Future Generation Computer Systems 27 (2011) 781–789

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Storing, reasoning, and querying OPM-compliant scientific workflow provenance using relational databases

Chunhyeok Lim [a], Shiyong Lu [a,*], Artem Chebotko [b], Farshad Fotouhi [a]

[a] Department of Computer Science, Wayne State University, Detroit, MI 48202, USA

[b] Department of Computer Science, University of Texas-Pan American, Edinburg, TX 78539, USA

ARTICLE INFO

Article history:
Received 15 December 2009
Received in revised form
20 October 2010
Accepted 25 October 2010
Available online 4 November 2010

Keywords:
Provenance
Scientific workflow
Metadata management
OPM
OPM-compliant provenance storage
RDBMS

ABSTRACT

Provenance, the metadata that records the derivation history of scientific results, is essential in scientific workflows to support the reproducibility of scientific discovery, result interpretation, and problem diagnosis. To promote and facilitate interoperability among heterogeneous provenance systems, the Open Provenance Model (OPM) was first proposed in 2008 and since then has played an important role in the community. In this paper, we present *OPMProv*, a relational database-based scientific workflow provenance system, that is compliant with OPM (v1.1). Our main contributions are: (i) we design an entity–relationship diagram for OPM and translate it into a relational database schema for the storage of provenance; (ii) we show that provenance reasoning defined in OPM (v1.1) can be sufficiently supported by *OPMProv* using recursive views and SQL queries alone without any additional reasoning engine. Experiments are conducted to evaluate the performance of *OPMProv* in data insertion and provenance querying. A case study is performed, demonstrating that *OPMProv* can answer all except two queries out of the 16 queries defined in the Third Provenance Challenge.

1. Introduction

Scientific workflows have become an increasingly popular paradigm for scientists to formalize and structure complex scientific processes to enable and accelerate many significant scientific discoveries [1–6]. A scientific workflow is a formal specification of a scientific process, which represents, streamlines, and automates the analytical and computational steps that a scientist needs to go through from dataset selection and integration, computation and analysis, to final data product presentation and visualization [7]. The importance of scientific workflows has been recognized by NSF since 2006 [8] and was reemphasized in a recent science article [9], which concluded, "In the future, the rapidity with which any given discipline advances is likely to depend on how well the community acquires the necessary expertise in database, *workflow management*, visualization, and cloud computing technologies".

Scientific workflow provenance captures the derivation history of a data product, including the sources, intermediate data products, and the steps that were applied to produce the data product. Provenance is essential for scientific workflows to support reproducibility of scientific discovery, result interpretation, and problem diagnosis [10,11]. Although numerous provenance systems [12–17,7,18] have been developed, their interoperability is poor due to the lack of a common data model for provenance. To address this issue, the Open Provenance Model (OPM) [19] was proposed in 2008. Since then, OPM has played an important role in provenance interoperability and has had a positive impact on ongoing provenance activities, including the IPAW workshops [20] and the Provenance Challenges [21].

While there is a growing effort in supporting OPM in existing provenance systems [12–17] and the evaluation of OPM in a particular domain [22,23], such as the scientific workflow domain, most of them focus on enhancing an existing provenance system with the import/export capability for OPM. In this paper, however, we take OPM as a starting point and develop a native OPM provenance store. By native, we mean that OPM is the conceptual data model that is used to design our provenance store and the input and output of such a store is OPM-compliant provenance data. Therefore, our work complements the existing work whose OPM support is based on back and forth transformations between the OPM model and proprietary models employed by these systems. Although using OPM as an implementation schema belongs to one of the non-requirements defined in OPM, an OPM-based provenance system can be useful for a scientific workflow whose workflow tasks are subworkflows enacted by different scientific workflow management systems. An example of such a workflow is *GENOMEFLOW* [24]. In this scenario, provenance from

* Corresponding author. Tel.: +1 313 577 1667; fax: +1 313 577 6868.
E-mail addresses: chlim@wayne.edu (C. Lim), shiyong@wayne.edu (S. Lu), artem@cs.panam.edu (A. Chebotko), fotouhi@wayne.edu (F. Fotouhi).

0167-739X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2010.10.013

different scientific workflow management systems needs to be integrated, and our *OPMProv* system can be used for this purpose. *OPMProv* is fully compliant with OPM and can store provenance generated by different scientific workflow management systems that are able to record OPM-compliant provenance. In particular, in the Third Provenance Challenge [25], different scientific workflow management systems, including Kepler, Taverna, and Swift, have shown the capability to export OPM-compliant provenance data by means of a mapping between the proprietary models and the OPM model; such heterogeneous provenance from different workflow management systems can be integrated in *OPMProv*. In our work, we are particularly interested in using relational database technologies to store and query OPM-compliant provenance data. Since relational databases are not specifically designed for inferences, we aim to investigate if we can use recursive views and SQL queries alone to perform provenance reasoning.

This paper has the following main contributions: (1) we design an entity–relationship diagram for OPM (v1.1) and translate it into a relational database schema for the storage of provenance; (2) we show provenance reasoning defined in OPM (v1.1) can be sufficiently supported by *OPMProv* using recursive views and SQL queries alone without any additional reasoning engine. Experiments are conducted to evaluate the performance of *OPMProv* in data insertion and provenance querying. A case study is performed, demonstrating that *OPMProv* can answer all except two queries out of the 16 queries defined in the Third Provenance Challenge [25].

