Polyflow: a Polystore-compliant Mechanism to Provide Interoperability to Heterogeneous Provenance Graphs[1]

Yan Mendes[1], Daniel de Oliveira[2], Victor Ströele[1]

[1] Federal University of Juiz de Fora, Juiz de Fora, Minas Gerais, Brazil
[yanmendes,victor.stroele]@ice.ufjf.br
[2] Fluminense Federal University, Niterói, Rio de Janeiro, Brazil
danielcmo@ic.uff.br

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.

Categories and Subject Descriptors: H.2.1 [**Database Management**]: Logical Design; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords: Polystore, Syntactic interoperability, Semantic interoperability

1. INTRODUCTION

Over the last decade the (big) data-driven science paradigm became a reality [Hey et al. 2009; de Oliveira et al. 2019]. As discussed by [Abbasi et al. 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, to adhere to this paradigm, some challenges must be overcome, such as the dependency on data quality [Hazen et al. 2014] and the lack of reproducibility of the results [Peng 2015; Schwab et al. 2000; Freire and Chirigati 2018; Chirigati and Freire 2018]. In order to foster reproducibility [de Oliveira et al. 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

[1]This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. This study was also financed in part by CNPq, FAPERJ and UFJF. The authors would like to thank Kary Ocaña (LNCC) for her support on the experiments.

of an experiment, and *Retrospective* (or simply *r-prov*), which is associated to the execution of an experiment. Since provenance represents generated that and the processes that generated them, it 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 et al. 2010], VisTrails [Bavoil et al. 2005], and SciCumulus [de Oliveira et al. 2010; de 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, *i.e.*, it is not possible to query all provenance databases in an integrated form since most solutions adopted proprietary data models. Thus, inter-operating provenance data captured by underlying WfMSs remains an issue [Oliveira et al. 2016].

This way, while in theory it should be possible for scientists to transparently query both provenance databases, the heterogeneity in the data models and implementations makes it difficult to query them in an integrated way, *i.e.*, one has to be aware of both database schemas and the association between them. 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 and Abdellatif 1986].

Recently, new approaches such as *Polystore systems* [Begoli et al. 2016; Hamadou et al. 2019; Khan et al. 2019] 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 article, we propose `Polyflow`, a Polystore-compliant mechanism that provides semantic interoperability of heterogeneous provenance graphs. We assume that all data models are from the same domain (*i.e.*, provenance), thus they follow 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 et al. 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 schemas. By using Polyflow, we support distributed environments, empowering geographically scattered researchers, since different researchers commonly use different WfMSs (and consequently different provenance databases).

The contributions of this article are summarized as follows. First, to propose a viable solution to solve the syntactic and semantic interoperability in provenance databases using a polystore approach. Second, we provide an overview of the research topic, comparing state-of-the-art approaches to Polyflow using a Systematic Literature Mapping (SLM). And finally, we have conducted a range of experiments to evaluate Polyflow with experts. This article is an extension of the conference paper [Mendes et al. 2019] published in the Proceedings of the 2019 Brazilian Symposium on Databases (SBBD). This extended version provides new empirical shreds of evidence regarding the proposed approach, an evaluation of the approach with experts and a broader discussion on related work.

The remainder of this article is organized as follows. Section 2 briefly provides the background knowledge that supports this article; Section 3 showcases related work, granting an overview of the research topic through a SLM; Section 4 details the proposed approach; Section 5 showcases two evaluations of Polyflow, and, finally, Section 6 concludes this article.

2. BACKGROUND KNOWLEDGE

In this section, we introduce the main concepts and techniques adopted in this article. 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 [de Oliveira et al. 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 [Mattoso 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 and Buyya 2005]. In other words, it can be seen as the formalization of a pipeline of computational tasks with their respective inputs and outputs. A workflow can be modeled as a graph $W(A, \phi)$, where A is the set of activities in W and ϕ is the set of data dependencies. Thus, $A = \{a_1, a_2, \ldots, a_n\}$ 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, $I = \{i_1, i_2, \ldots, i_d\}$, where i_d is an input file for activity a_i, $O = \{o_1, o_2, \ldots, o_k\}$, where each o_k is an output file for a_i and $P = \{p_1, p_2, \ldots, p_m\}$, where each p_m is a parameter of activity a_i. Each execution of a_i is associated to a tuple of m parameters $< p_1, p_2, \ldots, p_m >$, 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 $\phi = \{\varphi_{1,2}, \ldots, \varphi_{i,j}\}$, where each $\varphi_{i,j} = \langle i_d, a_i, a_j \rangle$, $input(a_i) \in I$, $i_d \neq \varnothing$ 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 et al. 2001]. These metadata bring transparency to a data product, enabling its reuse [Simmhan et al. 2005]. Moreover, it also helps data interpretability and audition [Groth and Moreau 2009]. There are two types of provenance: prospective (*p-prov*) and retrospective (*r-prov*) [Davidson and 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]. It aims to capture the evolution of the workflow definition throughout its versions [Cuevas-Vicenttín et al. 2015].

In the context of this article, 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. Due to space restrictions, only the main classes are detailed in this section. You can refer for more information in [Prabhune et al. 2016]. ProvONE's data model is represented in Figure 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).

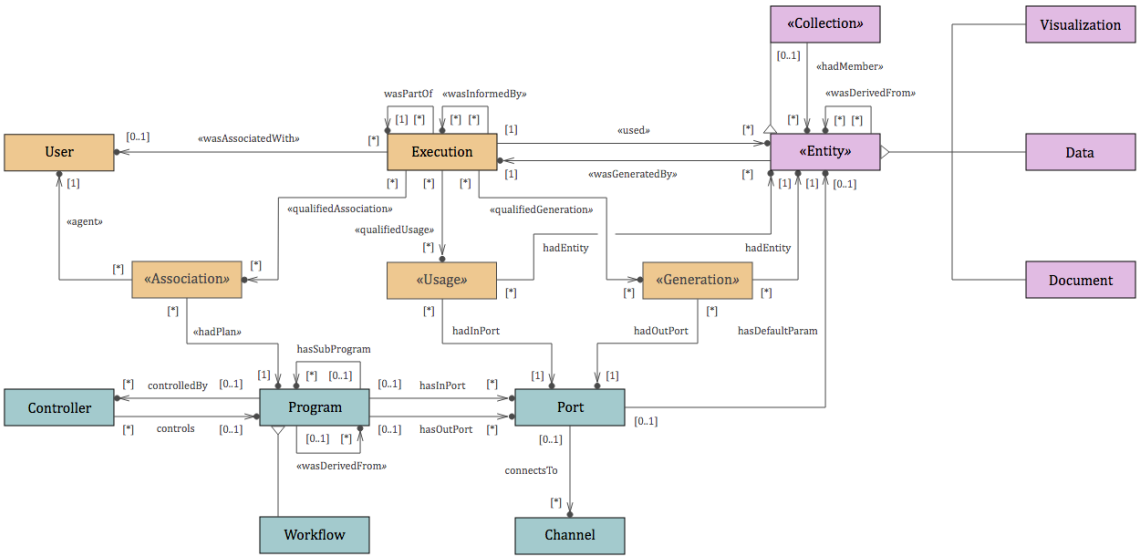

Fig. 1. ProvONE model adapted from [Cuevas-Vicenttín Víctor 2016]

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 and Muguira 2003], it represents what is known as *syntactic interoperability*. However, syntactic interoperability is just one existing type of interoperability. In this article'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 Section 1, the approach proposed in this article 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, 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]. BigDAWG[2] [Gadepally et al. 2016] is a pioneer Polystore system and is the engine responsible for the syntactic interoperability layer of this article. BigDAWG's architecture is presented in Fig. 2.

