Data & Knowledge Engineering 88 (2013) 37–59

Contents lists available at ScienceDirect

Data & Knowledge Engineering

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

Editorial

OPQL: Querying scientific workflow provenance at the graph level

Chunhyeok Lim [a], Shiyong Lu [a,*], Artem Chebotko [b], Farshad Fotouhi [a], Andrey Kashlev [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 21 December 2011
Received in revised form 30 August 2013
Accepted 31 August 2013
Available online 12 September 2013

Keywords:
OPQL
Provenance query language
Scientific workflow provenance

ABSTRACT

Provenance has become increasingly important in scientific workflows to understand, verify, and reproduce the result of scientific data analysis. Most existing systems store provenance data in provenance stores with proprietary provenance data models and conduct query processing over the physical provenance storages using query languages, such as SQL, SPARQL, and XQuery, which are closely coupled to the underlying storage strategies. Querying provenance at such low level leads to poor usability of the system: a user needs to know the underlying schema to formulate queries; if the schema changes, queries need to be reformulated; and queries formulated for one system will not run in another system. In this paper, we present *OPQL*, a provenance query language that enables the querying of provenance directly at the graph level. An *OPQL* query takes a provenance graph as input and produces another provenance graph as output. Therefore, *OPQL* queries are not tightly coupled to the underlying provenance storage strategies. Our main contributions are: (i) we design *OPQL*, including six types of graph patterns, a provenance graph algebra, and *OPQL* syntax and semantics, that supports querying provenance at the graph level; (ii) we implement *OPQL* using a Web service via our OPMProv system; therefore, users can invoke the Web service to execute *OPQL* queries in a provenance browser, called OPMProVis. The result of *OPQL* queries is displayed as a provenance graph in OPMProVis. An experimental study is conducted to evaluate the feasibility and performance of OPMProv on *OPQL* provenance querying.

1. Introduction

Provenance, which is one kind of metadata that captures the derivation history of a data product, including its original data sources, intermediate products, and the steps that were applied to produce it, has become increasingly important in the area of scientific workflows [1-8] to interpret, validate, and analyze the result of scientific computing. In general, provenance captures past workflow execution and data derivation information (i.e., which tasks were performed and how data products were derived) via a provenance collection mechanism during workflow execution. Provenance captured typically holds data dependencies, process dependencies, causality between data and processes, and annotations. Such provenance is often represented by a provenance graph. For example, Fig. 1(a) shows a sample scientific workflow (which is the *Load Workflow* defined in the Third Provenance Challenge [9]) that checks and reads CSV files before loading, creates a database to load CSV files, loads them into tables and validates tables, and compacts a database after loading. Fig. 1(b) shows a sample provenance graph produced via the execution of the *Load Workflow*, where a node of a rectangle shape represents a process (i.e., a task), a node of an ellipse shape represents an artifact (i.e., a data product), which was used or generated by a process, a node of an octagon shape represents a contextual entity acting as a catalyst of a process, and an edge represents a causal dependency between its source denoting the

* 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), chebotkoa@utpa.edu (A. Chebotko), fotouhi@wayne.edu (F. Fotouhi), andrey.kashlev@wayne.edu (A. Kashlev).