2. Provenance storage

In this section, we design an entity–relationship diagram for OPM, translate it into a relational database schema for the storage of provenance, and discuss how OPM-compliant provenance data in the XML format can be efficiently inserted into the proposed database schema.

2.1. Entity–relationship diagram

Based on the Open Provenance Model [26], we design an E–R diagram that captures the conceptual model of our provenance storage. The E–R diagram is shown in Fig. 1 and depicts three entity types *Process*, *Artifact*, and *Agent* and five relationship types *Used*, *WasGeneratedBy*, *WasControlledBy*, *WasTriggeredBy*, and *WasDerivedFrom*. These entity and relationship types have direct counterparts in OPM. According to OPM, artifacts, processes, and agents are identified by unique identifiers and the causal dependency edges are identified by their sources, destinations, and roles (for those that have roles) [26]. Thus, in the E–R diagram, each entity type has a primary key attribute that is underlined and each relationship type has a composite primary key that includes the two roles[1] of the relationship with participating entity types and the *Role* attribute (for those that have the *Role* attribute). For example, the primary key of relationship type *Used* is *ProcessId*, *ArtifactId*, and *Role* and relationship type *WasDerivedFrom* takes *ArtifactId* for the *Effect* role and *ArtifactId* for the *Cause* role as the primary key. Each entity type has attributes *Value* and *Account*; the latter is a set-valued attribute, such that a process, artifact, or agent can have multiple accounts. The relationship types have set-valued *Account* attributes and composite *OTime* attributes (*OTimeStart* and *OTimeEnd* for relationship type *WasControlledBy*). Composite attribute *OTime* is composed of the *OTimeLower* and *OTimeUpper* attributes which is consistent with the *OTime* annotation in OPM. The *WasControlledBy* relationship type has two composite attributes *OTimeStart* and *OTimeEnd* that are composed of (*OTimeStartLower*, *OTimeStartUpper*) and (*OTimeEndLower*, *OTimeEndUpper*), respectively. Finally, each entity/relationship type has a set-valued and composite attribute *Annotation*, such that *Process*, *Artifact*, *Agent*, *Used*, *WasGeneratedBy*, *WasControlledBy*, *WasTriggeredBy*, and *WasDerivedFrom* can have multiple annotations consisting of property–value pairs defined in OPM.

2.2. Database schema

We translate our proposed E–R diagram into the database schema that can be used for storing, reasoning, and querying the OPM-compliant provenance data. As shown in Fig. 2, we identify 29 relations, where the first 24 of them are materialized relations and the remaining five are non-materialized views. Relations *Artifact*, *Process*, *Agent*, *Used*, *WasGeneratedBy*, *WasControlledBy*, *WasDerivedFrom*, and *ExplicitWasTriggeredBy*[2] are directly derived from the E–R diagram, however, to handle the set-valued attributes *Account* and *Annotation*, additional relations are introduced, such as the corresponding relations *xxxHasAccount* and *xxxAnnotation*. For each relation, we introduce an OPM graph identifier (i.e., attribute *OPMGraphId*) to identify multiple workflow runs for a workflow. Being able to store multiple OPM graphs into the same schema is an important characteristic of the provenance storage system that allows querying and analysis of provenance recorded by multiple executions of the same or different scientific workflows. We also restrict that each row in relations *Artifact*, *Process*, *Agent*, *Used*, etc. has at least one account and therefore at least one row in the corresponding *xxxHasAccount* relations. This participation constraint eliminates the burden of dealing with missing values when computing relational joins and can be efficiently ensured on the data insertion stage by introducing a default account. The primary keys of these 24 relations are depicted in Fig. 2. For example, *(OPMGraphId, ArtifactId)* is the composite primary key of the *Artifact* relation, and the *ArtifactHasAccount* relation has *(OPMGraphId, ArtifactId, Account)* as the primary key and *(OPMGraphId, ArtifactId)* as the foreign key referencing the *Artifact* relation. Similarly, the *ArtifactAnnotation* relation has the primary key *(OPMGraphId, ArtifactId, Property, Value)* since a same property can have multiple values as defined in OPM, and it has the foreign key *(OPMGraphId, ArtifactId)* referencing the *Artifact* relation. Non-materialized views in our database schema are defined as follows. While view *WasTriggeredBy* (see Fig. 3(a)) implements the one-step inference (i.e., completion rule) defined in OPM [26], views *MultiStepWasDerivedFrom*, *MultiStepWasTriggeredBy*, *MultiStepUsed*, and *MultiStepWasGeneratedBy* (see Fig. 3(b) and (c)) implement the multi-step inferences (i.e., multi-step versions of existing edges) presented in OPM [26]. The semantics and implementation of these recursive views are further discussed in Section 3.

2.3. Data insertion

To insert provenance data represented in XML into *OPMProv*, we employ a data mapping procedure that shreds XML documents, which conform to the XML schema specification for OPM [27], into

[1] Note that the term "role" is used in both E–R Diagram and OPM, but they have slightly different meanings: roles in an E–R diagram represent the participation relationships between an entity type and a relationship type, while roles in OPM represent annotations on edges *Used*, *WasGeneratedBy*, and *WasControlledBy*.

[2] Note that this relation corresponds to relationship type *WasTriggeredBy* in the E–R diagram, but we name it *ExplicitWasTriggeredBy* to differentiate from non-materialized view *WasTriggeredBy* which can be inferred from relations *Used*, *WasGeneratedBy*, and *ExplicitWasTriggeredBy*.