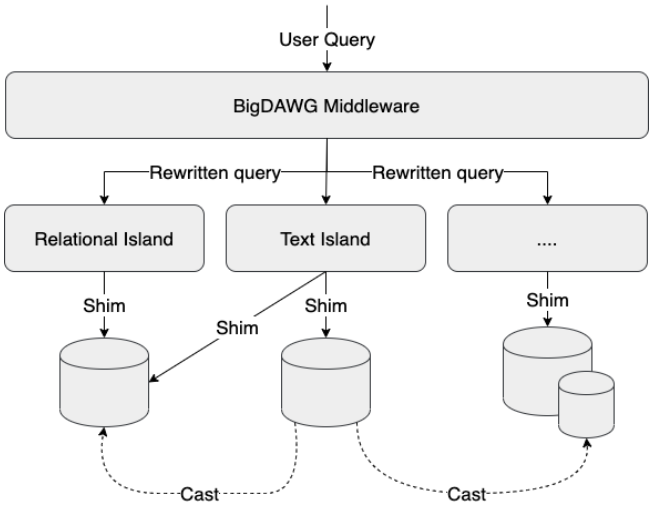

Fig. 2. BigDAWG's architectural design

The *middleware* layer is responsible for orchestrating incoming user queries, plan its submission to the underlying islands (by rewriting queries), and integrate intermediate responses. An *island* is the definition of a *data model* and a *query language* that represents a *data type*. 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 storage solutions.

As discussed by [Tolk and Muguira 2003], interoperability goes beyond the implementation 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 article, semantic interoperability is the difference between the data models that describe provenance data

[2]https://bigdawg.mit.edu

captured and stored by WfMSs. It comes down to the database integration problem with a bottom-up approach [Coulouris et al. 2005], *i.e.*, seamlessly integrating several different databases into one. The first step of the process proposed by [Coulouris et al. 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 et al. 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.

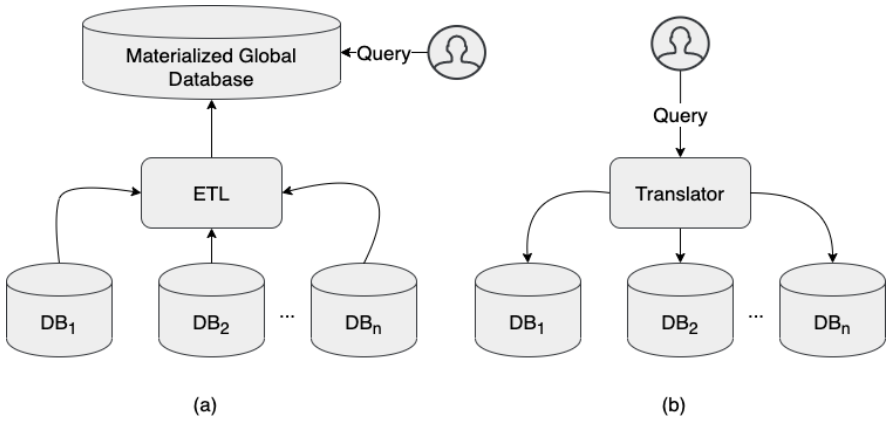

Fig. 3. (a) Physical approach to database integration. (b) Logical approach to database integration. Both figures adapted from [Coulouris et al. 2005]

3. RELATED WORK

In this section, 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 May, 2020.

3.1 Systematic Literature Mapping

The goal of the SLM is to identify and discuss interoperability solutions for heterogeneous provenance data, assessing them in a qualitative manner 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 et al. 2018], there are numerous surveys regarding provenance and WfMSs (*e.g.*, [Simmhan et al. 2005; Davidson and Freire 2008; Bose and 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]. Sub-section 3.1.1 discusses the planning of this SLM; Sub-section 3.1.2 presents the execution procedure; Sub-section 3.1.3 reports the findings; showcasing and discussing related literature. Finally, Sub-Section 3.3.1 exposes threats to this study's validity.

3.1.1 *Planning.* During the planning process, we identified the goals and defined a protocol, following the guidelines defined in [Kitchenham 2004]. The protocol specifies the method to be used in the SLM in order to reduce researcher bias [Steinmacher et al. 2013]. Moreover, a 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. In the context of this article, we follow the definition of interoperability proposed by [Asuncion and van Sinderen 2011], that is "the ability of different systems to use each other's services effectively". More specifically, 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 the scope of this article 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 terms used in the PICOC may have multiple synonyms, we have to define keywords and synonyms (Table I) that will compose the search string. The second step is to define questions 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 I. 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 in a more objective fashion, 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 take into account the used model - if it is already defined in the literature or if it is a novel approach. All aforementioned aspects 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)$\square$ - The solution does not satisfy the assessed metric.

Due to the huge number of papers returned during the search process, we had to define a *Filtering Process* to guarantee 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 workflow (*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 at two distinct occasions.

The second filtering was performed by assessing the title of the paper, its abstract and, when available, its introduction and conclusion. 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 Library[3], (ii) El Compendex[4], (iii) IEEE Digital Library[5], (iv) Scopus[6], and (v) Springer Link[7]. 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. In a iterative fashion, the knowledge bases were queried and some papers were read, enriching the Keywords and synonyms (Table I). 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")*

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

[3] `http://portal.acm.org`
[4] `http://www.engineeringvillage.com`
[5] `http://ieeexplore.ieee.org`
[6] `http://www.scopus.com`
[7] `http://link.springer.com`

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 (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 Parsifal[8]. Table II 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.

Table II. SLM filtering processes and the results of each step.

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

3.1.3 *Results.* In this Subsection, we briefly discuss the proposed solutions and assess them under the qualitative attributes defined in Subsection 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

[8] `http://parsif.al`

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 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 OPMProv. To evaluate it, the authors use the relational model to answer a set of queries defined in the Third Provenance Challenge (PC3) [thi 2009], an effort to promote studies regarding WfMSs interoperability. [Gaspar et al. 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 a 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 [pro 2013]. The solution supports RDBMS and a graph store (Neo4j[9]) 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 and Sugimoto 2014], storing data in a RDF solution (Apache Jena TDB [10]). 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.

[9] https://neo4j.com/
[10] https://jena.apache.org/documentation/tdb

3.2 *Ad-Hoc* Complementary Literature Review

Since some time has elapsed since the SLM execution and elaborating this article, 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 1.5 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 use PROV as their *lingua franca* to capture provenance from the external workflows. [Khan et al. 2019] 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 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 Figure 4. Figure 5 illustrates vehicle types in which the papers have been published and Figure 6 shows the number of citations of each paper. Following we discuss each of the questions.

(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 III. 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 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.

(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 IV. 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.