0169-023X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.datak.2013.08.008

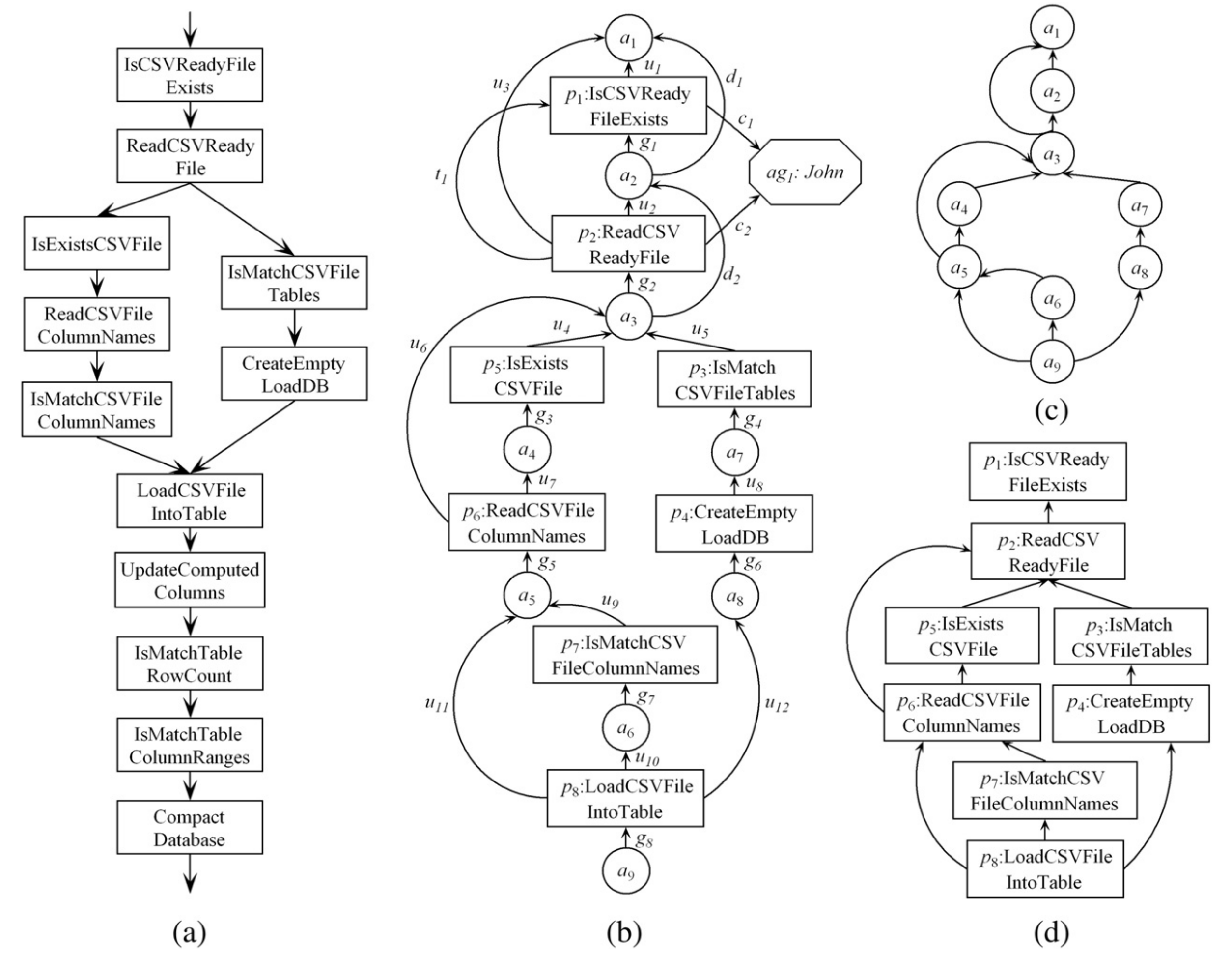

Fig. 1. An example of a scientific workflow and its provenance: (a) the *Load Workflow* defined in the Third Provenance Challenge; (b) a provenance graph generated via the execution of the *Load Workflow*; (c) a provenance graph representing data dependencies associated with artifact a$_9$; and (d) a provenance graph representing process dependencies associated with process p_8.

effect and its destination denoting the cause. Fig. 1(c) and (d) also represents a data dependency graph associated with artifact a_9 and a process dependency graph associated with process p_8 from the provenance graph, respectively.

Most existing systems [10,11,13,14] store provenance data in provenance stores with proprietary provenance data models and conduct provenance querying using languages such as SQL, SPARQL, and XQuery, over the physical provenance storages (i.e., relational, RDF, and XML). Such query languages are closely coupled to the underlying provenance storage strategies. As a result, querying provenance at such a low level leads to poor usability of the system because users need to know the underlying storage schema to formulate queries; if the schema changes, queries need to be reformulated; and queries formulated for one system will not run in another system. Moreover, to formulate complicated provenance queries, a user requires the expertise about grammars, syntax, and semantics of a query language. Using existing approaches, *provenance lineage queries* (queries for tracking ancestor nodes) often require a user to write recursive queries, directly typing recursive statements or using recursive functionality. For example, Fig. 2 shows two query languages SQL and *OPQL* answering for a provenance query (Q1), which asks for "which artifacts contributed to derive artifact a_5" over the provenance graph in Fig. 1(b). First, the SQL statement is expressed by a recursive query via the WITH ~ UNION ALL clause to track ancestor nodes associated with artifact a_5; thus, to answer query Q1, a user has to know the information of table *WasDerivedFrom* (i.e., how attributes define and what the attributes mean), a user needs the expertise to formulate a recursive query, which is nontrivial, and if the schema (i.e., the table information) changes, the SQL query needs to be reformulated, which is cumbersome. On the other hand, since the *OPQL* query is formulated at the graph level, a user does not have to know the storage schema; therefore, even though the storage schema changes, it does not affect the query. Moreover, *OPQL* construct WDF supports a recursive query pattern; thus, *OPQL* is easy and convenient to formulate a recursive query pattern.

In this paper, to address these issues, we propose *OPQL*, a provenance query language that enables the querying of provenance directly at the graph level. *OPQL* relies on the Open Provenance Model [15], a community-driven data model, which captures main aspects of the workflow provenance and does not enforce a particular physical representation of the provenance data. An *OPQL*