceedings of the 18th international conference on evaluation and assessment in software engineering**. [S.l.], 2014. p. 38.

WOLSTENCROFT, Katherine; HAINES, Robert; FELLOWS, Donal; WILLIAMS, Alan R.; WITHERS, David; OWEN, Stuart; SOILAND-REYES, Stian; DUNLOP, Ian; NENADIC, Aleksandra; FISHER, Paul; BHAGAT, Jiten; BELHAJJAME, Khalid; BACALL, Finn; HARDISTY, Alex; HIDALGA, Abraham Nieva de la; VARGAS, Maria P. Balcazar; SUFI, Shoaib; GOBLE, Carole A. The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. **Nucleic Acids Research**, v. 41, n. Webserver-Issue, p. 557–561, 2013. Available from Internet: <<<https://doi.org/10.1093/nar/gkt328>>>.

WOZNIAK, Justin M.; ARMSTRONG, Timothy G.; WILDE, Michael; KATZ, Daniel S.; LUSK, Ewing L.; FOSTER, Ian T. Swift/t: Large-scale application composition via distributed-memory dataflow processing. In: **13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2013, Delft, Netherlands, May 13-16, 2013**. [s.n.], 2013. p. 95–102. Available from Internet: <<<https://doi.org/10.1109/CCGrid.2013.99>>>.

YU, Jia; BUYYA, Rajkumar. A taxonomy of scientific workflow systems for grid computing. **ACM Sigmod Record**, ACM, v. 34, n. 3, p. 44–49, 2005.