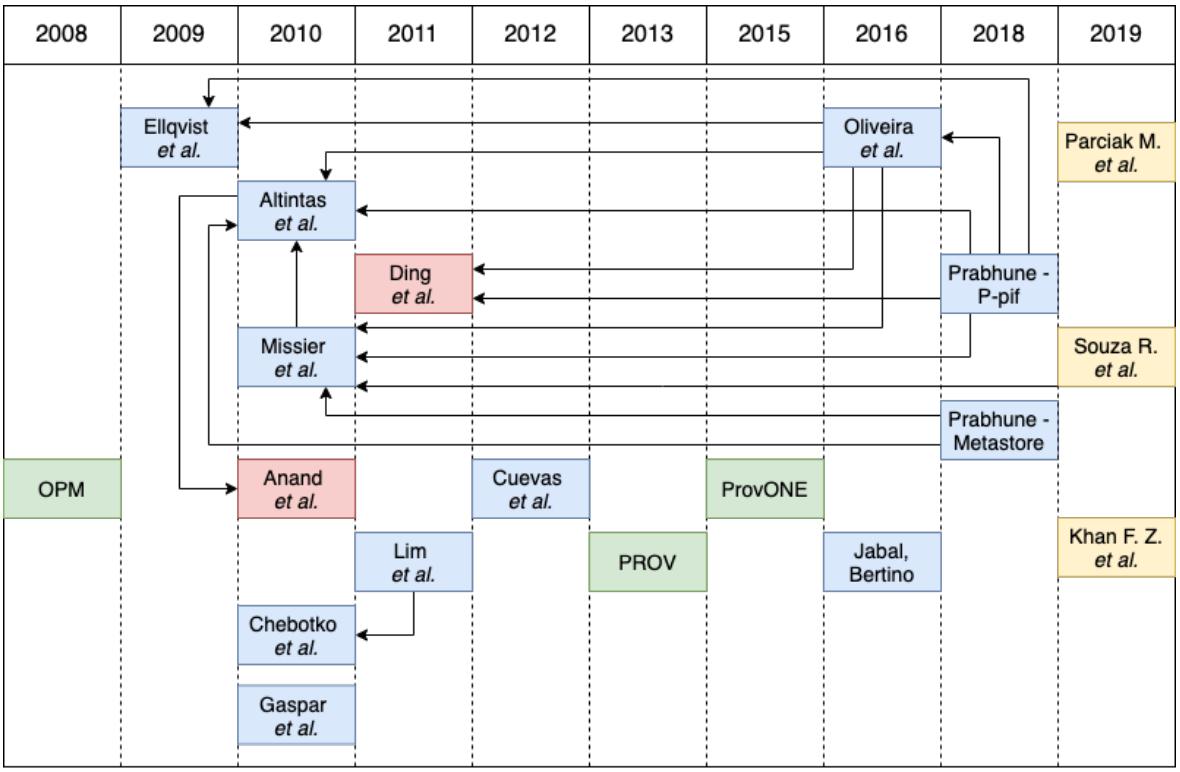

Fig. 4. Timeline of publications. Edges connecting two papers represent a citation. Works in red are the product of snowballing; the ones in blue were were found in the knowledge bases and the yellow ones were derived from the ad-hoc complimentary search.

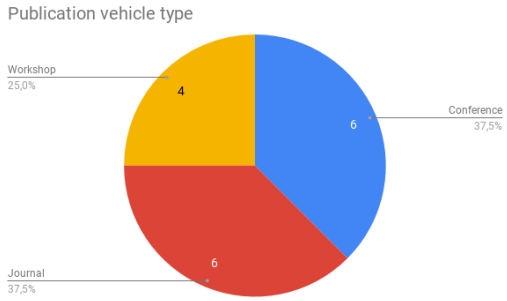

Fig. 5. Vehicle type in which the papers were published.

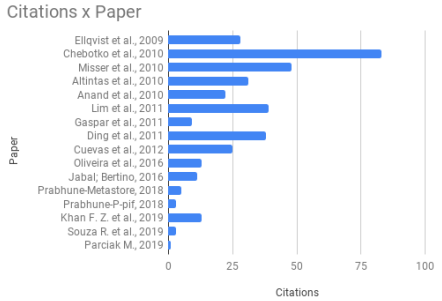

Fig. 6. Number of citations each paper has on July, 2020.

(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 V. 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 V is empty.

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

Table III. 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			⊇		≡	≡	≡	⊇	⊇	≡		≡	≡	≡	≡	≡

Table IV. 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											≡			≡		

Table V. 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	≡			≡							≡					

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.

3.3.1 *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. Finally, only one researcher conducted the filtering process. The bias would have been reduced if this process was done by a group, being the final identified threat.

4. POLYFLOW: INTEGRATING HETEROGENEOUS PROVENANCE GRAPHS

Polyflow's approach 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 article, it is important to formalize the concept of *Provenance Database*. A provenance database $\Im$ can be defined as a set of θ provenance graphs G_p, where $\Im = \{G_{p1}, G_{p2}, \ldots, G_{p\theta}\}$. For each $G_{p\theta} \in \Im$, 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 $Q_M(S, \Im) \Leftrightarrow \{G_{p\theta} \in \Im | \exists p_m \in G_{p\theta} \wedge \lambda(\star, \zeta_m(G_{p\theta}.p_m), v_m) \wedge (p_m, v_m) \in S\}$, where:

(i) Q_M is a set of pairs $S = \{(p_1, v_1), (p_2, v_2), \ldots, (p_m, v_m)\}$, 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 $G_{p\theta}$ through $\zeta_m(G_{p\theta.p_m)}$, using any relational algebra operator $\star$ (*e.g.*, σ, Π, $\bowtie$, *etc.*).

Consider two provenance graphs $G_{p1}, G_{p2} \in \Im$ that present semantically equivalent parameters p_m and p'_m ($p_m \equiv p'_m$), but with different names. Thus, given a set $P_\alpha = \{p_1, p_2, \ldots, p_\mu\} \in G_{p1}$ and $P_\beta = \{p_1, p_2, \ldots, p_\nu\} \in G_{p2}$ and a transformation language Υ, we should be able to find a mapping $\upsilon \in \Upsilon$ 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 $Q_M(\Gamma, \Im)$ (where $\Gamma = S_\alpha \cup S_\beta$, $S_\alpha = \{(p_1, v_1), (p_2, v_2), \ldots, (p_\mu, v_\mu)\}$ e $S_\beta = \{(p_1, v_1), (p_2, v_2), \ldots, (p_\nu, v_\nu)\}$), 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. The mapping between Global and source schemas can be classified as Global As View (GAV), Local As View (LAV), and Global and Local As View (GLAV). In LAV, sources are defined according to the global schema, *i.e.*, given the global schema, the sources are modeled based on it. On the other hand, in GAV approaches, the global schema is defined in terms of the source, representing many data sources. Finally, GLAV is a generalization of the two.

This article proposes a GAV approach to query heterogeneous provenance graphs in an integrated manner. The adopted strategy is based on *mediation* [Özsu and 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 a global (canonical) model. Thus, the strategy adopted in this article is to map schemas from different sources to the ProvONE model. It is worth noticing that the proposed mapping represents equivalences between entities of the global schema (ProvONE) and the local schemas (one for each SWfMS).

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) $1-N$, where $N \geq 2$: when the 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 multiple (*i.e.*, N) $1-1$ and $1-M$ mappings, respectively. At this time, just supports equivalence relations, but we want to expand this capability in future work. Figure 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 subsection). Figure 7(b) shows the Mapper of the APP_EXEC entity (Swift/T) to ProvONE's *Execution*.

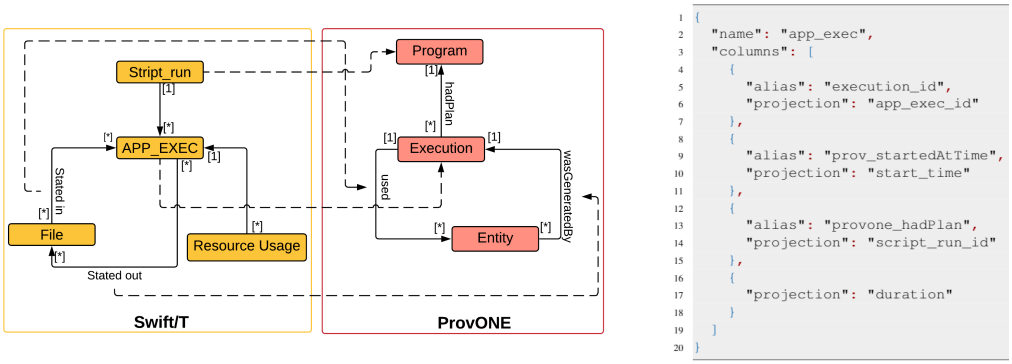

Fig. 7. (a) Mapping Swift/T schema to ProvONE; (b) *Mapper* of the *APP_EXEC* entity to *Execution* entity.

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 Figure 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 article are available at `https://github.com/UFFeScience/Polyflow`.

The *Data Source* layer represents the data sources that contain the multiple provenance databases $\Im_1, \Im_2, \ldots, \Im_f$, each of which is represented in different formats. All provenance data is collected by

WfMS during (or after) a wokflow execution (step [1] in Figure 8) and the stored in a provenance database (step [2] in Figure 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 Unique Resource Identifier (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 ability to fully express mappings with the structure we designed. On top of that, it's a popular and simple data format, mitigating the learning curve that other more complex formats (*e.g.,* RDF) could impose.

The *Query Processing* layer is responsible for receiving a query from the user and, based on the *Entity Mappers*, perform a query expansion (step [5] in Figure 8). The main component of this layer is the *Query Resolver* which is responsible for expanding queries based on a Global Schema (*i.e.*, ProvONE) into valid queries in the Local Schemas (steps [6] and [7] in Figure 8). The *Query Resolver* works similarly to the *shim* operator of the PolyStore architecture. Since `Polyflow` only supports equivalences between entities of the models, the *Query Resolver* accesses all mappings defined in a JSON format, and expands the ProvONE-based query by replacing ProvOne entities by the entity (or entities) of the underlying provenance schema using the *Entity Mappers*.

`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 Figure 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]*. All valid SQL constructs are supported by `Polyflow`. Queries are expanded at runtime by replacing mediators referenced in the query with a subquery using the elements present in the associated *Entity Mapper* (explained following). For example, let us assume that a given user needs to query all the executions of workflows that were performed in Swift/T WfMS. When querying for all *Executions* (*provone_ execution*), `Polyflow` performs the following query expansion (step [5] in Figure 8):

```
1 SELECT * FROM swift[provone_execution];
2 <=>
3 SELECT * FROM (
4     SELECT app_exec_id AS execution_id,
5     start_time AS prov_startedAtTime,
6     script_run_id AS provone_hadPlan, duration
```

```
7       FROM app_exec
8   ) AS table_0;
```

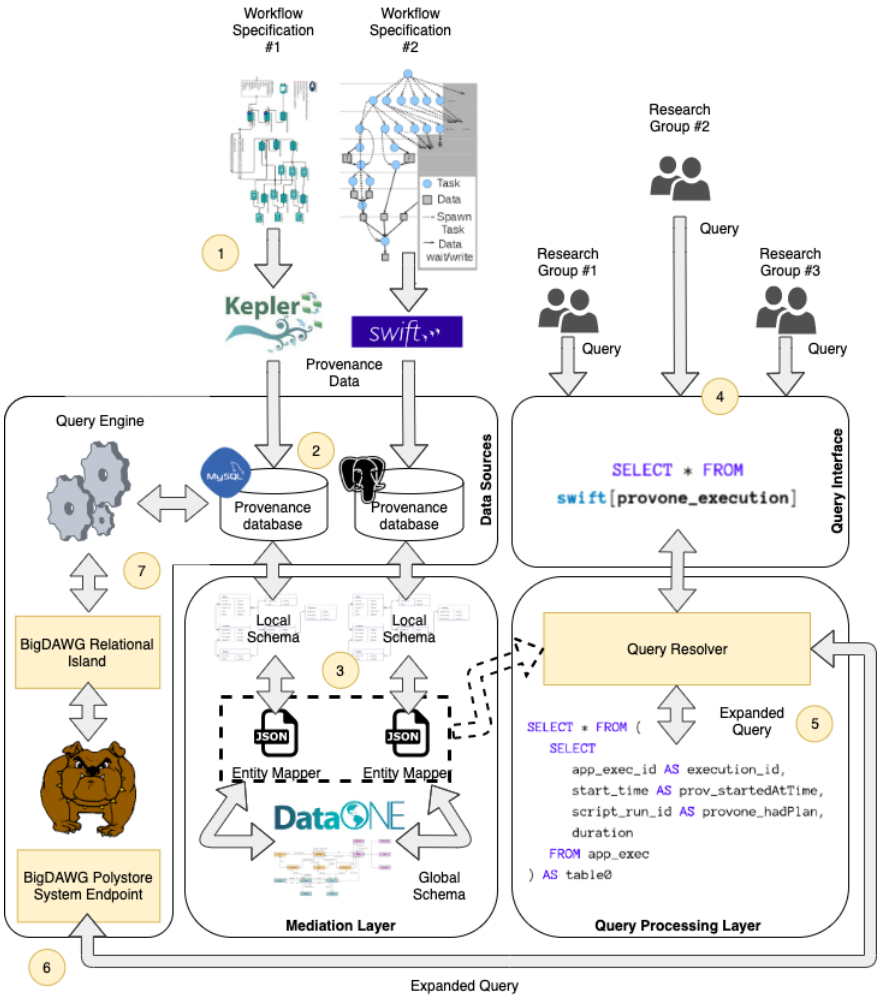

Fig. 8. `Polyflow` architecture.

4.4 `Polyflow` In Action

In this subsection we briefly expand formalism described in this section into usage scenarios to illustrate the proposed approach more clearly. `Polyflow` is implemented in Node.js due to its capability of dealing with I/O and HTTP requests asynchronously. This feature enables a looser coupling between `Polyflow` and the underlying data sources (*i.e.*, provenance databases), making it possible to return partial results as queries finish processing. We have 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. The main feature that compelled us to make this choice is its self-documenting capabilities, making consumption and discoverability of endpoints easier.

`Polyflow` has an internal catalog powered by a RDBMS where *Data Sources*, *Mediators* and *Entity Mappers* are stored. Its schema is illustrated in Fig 9. All transactions are negotiated with `Polyflow`

through the GraphQL API - one can refer to our Github repository[11] for the exact signatures of the proposed APIs).

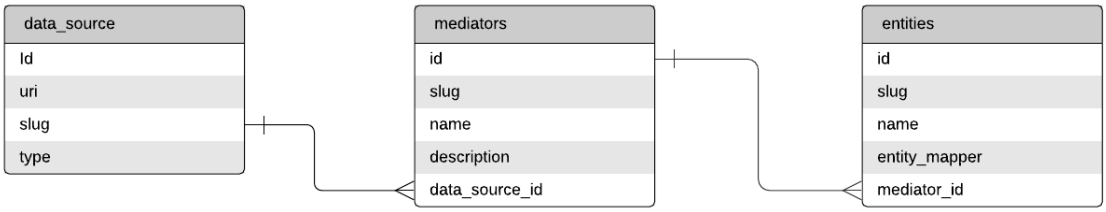

Fig. 9. Polyflow relational schema

Assuming an established connection with the 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 relationship, *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. 13(a) illustrates an example of ProvONE's *Program Entity Mapper* for Kepler's *Data Source*. This example showcases a $1-2$ relationship, *i.e.* an entity in the global schema is composed by 2 entities in the local schema. Note that this implies one level of recursion for this example, but Polyflow supports as many as needed (*i.e.* $1-N$). Fig. 13(b) showcases how Polyflow expands a query over all Kepler's *Programs*.

5. EVALUATION AND RESULTS

In this section we evaluate Polyflow in two different ways: (i) we assess the technical implementation in two dimensions: (a) *Completeness* (*i.e.*, Polyflow can query all provenance graphs across

[11] 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
```

Fig. 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.

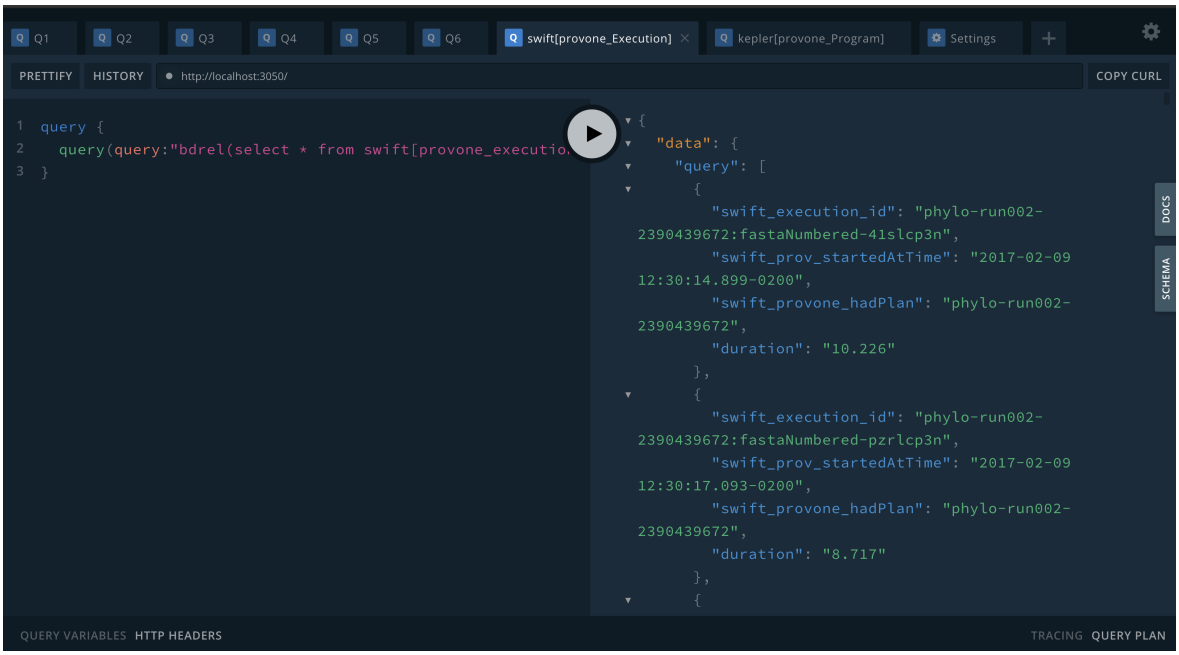

Fig. 11. Querying the Execution ProvONE entity in Swift/T's database.

```
{
  "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
```

Fig. 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.

```
SELECT
  *
FROM kepler[provone_program] AS p
JOIN swift[provone_execution] AS e
  ON e.provone_hadPlan = p.program_id

<=>
```

```
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 p
  )
  JOIN (
    SELECT
      app_exec_id AS execution_id,
      start_time AS prov_startedAtTime,
      script_run_id AS provone_hadPlan,
      duration
    FROM app_exec
  ) AS e
  ON p.program_id = e.provone_hadPlan
)
```

Fig. 13. (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.

distributed databases); (b) *Efficiency* (*i.e.,* `Polyflow` adds an acceptable overhead to query the provenance databases); (ii) conduct a pilot experiment with an expert researcher, which uses computational models empowered by workflows, to evaluate and grant reproducibility to her findings.

5.1 Evaluation Context

In the experimental evaluation, we used the *Rede Avançada em Biologia Computacional (Rabicó)*[12], 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. It has seven well-defined steps (Figure 14): (i) *Import Datasets*: data is collected from different biological repositories, such as RefSeq[13]; (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.*, 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 article, 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

[12] `https://www.labinfo.lncc.br/rabico/`
[13] `https://www.ncbi.nlm.nih.gov/refseq/`

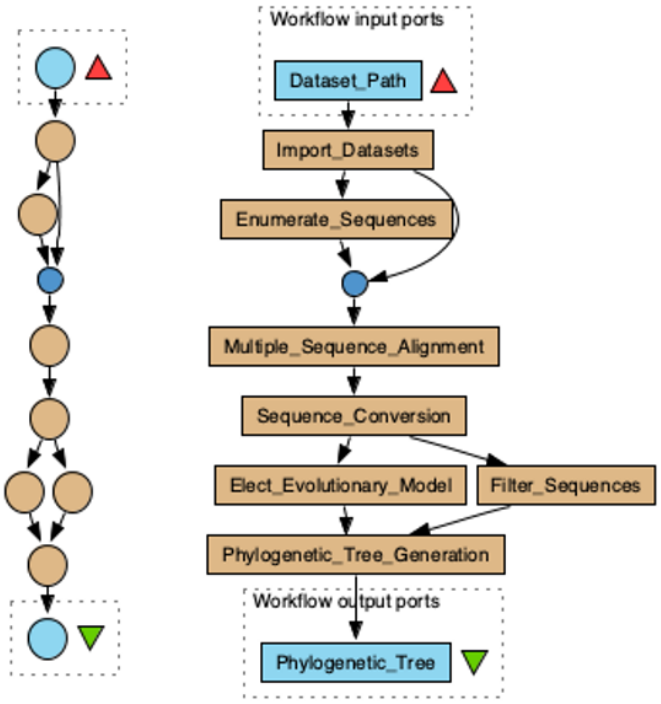

Fig. 14. Specification of the phylogenetic analysis experiment (represented using Apache Taverna notation)

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 case study provided in the previous subsection, we assume that the expected outcome of a feasibility study would be to allow researchers to *query integrated provenance data* generated by SciPhy and SwiftPhylo workflows. We use ProvONE as Global Schema because (i) it is able to capture p-prov, r-prov and evolution provenance; (ii) it is an extension of Prov-DM W3C standard for r-prov representation; and (iii) 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 to showcase every possible querying scenario using two provenance graphs. Their work inspires this evaluation, and, due to space limitations, we limit ourselves to three queries: (Q1) p-prov using one provenance graph; (Q2) r-prov using one provenance graph; and (Q3) p-prov across two provenance graphs. A working example with other queries can be obtained in the Projects Github repository[14]. The aforementioned queries can be implemented as:

—Q1 - List all SciPhy ports (connection between programs and their input/output parameters)

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

[14]`https://github.com/UFFeScience/Polyflow/tree/master/examples/BigDAWG`

Table VI. Tuple count at every touch point through the architecture.

Query	C4	C3	C2	C1
Q1	0	11	11	11
Q2	3000	0	3000	3000
Q3	2	11	22	22

—Q2 - Retrieve all activity executions with their generated data for SwiftPhylo provenance graph

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

—Q3 - Check if the all ports in SciPhy workflow have an equivalent on in SwiftPhylo workflow

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

In terms of *Completeness*, we first assessed that the returned tuples were equivalent for each query in every configuration: (C1) using `Polyflow`, (C2) using only BigDAWG, (C3) using only Kepler provenance database in the underlying DBMS and (C4) using only Swift/T provenance database in the underlying DBMS. The results are synthesized in Table VI, 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].

As explained in Section 4, when a query is submitted, `Polyflow` retrieves the *Entity Mappers* and performs a query expansion using entities and relationships from the Local Schema. 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 VII 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 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 repository[15]. `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*.

5.3 Pilot Experiment

Besides the feasibility study, we also carried out an evaluation with an expert in the bioinformatics domain. This pilot experiment aimed to assess the viability of `Polyflow` from the perspective of a

[15] `https://github.com/bigdawg-istc/bigdawg/issues`

Table VII. `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	123.0	196.0	157.3	74.5	118.0	189.0	151.6	72.2	10.0	4.0	5.7	5.8	**3.8**
Q2	274.0	1274.0	496.3	899.3	271.0	1270.0	489.6	901.7	16.0	3.0	6.7	11.7	**1.4**
Q3	393.0	594.0	472.0	177.5	386.0	590.0	466.3	179.9	8.0	4.0	5.7	4.0	**1.2**

user, *i.e.*, researchers.

The main objective of this experiment is to assess the semantic capabilities of `Polyflow`. In other words, we want to evaluate that `Polyflow` is capable of performing integrated queries over heterogeneous provenance graphs by using Global Schema's (ProvONE) entities in a more straightforward way than the current state of practice.

The subject of this pilot experiment is a member of Rabicó project and a researcher at LNCC. The questionnaire was answered on March 12th, 2020. All answers are available, in Portuguese, at this link[16]. Following we present a summary of the results of this pilot experiment:

—*Messy and convoluted logs and warning messages:* 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;

—*A viable and easy to perform integrated queries over provenance graphs*: After overcoming the problems above, the subject was able to query the provenance graphs and evaluated the solution as both viable and simpler than manually integrating the data.

With the authors' assistance, the subject was able to execute `Polyflow` and perform all the aforementioned queries. All issues reported by the subject were resolved, and we are still developing a more robust version of the experiment with several participants in the near future.

6. FINAL REMARKS AND FUTURE WORK

This article conducted a Systematic Literature Mapping to understand and compare state-of-the-art approaches and presented `Polyflow`. This polystore-compliant approach enables an integrated analysis of provenance graphs across multiple databases. We use ProvONE as a Canonical Conceptual Model/Global Schema, to which underlying provenance databases are queried at runtime. Real provenance databases, used by *Rede Avançada em Biologia Computacional (Rabicó)*, a Brazilian research network, were integrated using `Polyflow` to assess the proposal. The integration allows researchers to query, transparently, across multiple databases, providing a broader, unified analysis.

Results show that the overhead introduced by `Polyflow` is negligible compared to BigDAWG, a reference Polystore database. We also conducted a pilot experiment with an expert researcher that evaluated `Polyflow` and found it simple to use, as opposed to manually querying all data herself. For future work, we plan on expanding this pilot to a full-scale experiment with multiple participants to have more quantitative results.

REFERENCES

The third provenance challenge. `https://openprovenance.org/provenance-challenge/ThirdProvenanceChallenge.html`, 2009. [Online; accessed 24-August-2018].

Prov-overview. `https://www.w3.org/TR/prov-overview/`, 2013. [Online; accessed 24-August-2018].

[16] `https://bit.ly/2OEo4bw`

Abadi, D., Agrawal, R., Ailamaki, A., Balazinska, M., Bernstein, P. A., Carey, M. J., Chaudhuri, S., Dean, J., Doan, A., Franklin, M. J., Gehrke, J., Haas, L. M., Halevy, A. Y., Hellerstein, J. M., Ioannidis, Y. E., Jagadish, H. V., Kossmann, D., Madden, S., Mehrotra, S., Milo, T., Naughton, J. F., Ramakrishnan, R., Markl, V., Olston, C., Ooi, B. C., Ré, C., Suciu, D., Stonebraker, M., Walter, T., and Widom, J. The beckman report on database research. *Commun. ACM* 59 (2): 92–99, 2016.

Abadi, D., Ailamaki, A., Andersen, D., Bailis, P., Balazinska, M., Bernstein, P. A., Boncz, P. A., Chaudhuri, S., Cheung, A., Doan, A., Dong, L., Franklin, M. J., Freire, J., Halevy, A. Y., Hellerstein, J. M., Idreos, S., Kossmann, D., Kraska, T., Krishnamurthy, S., Markl, V., Melnik, S., Milo, T., Mohan, C., Neumann, T., Ooi, B. C., Ozcan, F., Patel, J., Pavlo, A., Popa, R. A., Ramakrishnan, R., Ré, C., Stonebraker, M., and Suciu, D. The seattle report on database research. *SIGMOD Rec.* 48 (4): 44–53, 2019.

Abbasi, A., Sarker, S., and Chiang, R. H. Big data research in information systems: Toward an inclusive research agenda. *Journal of the Association for Information Systems* 17 (2), 2016.

Altintas, I., Anand, M. K., Crawl, D., Bowers, S., Belloum, A., Missier, P., Ludäscher, B., Goble, C. A., and Sloot, P. M. Understanding collaborative studies through interoperable workflow provenance. In *IPAW*. Springer, pp. 42–58, 2010.

Altintas, I., Berkley, C., Jaeger, E., Jones, M. B., Ludäscher, B., and Mock, S. 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*. pp. 423–424, 2004.

Anand, M. K., Bowers, S., Altintas, I., and Ludäscher, B. Approaches for exploring and querying scientific workflow provenance graphs. In *International Provenance and Annotation Workshop*. Springer, pp. 17–26, 2010.

Anand, M. K., Bowers, S., Mcphillips, T., and Ludäscher, B. Exploring scientific workflow provenance using hybrid queries over nested data and lineage graphs. In *SSDBM*. Springer, pp. 237–254, 2009.

Asuncion, C. H. and van Sinderen, M. Towards pragmatic interoperability in the new enterprise—a survey of approaches. In *International IFIP Working Conference on Enterprise Interoperability*. Springer, pp. 132–145, 2011.

Atkinson, M. P., Gesing, S., Montagnat, J., and Taylor, I. J. Scientific workflows: Past, present and future. *FGCS* vol. 75, pp. 216–227, 2017.

Bavoil, L., Callahan, S. P., Scheidegger, C. E., Vo, H. T., Crossno, P., Silva, C. T., and Freire, J. Vistrails: Enabling interactive multiple-view visualizations. In *16th IEEE Visualization Conference, VIS 2005, Minneapolis, MN, USA, October 23-28, 2005*. IEEE Computer Society, pp. 135–142, 2005.

Begoli, E., Kistler, D., and Bates, J. Towards a heterogeneous, polystore-like data architecture for the US department of veteran affairs (VA) enterprise analytics. In *BigData 2016*. pp. 2550–2554, 2016.

Bose, R. and Frew, J. Lineage retrieval for scientific data processing: a survey. *ACM Computing Surveys (CSUR)* 37 (1): 1–28, 2005.

Budgen, D., Turner, M., Brereton, P., and Kitchenham, B. A. Using mapping studies in software engineering. In *PPIG*. Vol. 8. pp. 195–204, 2008.

Buneman, P., Khanna, S., and Wang-Chiew, T. Why and where: A characterization of data provenance. In *ICDT*. Springer, pp. 316–330, 2001.

Chebotko, A., Lu, S., Fei, X., and Fotouhi, F. Rdfprov: A relational rdf store for querying and managing scientific workflow provenance. *Data & Knowledge Engineering* 69 (8): 836–865, 2010.

Chirigati, F. and Freire, J. Provenance and reproducibility. In *Encyclopedia of Database Systems, Second Edition*, L. Liu and M. T. Özsu (Eds.). Springer, 2018.

Costa, C. and Murta, L. Version control in distributed software development: A systematic mapping study. In *2013 ICGSE*. IEEE, pp. 90–99, 2013.

Coulouris, G. F., Dollimore, J., and Kindberg, T. *Distributed systems: concepts and design*. pearson education, 2005.

Cuevas-Vicenttin, V., Dey, S., Wang, M. L. Y., Song, T., and Ludascher, B. Modeling and querying scientific workflow provenance in the d-opm. In *2012 SC Companion: High-Performance Computing, Networking, Storage and Analysis (SCC)*. IEEE, pp. 119–128, 2012.

Cuevas-Vicenttín, V., Ludäscher, B., Missier, P., Belhajjame, K., Chirigati, F., Wei, Y., and Leinfelder, B. Provone: A prov extension data model for scientific workflow provenance, 2015.

Cuevas-Vicenttín Víctor, Ludäscher Bertram, M. P. B. K. C. F. W. Y. D. S. K. P. K. D. B. S. A. I. J. C. B. J. M. W. L. S. P. L. B. C. Y. Provone data model. `http://jenkins-1.dataone.org/jenkins/view/Documentation%20Projects/job/ProvONE-Documentation-trunk/ws/provenance/ProvONE/v1/provone.html`, 2016. [Online; accessed 19-July-2020].

Davidson, S. B. and Freire, J. Provenance and scientific workflows: challenges and opportunities. In *2008 ACM SIGMOD*. ACM, pp. 1345–1350, 2008.

de Oliveira, A. H. M., de Oliveira, D., and Mattoso, M. Clouds and reproducibility: A way to go to scientific experiments? In *Cloud Computing - Principles, Systems and Applications, Second Edition*, N. Antonopoulos and L. Gillam (Eds.). Computer Communications and Networks. Springer, pp. 127–151, 2017.

de Oliveira, D., Liu, J., and Pacitti, E. *Data-Intensive Workflow Management: For Clouds and Data-Intensive and Scalable Computing Environments*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2019.

de Oliveira, D., Ocaña, K. A. C. S., Ogasawara, E. S., Dias, J., de A. R. Gonçalves, J. C., Baião, F. A., and Mattoso, M. Performance evaluation of parallel strategies in public clouds: A study with phylogenomic workflows. *Future Gener. Comput. Syst.* 29 (7): 1816–1825, 2013.

de Oliveira, D., Ogasawara, E. S., Baião, F. A., and Mattoso, M. 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*. pp. 378–385, 2010.

Deelman, E., Peterka, T., Altintas, I., Carothers, C. D., van Dam, K. K., Moreland, K., Parashar, M., Ramakrishnan, L., Taufer, M., and Vetter, J. S. The future of scientific workflows. *Int. J. High Perform. Comput. Appl.* 32 (1): 159–175, 2018.

Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P., Mayani, R., Chen, W., da Silva, R. F., Livny, M., and Wenger, R. K. Pegasus, a workflow management system for science automation. *Future Generation Comp. Syst.* vol. 46, pp. 17–35, 2015.

Ding, L., Michaelis, J., McCusker, J., and McGuinness, D. L. Linked provenance data: A semantic web-based approach to interoperable workflow traces. *Future Generation Computer Systems* 27 (6): 797–805, 2011.

Duggan, J., Elmore, A. J., Stonebraker, M., Balazinska, M., Howe, B., Kepner, J., Madden, S., Maier, D., Mattson, T., and Zdonik, S. The bigdawg polystore system. *ACM Sigmod Record* 44 (2): 11–16, 2015.

Ellqvist, T., Koop, D., Freire, J., Silva, C., and Strömbäck, L. Using mediation to achieve provenance interoperability. In *Services-I, 2009 World Conference on*. IEEE, pp. 291–298, 2009.

Freire, J. and Chirigati, F. S. Provenance and the different flavors of reproducibility. *IEEE Data Eng. Bull.* 41 (1): 15–26, 2018.

Freire, J., Koop, D., Santos, E., and Silva, C. T. Provenance for computational tasks: A survey. *CS & E* 10 (3), 2008.

Gadepally, V., Chen, P., Duggan, J., Elmore, A., Haynes, B., Kepner, J., Madden, S., Mattson, T., and Stonebraker, M. The bigdawg polystore system and architecture. In *High Performance Extreme Computing Conference (HPEC), 2016 IEEE*. IEEE, pp. 1–6, 2016.

Gaspar, W., Braga, R., and Campos, F. Sciprov: an architecture for semantic query in provenance metadata on e-science context. In *International Conference on Information Technology in Bio-and Medical Informatics*. Springer, pp. 68–81, 2011.

Gesing, S., Dooley, R., Pierce, M. E., Krüger, J., Grunzke, R., Herres-Pawlis, S., and Hoffmann, A. Gathering requirements for advancing simulations in HPC infrastructures via science gateways. *FGCS* vol. 82, pp. 544–554, 2018.

Glatard, T., Étienne Rousseau, M., Camarasu-Pop, S., Adalat, R., Beck, N., Das, S., da Silva, R. F., Khalili-Mahani, N., Korkhov, V., Quirion, P.-O., Rioux, P., Olabarriaga, S. D., Bellec, P., and Evans, A. C. Software architectures to integrate workflow engines in science gateways. *Future Generation Computer Systems* vol. 75, pp. 239 – 255, 2017.

Groth, P. and Moreau, L. Recording process documentation for provenance. *IEEE TPDS* 20 (9): 1246–1259, 2009.

Hamadou, H. B., Gallinucci, E., and Golfarelli, M. Answering GPSJ queries in a polystore: A dataspace-based approach. In *Conceptual Modeling - 38th International Conference, ER 2019, Salvador, Brazil, November 4-7, 2019, Proceedings*, A. H. F. Laender, B. Pernici, E. Lim, and J. P. M. de Oliveira (Eds.). Lecture Notes in Computer Science, vol. 11788. Springer, pp. 189–203, 2019.

Hazen, B. T., Boone, C. A., Ezell, J. D., and Jones-Farmer, L. A. 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* vol. 154, pp. 72–80, 2014.

Hey, T., Tansley, S., Tolle, K. M., et al. *The fourth paradigm: data-intensive scientific discovery*. Vol. 1. Microsoft research Redmond, WA, 2009.

Huynh, T. D., Ebden, M., Fischer, J. E., Roberts, S. J., and Moreau, L. Provenance network analytics - an approach to data analytics using data provenance. *Data Min. Knowl. Discov.* 32 (3): 708–735, 2018.

Jabal, A. A. and Bertino, E. Simp: Secure interoperable multi-granular provenance framework. In *IEEE e-Science 2016*. IEEE, pp. 270–275, 2016.

Jagadish, H. V., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J. M., Ramakrishnan, R., and Shahabi, C. Big data and its technical challenges. *Commun. ACM* 57 (7): 86–94, 2014.

Jhingran, A., Mattos, N., and Pirahesh, H. Information integration: A research agenda. *IBM systems Journal* 41 (4): 555–562, 2002.

Khan, F. Z., Soiland-Reyes, S., Sinnott, R. O., Lonie, A., Goble, C., and Crusoe, M. R. Sharing interoperable workflow provenance: A review of best practices and their practical application in cwlprov. *GigaScience* 8 (11): giz095, 2019.

Khan, Y., Zimmermann, A., Jha, A., Gadepally, V., d'Aquin, M., and Sahay, R. One size does not fit all: Querying web polystores. *IEEE Access* vol. 7, pp. 9598–9617, 2019.

Kitchenham, B. Procedures for performing systematic reviews. *Keele, UK, Keele University* 33 (2004): 1–26, 2004.

Li, C. and Sugimoto, S. Provenance description of metadata using prov with premis for long-term use of metadata. In *International Conference on Dublin Core and Metadata Applications*. pp. 147–156, 2014.

Lim, C., Lu, S., Chebotko, A., and Fotouhi, F. Storing, reasoning, and querying opm-compliant scientific workflow provenance using relational databases. *FGCS* 27 (6): 781–789, 2011.

Litwin, W. and Abdellatif, A. Multidatabase interoperability. *Computer* (12): 10–18, 1986.

Mattoso, M., Werner, C., Travassos, G. H., Braganholo, V., Ogasawara, E. S., de Oliveira, D., da Cruz, S. M. S., Martinho, W., and Murta, L. Towards supporting the life cycle of large scale scientific experiments. *IJBPIM* 5 (1): 79–92, 2010.

Mendes, Y., Ströele, V., de Oliveira, D., and Ocaña, K. Análise integrada de grafos de proveniência heterogêneos por meio de uma abordagem polystore. In *Anais Principais do XXXIV Simpósio Brasileiro de Banco de Dados*. SBC, Porto Alegre, RS, Brasil, pp. 73–84, 2019.

Missier, P., Ludäscher, B., Bowers, S., Dey, S., Sarkar, A., Shrestha, B., Altintas, I., Anand, M. K., and Goble, C. Linking multiple workflow provenance traces for interoperable collaborative science. In *WORKS 2010*. IEEE, pp. 1–8, 2010.

Mondelli, M. L., Magalhães, T., Loss, G., Wilde, M., Foster, I. T., Mattoso, M., Katz, D. S., Barbosa, H. J. C., de Vasconcelos, A. T. R., Ocaña, K. A. C. S., and Jr., L. M. R. G. Bioworkbench: A high-performance framework for managing and analyzing bioinformatics experiments. *CoRR* vol. abs/1801.03915, 2018.

Moreau, L., Freire, J., Futrelle, J., McGrath, R. E., Myers, J., and Paulson, P. The open provenance model: An overview. In *International Provenance and Annotation Workshop*. Springer, pp. 323–326, 2008.

Moreau, L., Groth, P. T., Cheney, J., Lebo, T., and Miles, S. The rationale of PROV. *J. Web Semant.* vol. 35, pp. 235–257, 2015.

Ocaña, K. A., de Oliveira, D., Ogasawara, E., Dávila, A. M., Lima, A. A., and Mattoso, M. Sciphy: a cloud-based workflow for phylogenetic analysis of drug targets in protozoan genomes. In *BSB11*. Springer, pp. 66–70, 2011.

Ogasawara, E. S., Dias, J., Sousa, V. S., Chirigati, F. S., de Oliveira, D., Porto, F., Valduriez, P., and Mattoso, M. Chiron: a parallel engine for algebraic scientific workflows. *Concurr. Comput. Pract. Exp.* 25 (16): 2327–2341, 2013.

Oliveira, W., Missier, P., Ocaña, K., de Oliveira, D., and Braganholo, V. Analyzing provenance across heterogeneous provenance graphs. In *IPAW*. Springer, pp. 57–70, 2016.

Özsu, M. T. and Valduriez, P. *Principles of distributed database systems*. Springer Science & Business Media, 2011.

Parciak, M., Bauer, C., Bender, T., Lodahl, R., Schreiweis, B., Tute, E., and Sax, U. Provenance solutions for medical research in heterogeneous it-infrastructure: An implementation roadmap. *Studies in health technology and informatics* vol. 264, pp. 298–302, 2019.

Peng, R. The reproducibility crisis in science: A statistical counterattack. *Significance* 12 (3): 30–32, 2015.

Pérez, B., Rubio, J., and Sáenz-Adán, C. A systematic review of provenance systems. *Knowledge and Information Systems*, 2018.

Prabhune, A., Stotzka, R., Sakharkar, V., Hesser, J., and Gertz, M. Metastore: an adaptive metadata management framework for heterogeneous metadata models. *DPD* 36 (1): 153–194, 2018.

Prabhune, A., Zweig, A., Stotzka, R., Gertz, M., and Hesser, J. Prov2one: an algorithm for automatically constructing provone provenance graphs. In *IPAW*. Springer, pp. 204–208, 2016.

Prabhune, A., Zweig, A., Stotzka, R., Hesser, J., and Gertz, M. P-PIF: a provone provenance interoperability framework for analyzing heterogeneous workflow specifications and provenance traces. *Distributed and Parallel Databases* 36 (1): 219–264, 2018a.

Prabhune, A., Zweig, A., Stotzka, R., Hesser, J., and Gertz, M. P-pif: a provone provenance interoperability framework for analyzing heterogeneous workflow specifications and provenance traces. *DPD* 36 (1): 219–264, 2018b.

Schwab, M., Karrenbach, N., and Claerbout, J. Making scientific computations reproducible. *CS & E* 2 (6): 61–67, 2000.

Simmhan, Y. L., Plale, B., and Gannon, D. A survey of data provenance in e-science. *ACM Sigmod Record* 34 (3): 31–36, 2005.

Souza, R., Azevedo, L., Thiago, R., Soares, E., Nery, M., Netto, M., Brazil, E. V., Cerqueira, R., Valduriez, P., and Mattoso, M. Efficient runtime capture of multiworkflow data using provenance, 2019.

Steinmacher, I., Chaves, A. P., and Gerosa, M. A. Awareness support in distributed software development: A systematic review and mapping of the literature. *CSCW* 22 (2-3): 113–158, 2013.

Tolk, A. and Muguira, J. A. The levels of conceptual interoperability model. In *Proceedings of the 2003 fall simulation interoperability workshop*. Vol. 7. Citeseer, pp. 1–11, 2003.

Watson, P., Hiden, H., and Woodman, S. e-science central for CARMEN: science as a service. *Concurrency and Computation: Practice and Experience* 22 (17): 2369–2380, 2010.

Wegner, P. Interoperability. *ACM Computing Surveys (CSUR)* 28 (1): 285–287, 1996.

Wiederhold, G. Mediators in the architecture of future information systems. *Computer* 25 (3): 38–49, 1992.

Wohlin, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. ACM, pp. 38, 2014.

Wolstencroft, K., Haines, R., Fellows, D., Williams, A. R., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., Bhagat, J., Belhajjame, K., Bacall, F., Hardisty, A., de la Hidalga, A. N., Vargas, M. P. B., Sufi, S., and Goble, C. A. The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic Acids Research* 41 (Webserver-Issue): 557–561, 2013.

Wozniak, J. M., Armstrong, T. G., Wilde, M., Katz, D. S., Lusk, E. L., and Foster, I. 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*. pp. 95–102, 2013.

Yu, J. and Buyya, R. A taxonomy of scientific workflow systems for grid computing. *ACM Sigmod Record* 34 (3): 44–49, 2005.